Linuxを搭載し単機能に特化したExpress5800インターネットアプライアンスサーバ

NEC（NECソリューションズ）は、1U（高さ44.5mm）サイズでLinuxを採用したインターネット専用サーバ「Express5800インターネットアプライアンスサーバ」5機種を、7月31日より発売した。5機種は、それぞれ単機能に特化しており、Webサーバ「Express5800/WebServer」、メールサーバ「同/MailServer」、ファイアウォールサーバ「同/FirewallServer」、負荷分散サーバ「同/LoadBalancer」、キャッシュサーバ「同/CacheServer」が用意されている。

/WebServer、/MailServer、/LoadBalancerの各機種は、CPUにCeleron 566MHz、メモリ128Mバイト、100Base-TXインターフェイスを2ポート、15Gバイトハードディスクを2台内蔵し、ソフトウェアミラーリング（二重化）で信頼性を向上させている。価格は、/WebServer、/MailServerが各49万8000円。/LoadBalancerが71万円。

なお、「Express5800インターネットアプライアンスサーバ」とルータやスイッチなどのネットワーク機器、管理ソフトウェアを同一ラックに搭載し、用途別に最適なシステムを構成した13種類の「Express5800インターネットサーバパック」も発売された。サーバパックには、システム導入時のセキュリティ診断、トラフィック診断を始め、導入後のシステム監視や運用代行などの総合的なサービスが提供される。

発売日
2000年7月31日
発売
日本電気株式会社
TEL
03-3456-6169
価格
49万8000円〜
URL
http://www.express.nec.co.jp/

Hardware　発売日　2000年7月31日

スリムタワー型のオールインワン・インターネットサーバ Express5800/SURFNAVI<Liteモデル>

URL　http://www.express.nec.co.jp/

発売　日本電気株式会社
TEL　03-3456-6169
価格　34万8000円〜

NEC（NECソリューションズ）は、省スペースのスリムタワー型ケースにインターネットに必要な機能をすべて搭載した、オールインワン型サーバ「Express5800/SURFNAVI<Liteモデル>」を、7月31日より発売した。

CPUにCeleron 566MHz、メモリ64Mバイト、15Gバイトハードディスク、40倍速CD-ROM、3.5インチフロッピードライブ、100Base-TXネットワークインターフェイスを2ポート搭載している。

Webサーバ、メールサーバ、DNSサーバ、FTPサーバ、キャッシュサーバ、ファイアウォール機能を備えている。インターネットにはダイヤルアップルータか専用線で接続する。各種システム設定は「マネージメントコンソール」を利用してWebブラウザから行うことができる。また、運用管理ツール「ESMPRO」が添付されている。

Hardware　発売日　2000年8月4日

1Uサイズの薄型ラックマウントサーバ HA8000/110

URL　http://www.hitachi.co.jp/ha8000/

発売　株式会社日立製作所
TEL　0120-2580-91
価格　23万2000円〜

日立製作所は、「HITACHI Advanced Server HA8000シリーズ」に、1U（高さ44.45mm）サイズの薄型ラックマウントサーバ「HA8000/110」を追加し、8月4日より発売した。

HA8000/110は、CPUにPentiumIII800／650MHzまたはCeleron 566MHz、メモリ最大1Gバイト、ハードディスクは2台内蔵可能で、Ultra160 SCSIで最大36Gバイト、IDEで最大20Gバイトまで搭載することができる。ネットワークは10/100BASE-TXを2ポート。OSはWindows 2000／NT 4.0、Linuxに対応する。

価格は、最小構成のCeleron 566MHz、10GバイトIDEハードディスク、メモリ／OSなしのモデルで23万2000円。

Hardware　発売日　2000年8月7日

1UラックマウントタイプにPentiumIII866MHzを搭載 Raxys-CS 1U GVシリーズ

URL　http://www.logicaleffect.com/

発売　ロジカルイフェクト株式会社
TEL　03-5822-3322
価格　18万9700円〜

ロジカルイフェクトは、高さ1Uサイズのラックマウントケースを採用した「Raxys-CS 1U GVシリーズ」を8月7日より発売した。

ケースはGenesysRack社との共同で開発したもので、1Uサイズでは世界最高のPentiumIII 866MHzで動作が可能になった。チップセットはVIA PM133を採用し、FSB 133MHzに対応している。グラフィックスにSavage4、メモリはオプションでECC対応、最大512Mバイトまで拡張可能。ハードディスク2台まで装着可能で、RAIDコントローラを装着してRAID 0／1に対応したモデルも用意されている。

価格は、CPUにPentiumIII600EB MHz、128MバイトSDRAM、20GバイトIDEハードディスク、24倍速スリムCD-ROM、フロッピードライブ、オンボードに10/100Base-TXネットワークインターフェイスを搭載したモデルで18万9700円。3万円の追加でCPUをPentiumIII866MHzに変更可能。

Hardware　発売日　2000年7月31日

AlphaStation XP900／XP1000・Kondara MNU/Linux SP -Optimized for Alpha Systems-プリインストールモデル

URL　http://q.sse.co.jp/

発売　住商エレクトロニクス株式会社
TEL　03-5228-5620
価格　55万5000円〜

住商エレクトロニクスは、デジタルファクトリとコンパックコンピュータが共同開発した「Kondara MNU/Linux SP -Optimized for Alpha Systems-」をプレインストールしたAlphaStationの提携記念モデルを7月31日より発売した。今回発売されたのは2モデルで、OSサポートはデジタルファクトリが、ハードウェアサポートはコンパックが担当する。

価格は、Alpha21264（2Mバイトキャッシュ）466MHz、256Mバイトメモリ、9GバイトSCSIハードディスクを搭載したAlphaStation XP900をベースにしたモデルが55万5000円。Alpha21264（4Mバイトキャッシュ）667MHz、512Mバイトメモリ、18GバイトSCSIハードディスクを搭載したAlphaStation XP1000をベースにしたモデルが88万8000円。

両モデルとも、コンパックの協力で最適化が図られたKondara MNU/Linux SP -Optimized for Alpha Systems-と、Fortran V1.0がプレインストールされるほか、Alpha用最適化コンパイラ「Compaq C V6.2」と、日本語全文検索ソフトウェア「MitakeSearch V2.0」がバンドルされる。

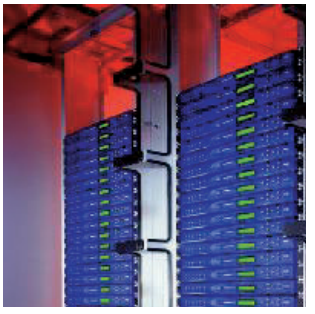
AlphaStation XP900

Hardware 発売日 2000年8月21日

1Uサイズの高機能Webサーバアプライアンス Cobalt RaQ4シリーズ

URL http://japan.cobalt.com/

発売 コバルト・ネットワークス株式会社
TEL 03-3599-0722
価格 オープンプライス



コバルト・ネットワークスは8月21日に、ラックマウントタイプのサーバアプライアンスRaQファミリーの新製品として「RaQ4シリーズ」を発売した。「RaQ4i」と「RaQ4r」の2モデルあり、RaQ4rはハードディスクを2台内蔵し、RAID 1（ミラーリング）に対応している。

CPUはK6-2 450MHz、メモリは128M～512Mバイト、15.2G～30Gバイトのハードディスク、10/100BASE-TXネットワークインターフェイスを2ポート備えている。サイズは432（W）×318（D）×45（H）mmで、1Uサイズサーバとしては奥行きが短めになっている、

OSにLinuxを採用し、Apacheを中心としたWebサーバ機能があらかじめ設定してあり、すぐにサービスを開始することができる。搭載されているChili! Soft ASP（Active Server Page）、PHP 4.0、InterBase 6.0を使うことで、ダイナミックWebコンテンツやデータベースを構築することが可能になっている。

Hardware 発売日 2000年8月23日

携帯電話から利用できるWebメールシステムをCobalt RaQ3に搭載 xGateアプライアンスサーバー

URL http://www.pc-net.nissho-ele.co.jp/

発売 日商エレクトロニクス株式会社
TEL 03-3544-8396
価格 52万5000円（20ユーザー）～

Cobalt Networksの国内代理店である日商エレクトロニクスは、携帯電話から利用できるWebメールシステム「xGate」をCobalt RaQ3に搭載した「xGateアプライアンスサーバー」を8月23日より発売した。価格は20ユーザーで52万5000円。

xGateは、EZweb／EZaccessなどのWAPブラウザに対応した携帯電話と、iモードに対応した携帯電話から、Webブラウザ機能を利用してメールをやりとりするソフトウェア。

xGateアプライアンスサーバーは、ファイアウォールの外側で常時インターネットに接続し、携帯電話からのWebアクセスがあった時点で、ファイアウォール内部にあるメールサーバとIMAP4プロトコルで通信してユーザーのメールにアクセスする。

メールを転送したりせずに、サーバ上で一元管理するので、PCと携帯電話で同じようにメールを見ることができる。IMAP4のサーチ機能を利用して、目的のメールだけを読むことが可能。Webブラウザでメールを読むため、受信文字数に制限がないといった利点がある。

Software 発売日 2000年7月25日

システム管理機能を強化したWebメールサーバソフトウェア GraceMail Ver.2.0

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 18万9000円（クライアント数制限なし）

日立マイクロソフトウェアシステムズは、PCからWebブラウザでメールの読み書きが行えるメールサーバソフトウェアの最新バージョン「GraceMail Ver.2.0」を、7月25日より発売した。対応サーバOSは、Linux、Solaris、FreeBSD、UP-UX、Sun OS。価格は、18万9000円（クライアント数制限なし）。

新たにIMAP4サーバに対応し、ユーザー認証方式を、UNIXアカウント統合／専用パスワードファイル利用／メール認証の3タイプから選択が可能。メール利用者の操作画面の改善と下書き機能の強化も行われた。

オプションで、利用者ごとにメールサーバのディスク容量を制限することや、メールを自動削除する機能、携帯電話でのメール確認機能が用意されている。

Software 発売日 2000年8月28日

インターネットゲートウェイサーバ用ウイルス対策ソフト InterScan VirusWall for Linux Ver.3.6

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

発売 トレンドマイクロ株式会社
TEL 03-5334-3650
価格 36万円（30ユーザー）～

トレンドマイクロは、インターネット・ゲートウェイウイルス対策製品「InterScan VirusWall for Linux Ver.3.6」（以下InterScan）を8月28日から発売した。新たにRed Hat Linux 6.2JとTurboLinux Server日本語版6.1に対応版した。価格は、1年間のサポートサービスを含んで、30ユーザーライセンスで36万円から。

InterScanは、インターネットに接続するサーバ上で動作し、各種コンピュータウイルスの媒体となる可能性のあるSMTP／HTTP／FTPプロトコルを監視し、ウイルスの侵入／流出の両方を防ぐ。

ウイルスを発見した場合の処理は、システム管理者への報告だけでなく、送信者と受信者にメールで警告を通知するなど、自由に設定できる。

毎時間自動的にウイルスパターンファイルがアップデートされ、最新のウイルスに対応する。

Hardware 発売日 2000年8月31日

TurboLinux Server 6.1とロータスノーツ／ドミノR5を添付 Netfinity 1000

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

発売	日本アイ・ビー・エム株式会社
TEL	0120-04-1992
価格	13万8000円

日本IBMは、タワー型PCサーバ「Netfinity 1000」の新モデルを8月31日より販売した。従来モデルからの変更箇所は、CPUをPentiumIII600MHzから650MHzに高速化した、TurboLinux Server日本語版6.1 OEM Versionとロータスノーツ／ドミノR5を添付した点である。24時間365日のオンサイト保証（1年間）が付いていて、価格は13万8000円と据え置かれた。

Netfinity 1000（8477-51J）は、CPUにPentiumIII 650MHz、メモリ64Mバイト（最大768Mバイト）、10.1GバイトIDEハードディスク（最大3台）、10/100BASE-TXネットワークインターフェイス、40倍速CD-ROMドライブ、AGP対応グラフィックスカード（4Mバイト）を装備している。

Hardware 発売日 2000年8月8日

2Uサイズの薄型ラックマウントサーバ Astrike ISシリーズ

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

発売	プロサイド株式会社
TEL	043-279-9280
価格	37万5000円～

プロサイドは、19インチラックマウント2Uサイズの薄型サーバ「Astrike IS」シリーズを、8月8日より発売した。

CPUはPentiumIIIを2個搭載可能で、メモリは最大2Gバイトまで拡張可能、前面からホットスワップ可能なハードディスクベイを4ドライブ分備えている。OSは、Windows NT／2000、Linux、FreeBSD、Solaris（7以降）に対応する。

価格は、PentiumIII700MHz、128MバイトECCメモリ、9GバイトSCAハードディスク、24倍速CD-ROM、フロッピードライブ、オンボードに10/100Base-TXネットワークインターフェイスとAdaptec AIC7890 SCSIコントローラを搭載した「Astrike IS2 G700」は37万5000円。

PentiumIII700MHzを2個、256MバイトECCメモリ、9GバイトSCAハードディスクを3個、Ultra2 Wide SCSI対応のRAIDコントローラを搭載した「Astrike IS2 G700-U22」は65万8000円となっている。

Software 発売日 2000年8月30日

Linux版エンタープライズ向けRDBMS Oracle8i Enterprise Edition for Linux R8.1.6

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

発売	日本オラクル株式会社
TEL	03-5645-7373
価格	160万円（8同時ユーザー）～

日本オラクルは、Linux版のリレーショナル・データベース管理システム（RDBMS）「Oracle8i Enterprise Edition for Linux R8.1.6」を、8月30日より発売した。価格は160万円（8同時ユーザー）から。

すでに4月より提供されている、Sun SPARC Solaris、HP-UX、IBM AIX、Compaq Tru64 UNIX、Windows NTの主要プラットフォーム用に対応する「Oracle8i Enterprise Edition R8.1.6」のLinux版で、日本オラクルが提供するLinux版RDBMSとしては、初めてのエンタープライズ向け製品である。

また、ワークグループ・レベル向け「Oracle8i Workgroup Server for Linux R8.1.6」を、9月中旬より出荷する。価格は22万円（5同時ユーザー）から。

対応ディストリビューションは、Red Hat Linux 6.0/6.1/6.1J/6.1J改/6.2/6.2J/6.2J改訂版、TurboLinux Server日本語版6.0/6.1。Miracle Linux Standard Edition V1.0への対応も予定している。

Software 発売日 2000年8月1日

オブジェクト指向アプリケーション開発ツール TIPPLER for Linux 2.0

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

発売	日本ユニシス株式会社
TEL	03-5546-6086
価格	30万円（開発用パッケージ）

日本ユニシスは、GUIアプリケーション開発支援ツール「TIPPLER」のLinux版の新バージョン「TIPPLER for Linux 2.0」を8月1日より発売した。

WebブラウザのFORMによるデータ入力や、Cookie情報の作成および受け渡しが容易に行えるCGI機能の実装、Webブラウザからデータベースへのデータ入力／参照が可能、データベース上の数値や計算結果を元に、Webブラウザ上に表／グラフ／図形描画を表示する、などの新機能が追加された。

これらのWebサーバアプリケーション開発機能を使用することで、C言語やJava、Perlよりも生産性が10倍以上アップするという。

TIPPLER for Linux 2.0は、Linuxカーネル2.2、glibc 2.1で稼働し、TurboLinux Server日本語版6.0とTurboLinux Workstation日本語版6.0に対応する。

価格は、開発用パッケージが30万円、デスクトップ版実行用ライセンスが5万円（1ユーザー）から、Webサーバアプリケーション版実行用ライセンスが5万円（10ユーザー）から100万円（無制限）。

News Headline from 日刊アスキーLinux

www.Linux24.com

Sybase、Watcom C/C++/Fortranコンパイラをオープンソース化

2000年8月23日

Sybaseは8月22日、Watcom C/C++およびWatcom Fortranコンパイラのソースコードをオープンソースでリリースすると発表した。ユーザーによるWatcomコンパイラの改良を可能にするのが狙いだ。

公式メンテナーとしてWatcomコンパイラのソースコードを管理するのは、Watcomコンパイラのヘビーユーザーという SciTech Software。同社は「Open Watcom」としてソースコードを公開し、http://www.openwatcom.org/の管理を行う。

ライセンスは、Open Source Initiativeが定めた「オープンソースの定義」（Open Source Definition）に適合するものになる予定。オープンソースの定義は、ただソースコードが入手できるだけではなく派生ソフトウェアの作成や再配布を認めることを要求しているため、ユーザーが自由に改良できるライセンスになるはずだ。しかしGPL互換でないライセンスであれば、GPLでライセンスされるGCCなどがWatcomコンパイラのソースコードを取り込むことは認められない。

今後のリリースの予定としては、まず現在の商用バージョン（11.0c）に対するバイナリパッチが、SciTechが選んだ外部開発者によって公開される。その後、ソースコードが公開されるとしている。

Sybase（http://www.sybase.com/）

Palm対抗のLinuxマシン「Agenda VR3」、149ドルで登場！

2000年8月19日

Agenda Computingは、Palmライクなポータブル PC「Agenda VR3」を発表した。スタイラスペンで操作するこのコンパクトなマシンのOSはLinuxだ。形状、操作方法ともにPalmに似ていて、160×240ドットの液晶を装備し、重量は110g。価格は149ドルから。

搭載したCPUはNECのMIPSプロセッサVR4181 66MHz。これはPalmが採用したDragonball EZ 20MHzなどと比較すればかなり高速な部類に入る。メモリは8Mバイトの RAMと2M～8Mバイトのフラッシュ ROMが搭載されている。

デスクトップPCとは「QuickSync」クレイドルに本体を載せてデータの連携を取ることが可能。PC側に必要なソフトウェア「QuickSync」はLinux版とWindows版が用意される。また、赤外線ポート経由でほかのAgenda VR3やPalmとデータを交換することもできる。

アルファベットの入力は、Palmと同じくスタイラスペンで行う。ただし、PalmのようにGraffiti（認識しやすいように一筆書きにしたアルファベット）を覚える必要はなく、ふつうに文字を書くだけで正しく認識されるという。

プレインストールのソフトウェアは、ToDoリストやスケジューラ、メモ帳、メーラ、FAXなど。メーラはMicrosoft Outlookと連携することができ、QuickSyncクレイドルに載せたタイミングでメールを送受信することも可能だ。

価格はフラッシュROMの容量によって分かれており、2Mバイトフラッシュ ROMモデルが149ドル、4Mバイトモデルが199ドル、8Mバイトモデルが249ドル。

Agenda Computingは上位機種「Agenda VR5」の出荷を2000年11月に予定している。このモデルは、リチウムイオン二次電池にメタルケースの外装を備え、価格は299ドルを予定。

Agenda Computing（http://www.agendacomputing.com/）

Linux、AMDの64ビットプロセッサ“Hammer”をサポートへ

2000年8月17日

AMDは8月15日、開発中の64ビットプロセッサアーキテクチャ「x86-64」のサポートを、Red HatとSuSEから得たと発表した。両社はそれぞれのLinuxディストリビューションや開発ツールをx86-64に移植する。x86-64の第1世代のプロセッサ（コードネーム“Hammer”）は、2001年末のリリースが予定されている。

x86-64は、AMDが独自にx86を64ビットに拡張したアーキテクチャだ。「IA-64」アーキテクチャをVLIW(Very Long Instruction Word）技術を利用して新たに開発するというIntelの野心的な計画とは異なり、AMDは現実的な路線を取って、x86をほとんどそのまま64ビットに拡張することを決定した。

x86-64は「64ビットモード」と「互換モード」の2つのモードがあり、64ビットモードに切り換えることで、4Gバイトを超える巨大なメモリ空間を扱うことができる。この機能は、とくに大きなデータを処理するデータベースなどに必要とされている。64ビットモードでは、増設された8つの汎用レジスタを利用することが可能だ。また、互換モードでは従来の32／16ビットアプリケーションを動作させることができる。

Linuxが新たなプロセッサに対応するには、コンパイラなどの開発ツールと、カーネルなどのハードウェアに密着したソフトウェアの対応が必要だ。また、64ビットで正しくコンパイルできないアプリケーションは書き直さなければならない。

GCCなどを開発するCodeSourceryは、開発ツールをx86-64に対応させる計画を発表している。

AMDは2000年9月、x86-64アーキテクチャのエミュレータをリリースする。このエミュレータは、http://www.x86-64.org/で公開される予定だ。

まだプロセッサも市場に出ていないにもかかわらず、Linuxはすでに IA-64へ対応している。これはIntelなどの努力によるものだ。AMDのx86-64アーキテクチャもこれと同じ状況を目指すとみられる。

AMD（http://www.amd.com/）

Yahoo！ MessengerもLinux／FreeBSDに対応

2000年8月17日

AOL Instant Messenger（AIM）のLinux対応版に引き続き、Yahoo！も「Yahoo！ Messenger for Unix」をリリースした。対応プラットフォームはIntel版のRed Hat Linux 6.0/6.1/6.2と、FreeBSD 2.2/3.4/4.1。動作にはGTK 1.2.3以上が必要だ。

これによりLinuxユーザーは、AIMとYahoo！ Messengerという2大インスタントメッセージングの正式なクライアントを手にしたことになる。残る主な競合サービスはMSN MessengerとICQだが、これらには非公式なクローンが在存するものの、Linuxに対応した正式バージョンはまだ公開されていない。

Yahoo！ Messengerはいままで、AIMと同じように、Linuxに対応した非公式なクローンがリリースされていた。そのうちのひとつ、「GTKYahoo」は、Windows版と同様にチャットが可能なうえ、メッセージの自動返答やメールによるフォワードなど正式版にない機能さえ併せ持つ高機能クローンだ。ソースコードはGPLに従い配布されているので、ユーザーは、これらのクローンによってiモードとYahoo！ Messengerのゲートウェイさえ構築可能になる。

そのほかのクローンとしては、KDEのクライアント「KYahoo」や、Yahoo！ Messenger、ICQ、AIMの機能をオールインワンにした「EveryBuddy」などが存在する。

「Yahoo！Messenger for Unix」のダウンロード
（http://messenger.yahoo.com/messenger/download/unix.html）

GNOMEの普及を目指す新団体「GNOME Foundation」を設立

2000年8月16日

Sun MicrosystemsやCompaqなどの大手コンピュータ会社は、Linuxディストリビュータである Red Hat、TurboLinuxなどと共同でフリーのGUI環境である「GNOME（GNU Network Object Model Environment）」の普及を目指す新団体「GNOME Foundation」を設立すると発表した。

「GNOME」はユーザーに優しいデスクトップ環境を目指して開発が進められているフリーソフトウェア。アプリケーションに統一されたルック＆フィールを持たせるとともに、その連携方法まで規定しているのが特徴である。

今回設立された「GNOME Foundation」は、開発元であるGNOMEプロジェクトに対して、財政的、法的な支援を行い、またGNOMEの仕様やロードマップを決定するための手助けを行なっていく予定である。なお、「GNOME Foundation」は「Apache Foundation」をモデルにして組織され、役員会には数百人のGNOME開発者から選ばれた人たちが含まれる。

また、GNOMEプロジェクトの開発しているオフィススイート「GNOME Office」に、先日オープンソース化を発表したSun Microsystemsの「StarOffice」が統合されることが明らかになった。今回の「GNOME Foundation」設立によって、Sun Microsystemsでは、GNOMEがより安定した環境をユーザーに供給することを期待している。

さらに、オープンソースで現在開発の進んでいるブラウザ「Mozilla」が、GNOMEに統合されることになった。「Mozilla」は小サイズ化、高速化した新レンダリングエンジン「Gecko」を使用しているブラウザで、現在テスト版のM17がリリースされている。

なお、「GNOME Foundation」のサポートを表明している主な企業は、以下のとおり。

- Compaq
- Eazel
- Helix Code
- Hewlett-Packard
- IBM
- Red Hat
- Sun Microsystems
- Turbo Linux
- VA Linux
- Free Software Foundation(FSF)

今後、SolarisやHP-UX上で、GNOMEが標準のデスクトップ環境になる日が来るかもしれない。

プレスリリース
（http://www.gnome.org/pr-foundation.html）

IBM Research、腕時計サイズ「スマート・ウォッチ」でLinuxとX11R6を動作

2000年8月8日

IBMは、S/390などのハイエンドサーバからスーパーコンピュータ、そして腕時計サイズのデバイスまでLinuxを動作させようとしている。

IBM Researchは8月7日、腕時計「スマート・ウォッチ」でLinuxとX Window Systemを動作させることに成功したと発表した。このスマート・ウォッチは、PCや携帯電話などとのワイアレス通信に対応しており、メールなどの受信が可能だという。また、カレンダーやアドレス帳、ToDoリストなどの機能も備えている。操作は、タッチスクリーンとホイールを組み合わせて行う。

スマート・ウォッチには、8Mバイトのフラッシュメモリと8MバイトのDRAMが搭載されている。重さは44グラム、サイズは56mm（W）×48mm（D）×12.25mm（H）だ。日本IBM東京基礎研究所がハードウェアのアーキテクチャとシステム設計を行い、日本IBM野洲事業所が基板を製造した。

組み込み用途におけるLinuxの人気は最近急激に高まってきた。TCP/IPのサポートが簡単であることや、ライセンス料を払わなくてよいということがその理由だ。Ericssonは今年末、組み込みRed Hat Linuxを利用したコードレススクリーン電話の発売を予定している。

IBM Research
（http://www.research.ibm.com/）

Linux用の戦国シミュレーションゲームが今秋に登場

2000年8月8日

2000年7月に設立されたばかりのベンチャー企業、キューブ・ソフトウェアが、戦国シミュレーションゲーム「戦国 for Linux」を秋頃に発売する。1560年以

降の日本の戦国時代を舞台に、プレイヤーが当主を操作することによって全国統一を目指すゲームだ。価格は未定。

キューブ・ソフトウェア
(http://www.cube-software.co.jp/)

カナダCorel、画像作成／加工ソフトウェア「Corel PHOTO-PAINT 9 for Linux」を公開

2000年8月6日

カナダのCorelは、フォトレタッチや画像作成が可能なソフトウェア「Corel PHOTO-PAINT 9 for Linux」（以下PHOTO-PAINT 9）を公開した。アーカイブは、RPM系用（約90Mバイト）とDebian GNU/Linux系用（約90Mバイト）、そしてこの2つをまとめたもの（約180Mバイト）が用意されている。

PHOTO-PAINT 9は、画像作成もさることながら、Photoshop並みのレタッチ機能が強力なソフトウェアだ。WINE（WINdows Emulator： UNIX系OSにWindows互換のAPIを実装することにより、X Window System上における Windowsアプリケーションの実行を可能にするソフトウェア）を使用するアプリケーションのため、ユーザーインターフェイスもWindowsユーザーには慣れ親しんだものになっている。Photoshopの画像フォーマットであるpsd形式での保存も可能で、その際は画像のレイヤー情報などもきちんと残される。

Corel PHOTO-PAINT 9 for Linuxダウンロードページ
(http://linux.corel.com/products/pp9/download.htm)

Caldera Systems、SCOのサーバソフトウェア部門などを買収

2000年8月3日

LinuxディトリビュータのCaldera Systemsと“UnixWare”を取有するSanta Cruz Operation（SCO）は8月2日、Caldera SystemsがSCOのサーバソフトウェア部門およびプロフェッショナルサービス部門を買収することに合意したと発表した。この契約によりCaldera Systemsは新たな子会社として“Caldera”を設立し、SCOはCalderaの株の28%と700万ドルの現金を受け取る。

新会社のCalderaは、SCOのOpenServer製品ラインの排他的販売権を手にし、現在のSCO OpenServerカスタマーに対してサービスを行う。また、UNIXとLinuxのサーバ／サービスを組み合わせた「Open Internet Platform」を、クライアントから大規模データセンタにまで提供するとしている。

Calderaの社長には、Caldera Systemsの現CEO Ransom Love氏が就任する。SCOからは、社長兼COOとしてプロフェッショナルサービス部門の社長Jim Wilt氏が、SCOのCEO Doug Michels氏が取締役に就任する。

Caldera Systems社長Ransom Love氏はリリースの中で、「Calderaは、SCO OpenServerおよびUnixWareコミュニティに対してサポート／サービスを行なっていく」と語っている。

Caldera Systems
(http://www.calderasystems.com/)
SCO（http://www.sco.com/）

日本IBM、1台のS/390で数万規模のLinuxシステムを同時に稼動可能に

2000年8月3日

日本IBMは8月2日、1台の「S/390」エンタープライズサーバで複数のLinuxシステムを稼動可能にするソフトウェア「バーチャル・イメージ・ファシリティーfor LINUX」を発表した。料金は264万円からで、2000年10月の出荷が予定されている。

S/390のハードウェアは、複数OSのネイティブな同時稼動を可能にする論理区画（バーチャルマシン）機能を持っているが、Linuxの稼動数は15までに限られていた。これに対して「バーチャル・イメージ・ファシリティーfor LINUX」では、さらに多くのLinuxを1台のS/390で動かすことが可能になる。日本IBMは「数万規模のLinuxサーバの同時稼動を実現できる」としている。そのほかの利点として、各Linuxシステムの利用しているプロセッサ・メモリ・ディスク資源などを一元管理できることが挙げられるという。

日本IBMは同ソフトウェアの適用業務として、Windows NTやUNIXで稼動していたWebアプリケーションの統合や、ISP／ASPにおけるLinuxアプリケーションの大規模利用、大学・研究機関での学術利用を見込んでいる。

また、すでにS/390を利用しているユーザーのために、新しいLinux専用プロセッサ「LINUX処理機構」が発表された。ユーザーはこのプロセッサを増設することによって、Linuxの処理速度を最大9倍まで高めることができるという。プロセッサの価格は月額53万7000円からで、2000年10月の出荷が予定されている。

日本IBM（http://www.ibm.co.jp/）

Inprise、RDBMS「InterBase 6」のソースコードを公開

2000年7月31日

Inpriseは7月25日、同社のRDBMS「InterBase 6」のソースコードを公開した。またバイナリについてもLinux／Windows／Solaris版が公開されている。

ライセンスには、MPL（Mozilla Public License）V1.1を部分的に変更した「InterBase Public License」を採用した。MPLは、NetscapeがNetscape Communicatorのソースコードを公開した際に、GPLやBSDライセンスといった既存のライセンスよりビジネス利用が容易になるよう意図して作成されたものだ。「InterBase Public License」では改変したコードを非公開にすることが許されているため、改良版をソース非公開の商用ソフトウェアとして販売することが可能だ。同ライセンスは、InterBaseのすべてのプラットフォームのソースコードに対して適用される。

オープンソースなRDBMSとしては、今回発表されたInterBaseのほかに、BSDライセンスの「PostgreSQL」、そして2000年6月にGPLへライセンスを変更した「MySQL」がある。

InterBaseのWebページ
(http://www.inprise.com/interbase/)

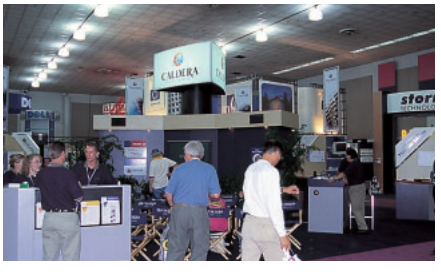
LinuxWorld Conference & EXPO 2000 San Jose レポート

文・写真：宮原 徹
Text/Photo：Toru Miyahara

8月14日より17日までシリコンバレーの中心地San JoseにてLinuxWorldが開催されました。展示会場の様子を中心に、アメリカでのLinux最新事情をレポートします。

▼基調講演を行うデル・コンピュータの会長兼CEOであるMichael Dell氏。

GNOME Foundation設立の発表が多くの参加企業によって行われた。



基調講演

今回で4回目となるLinuxWorldは毎回規模を拡大し、今回は展示が展示会場に入りきれずに廊下にまで溢れ出し、基調講演は会場の向かいの市民ホールで行われるなど過去最大の規模となりました。すでに来年2回の開催が決まっており、Linux・オープンソースに関する一大イベントに成長したようです。

これまでは基調講演にLinuxの開発者であるLinus Torvalds氏が登壇していたが、今回は同氏の基調講演は行われなかった。さすがに過去3回ともカーネル2.4の話題を中心に講演を行ってきたが、2.4がまだリリースされていないので特別話をすることもなかったのでしょう。

今回基調講演を行ったのはデル・コンピュータの会長兼CEOであるMichael Dell氏。講演では同社のサーバビジネスの成長や、自社でのLinuxシステム導入事例、トヨタでのメディアサーバとしての導入事例などが紹介された。しかし、聴衆からの「Linuxに対してどのような貢献をしているのか」という質問に対して明確な解答がないなど、若干迫力を欠いた感は否めない。ぜひとも今後に期待したいと思う。

Debian

本イベントでの最大のニュースは

Debian GNU/Linux 2.2のコードネーム「potato」にちなんで配布されたMr. PotatoHead。

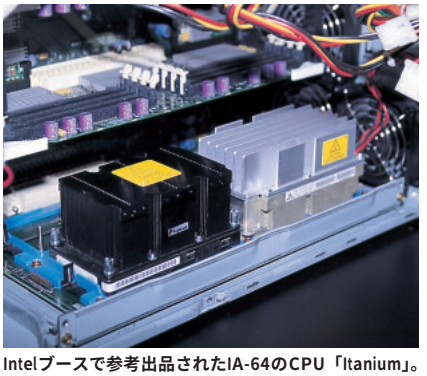
オープンソースコミュニティでお馴染みの人達がGeeksチームとNerdsチーム（どちらもオタクの意味）に分かれてオタククイズ合戦。

CNNのキャスターをモデレーターにLinuxビジネスでの著名ベンダーのトップによるパネル討論。今後のLinuxビジネスについて熱い討論が交わされた。


Slashdotのブースに山積みにされたDebianのパッケージ。朝には山積みにされていたものが午後にはすっかりなくなってしまっていた。

X Window Systemをデュアルモニターに表示し、リアルタイム3Dアニメーションをデモンストレーション。グラフィックス・オペレーションなどに。


IBMブースで参考出展された腕時計型のLinuxマシン。X Window Systemが動き、時計も表示できる（笑）これが実用化されれば、いつでもどこでもカーネルハックできるようになる？


Intelブースで参考出品されたIA-64のCPU「Itanium」。とにかく大きい。黒い部分はCPUファン、銀色の部分は電源冷却用の放熱フィン。


TurboLinuxブースに参考出品されていたNEC製Itanium 16 CPU搭載マシン「AzusA」。動作しているLinuxはカーネル2.4をベースに、16CPUまで動作させることによって発見されたバグを修正したパッチを当てているとのこと。ここまで来ると、もうPCとは呼べない。

Debian GNU/Linux 2.2（コードネーム「potato」）のリリースであろう。Debianは最も活躍しているコミュニティを表彰する「IDG/Linus Torvalds Community Award」にも選ばれている。また会場ではパッケージの箱が山積みにされて配布されるなど、他のディストリビューションに較べて活発な活動が見られた。

GNOME Foundation設立

もうひとつ大きなニュースとして、X Window Systemのデスクトップ環境「GNOME」の普及促進を図るためにGNOME Foundationが設立されたことであろう。Sun、HP、IBMといった商用UNIXベンダーも参画し、それぞれの商用UNIXの標準デスクトップとしてGNOMEを採用することを表明した。

特に積極的なSunは、先日GPL化を発表した「StarOffice※1」を「Bonobo※2」に対応したGNOMEの標準オブジェクトとして統合していくという。言ってみればWindowsに最初からMS Officeが組み込まれているようなものであり、今後のThinクライアントでの利用を睨んだ戦略と考えられる。

今後のより具体的な活動をどのようにしてゆくのか、これまでGNOMEの開発を担ってきたGNOME Projectとの関係についてなど、まだはっきりしていない部分も多く、今後の動きに注目したい。

組み込みLinux

今回の展示会場で最も目立ったのは小型機器へのLinux組み込みであろう。IBMの腕時計サイズのLinuxマシンやCOMPAQの Windows CE機 「iPaq」へのLinux組み込みといったように、大手ベンダーも注目をしている。また、LynuxWorks社の「BlueCat Linux」やMontaVista Software社の「Hard Hat Linux」といった新規参入の組み込み用パッケージも現れ、活況を呈していた。

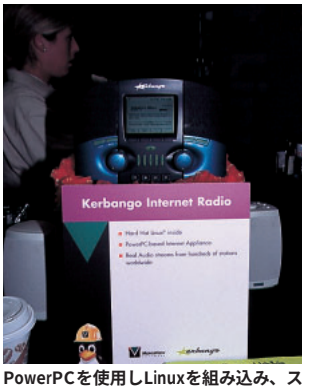
※1 日本国内ではSunOffice
※2 次期バージョンのGNOMEで採用されるCORBAベースの分散オブジェクトモデル

BSDブースで発見した、FreeBSD組み込みのドキュメントファイルサーバ。ネットワークに接続するだけでファイルサーバとして機能する。リコー・アメリカ製。

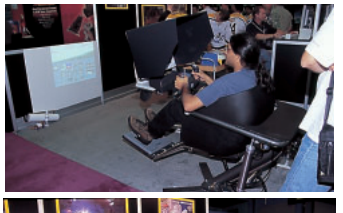
PFUアメリカブースで発見した完全防水型キーボード。黄色いゴムでキー・スイッチが裏側まで完全に覆われていて、巻いて持ち運びも可能。そういえばこれのスケルトンタイプが秋葉原で売られていたような？？

狭いブースに人が溢れかえっているのはTransvirtual社のPocketLinuxの展示。PDAへの組み込みLinuxにJava VMを載せて、さまざまなアプリケーションを動かそうというもの。まだ開発者キットの提供が始まった段階だが、各種PDAに対応していくという点から、完成度が高まれば人気が出そうである。

Inpriseのブースでは、DelphiのLinux版のプレビュー版をデモンストレーション。さすがに開発ツールとしての完成度は高く、1日も早い製品版のリリースが期待される。



PowerPCを使用しLinuxを組み込み、ストリーミング情報を再生するInternet Radio。Hard Hat LinuxのMonta Vista社ブースにて。



フライトシミュレーション用のコックピットを置いてデモンストレーションしていたのは、さまざまなゲームをLinuxに移植しているLoki社のブース。単純にコントローラが組み込まれているのではなく、操作に合わせて圧搾空気を使って筐体が動き回る優れモノ。残念ながら商品ではありませんでしたが。

Distribution

新着ディストリビューション

Kondara MNU/Linux 2000

使いやすい日本語デスクトップ環境の裏側に隠された先進性。Wnn6など日本語デスクトップ環境をさらに強化する商用ソフトをバンドルしてリリースされたKondara MNU/Linux 2000は、ジャーナリング機能を備えたファイルシステムReiserFSなどをサポートし用途やユーザーを選ばない非常に完成度の高い出来栄えとなっている。

マイスターLinux Mandrake 7.1

欧米で人気が高いというフランス産ディストリビューションのLinux Mandrake 7.1。洒落たデスクトップデザインに似合わず、Mandrakeのシステムは高速なバイナリを生成するコンパイラPGCCでビルドされている。そのMadrakeに、日本の老舗ディストリビュータ五橋研究所が日本語強化パッケージなどを付属したマイスター版をリリースした。

Red Hat Linux 6.2J改訂版

オープンソースプロジェクトとの融合。Red Hat Linux 6.2J改訂版は、レッドハット社とProject Vineの融合による初めての成果だ。Vine LinuxからRed Hat Linux 6.2J改訂版に収録されたアップデートパッケージは、日本語ディストリビューション最大勢力の誕生を予見させる内容だ。

Kondara MNU/Linux 2000

Kondara MNU/Linuxは日本語環境としての使いやすさと、ほかのディストリビューションの一歩先をいく豊富な先進機能で定評のあるディストリビューションだ。Kondara Projectが開発しており、現在最新のバージョン1.2（以下、Kondara 1.2）がFTPサイトで公開されている。

一方、9月8日に発売されるKondara MNU/Linux 2000（以下、Kondara 2000）は、デジタルファクトリがKondara 1.2をベースに独自拡張し、商用ソフトウェアを付属して販売する製品で、カーネルにバージョン2.2.16、ライブラリにglibc2.1.3、XにXFree86 3.3.6を採用している。Kondara 1.1までは商用ソフトが付属しなかったが、今回より他のディストリビューションと同様の形式になった。価格は税別1万2800円で、製品版購入ユーザーは、電話、FAX、電子メールによるサポートを、件数無制限で90日間受けられる。

Kondara 2000の新機能

現在ほぼすべてのLinuxディストリビューションでext2というファイルシステムが採用されている。このext2は、停電などで不正にシャットダウンした際に、シャットダウンの直前に変更されたデータが消えたり、大容量ハードディスクの環境下では、再起動時に走るfsckというプログラムのため、Linuxの起動に数十分以上を要する場合がある。しかし、Kondara 2000で新しくサポートされたReiserFSは、データ更新に関する情報を別に確保しているため、ext2で見られる膨大な起動時間からユーザーを解放してくれる。

このほかにも、本格的なGUI操作がウリのメーラSylpheedや、個人データを守るGnuPGといった暗号化ソフトも新しく収録され、Intelの新しいチップセットi815もサポート対象になっている。

Kondara 1.2との違いは？

さて、Kondara 1.1からのバージョンアップ内容もさることながら、現在FTPサイトで公開されているKondara 1.2と、Kodnara 2000の違いがどこにあるのか気になる読者もいるだろう。Kondara 2000は、Kondara 1.2のカーネルを拡張したことにより、i815チップセットやATA66に対応して、商用アプリケーションがバンドルされているのが特徴だ（詳細は**表1**、**表2**を参照）。ひらたく言えば、Kondara 1.2+αに、商用ソフトとユーザーサポートを加えたものといったところだ。ともに最新バージョンであるKondara 2000とKondara 1.2、ユーザーは上記の違いを踏まえて選択すると良いだろう。

本誌付録CD-ROM収録のKondara MNU/Linux 2000はFTP版です。非商用ソフトだけが含まれており、製品版を販売しているデジタルファクトリからサポートを受けることはできません。

	Kondara 2000	Kondara 1.2	Kondara 1.1
カーネル	2.2.16	2.2.16	2.2.14
GNOME	1.2	1.1.90	1.1.4
OpenLDAP	○	○	×
Sylpheed	○	○	×
Mozilla	○	○	×
GnuPG関連ツール	○	○	×
JDK 1.2.2	×	○	○
i815サポート	○	×	×
ATA66サポート	○	×	×
ReiserFSサポート	○	○	×
パッケージ販売	○	×	○

表1　3つのバージョン間の違い
Kondara 1.1とKondara 2000/1.2は主に収録パッケージが異なり、Kondara 2000とKondara 1.2の違いは、主にサポートされるデバイス、商用ソフトの有無である。Kondara 1.2の製品版がないことに注意しよう。

商用アプリケーション	概要
DynaFont 5書体	美しく日本語を表示する商用フォント
Wnn6 ver.3	UNIX系OS上で定番の日本語入力プログラム
翻訳魂	英和・和英の翻訳ソフト
ThinkFree Office	Microsoft Office互換のオフィスアプリケーション
System Commander Lite	マルチブートプログラムの簡易版

表2　Kondara 2000にバンドルされる商用アプリケーション
Kondara 2000には上記の商用アプリケーションが収録される。このほかにも体験版ソフトが収録される。

製品名　Kondara MNU/Linux 2000
価格　1万2800円（アカデミック版は8900円）
問い合わせ先　デジタルファクトリ株式会社
06-6882-5850
http://www.digitalfactory.co.jp/

マイスターLinux Mandrake 7.1

マイスターLinux Mandrake 7.1（以下、マイスターMandrake 7.1）は、フランスのMandrakeSoft SAが開発するLinux Mandrake 7.1（以下、Mandrake 7.1）を、日本の老舗Linuxディストリビュータである五橋研究所が、日本語環境を強化して9月1日に発売したディストリビューションである。

Mandrake 7.1は、日本での認知度はさほど高くないものの、ほぼ全自動化されたインストーラ、大容量ハードディスク環境のもとで威力を発揮する、ジャーナリング機能を備えたファイルシステムReiserFS、インストール時に選択できるXFree86の最新バージョン4.0などをサポートしており、欧米では人気の高いディストリビューションのひとつである。

画面1　日本語対応になったNetscape
Meister CD-ROMに付属するスクリプトを使って、Netscapeを日本語対応にした様子。画面左側は標準のもの、画面右側がMeister CD-ROMに付属するものである。

マイスターってなに？

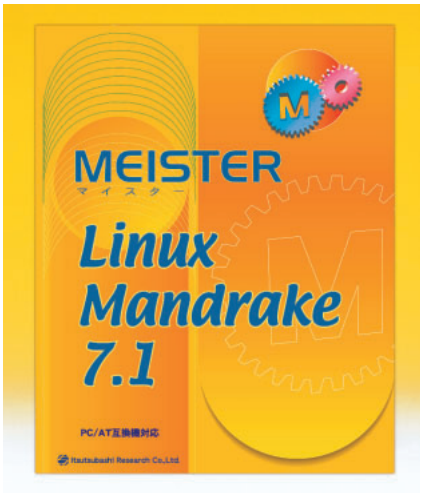
マイスターMandrake 7.1は、FTPサイトで公開されているMandrake 7.1用のインストールCD-ROMに、五橋研究所が日本語環境強化のために用意したMeister CD-ROM、ユーザーからの収録要望が多いと思われる追加パッケージを含むExtensions CD-ROM、Linux初心者を考慮して作られたインストールマニュアルを加えて販売する製品だ（詳細は**表1**を参照）。DynaFont以外はフリーソフトで構成して、ユーザーサポートも行わないことで、3000円台という低価格を実現している。

さっそく五橋研究所が提供するマイスタースクリプトを実行したところ、KDEのメニューが英語表示から日本語表示に変わったほか、Netscapeもしっかりと日本語対応となった（**画面1**）。このほかにも、FTP版にあったキーボード配列の不具合が解消され、X上でもコンソール上でも問題なく日本語キーボードが使えるようになっている。

ほかよりも一歩進んだ日本語デスクトップ環境として、もう少しMandrake 7.1を使い込んでみたい、そんなユーザーにとって、安価ながら必要なツボが押さえられたマイスターMandrake 7.1は、お買い得感の強い製品と言えるだろう。

マイスターのみの収録物	概要
Meister CD-ROM	Mandrake 7.1をより良く使うためのパッケージを収録 ・日本語化されたWebブラウザのNetscape ・商用日本語フォントのDynaFont 5書体 ・日本語キーボードを利用可能にするスクリプト ・KDEのメニューを日本語表示するための設定ファイル ・バグフィックスなどで更新されたRPMパッケージ
Extensions CD-ROM	ユーザーから収録要望が多いと思われる追加パッケージを収録 ・高速なオープンソースRDBMSのMySQL ・オブジェクト指向のスクリプト言語Ruby ・Linuxでネットワーク越しにWindowsを操作するVNC
日本語インストールマニュアル	初心者にもわかりやすいインストールガイド

表1　マイスターMandrake 7.1のみの収録物
FTP版のMandrake 7.1と、マイスターMandrake 7.1の違いは、表のように追加収録された2枚のCD-ROMと、インストールマニュアルの有無だ。



製品名　マイスター Linux Mandrake 7.1
価格　3480円（税別）
問い合わせ先　株式会社 五橋研究所
03-5818-6608
http://www.cdrom.co.jp/

Red Hat Linux 6.2J改訂版

先頃リリースされたRed Hat Linux 6.2J改訂版（以下、Red Hat 6.2J改）は、4月に発売されたRed Hat Linux 6.2J（以下Red Hat 6.2J）にアップデートパッケージなどを追加収録したものだ。Red Hat 6.2JとRed Hat 6.2J改の具体的な違いは、Project Vineからのアップデートパッケージなどを収録したCD-ROMの有無で（詳細は**表1**を参照）、インストール用のCD-ROMは、Red Hat 6.2JとRed Hat 6.2J改で共通である。

Project Vineとの協力体制

去る7月4日、Project VineからVine Linuxの開発、製品版の販売、サポートに関して、レッドハット株式会社と協力して行うというアナウンスが流れた（40ページからのインタビュー記事を参照）。今回Red Hat 6.2J改に収録されたパッケージは、Project Vineとレッドハット社が敷く協力体制の第一弾となるもので、主に、

- **GNOME 1.2.3**
- **Emacs関連パッケージ**
- **Netscape**
- **XFree86 3.3.6**

などがVine Linuxから収録されている。これら追加パッケージはRed Hat Linux用の調整されており、rpmコマンドか、レッドハット社がWebサイトで配布しているインストールスクリプトを使ってインストールする。この中からEmacsに関するアップデートパッケージを検証してみた。

Red Hat 6.2Jに収録されているEmacsには、Cannaなどによる日本語入力ができないという不具合がある。この不具合を解消するために、Project Vineから3つのアップデートパッケージが寄贈された。実際にこれらをインストールしたところ、Emacs上でCannaを使って日本語入力することができた（**画面1**）。

しかし、Emacs関連のパッケージをはじめ、この他の追加パッケージの使用については、レッドハット社からサポートを受けることができないため、安定動作を望むRed Hat Linuxユーザーは、次期バージョンで、Vine Linuxのパッケージが不具合なく取り込まれるのを待つのが懸命だろう。

次期バージョンに期待

Vine Linux Technologyのほかにも、韓国製日本語ワープロのハンコムワード、オープンソースのRDBMSとして代表的なPostgreSQLの最新版のバージョン7.0.2がバンドルされ、商用RDBMSのOracle 8iの公式プラットホームに指定されていることを受けて、Oracle 8iの導入ガイドが含まれている。しかし、Red Hat 6.2J改で追加収録されたパッケージは、導入作業もそれほど簡単ではないことから、エンドユーザーが安全にその恩恵を受けるのはやや難しい。とはいえ、レッドハット社とProject Vineの協力体制は始まったばかりなので、今後日本語環境の完成度が高いVine Linuxと、本家Red Hat Linuxが融合され、どんな日本語ディストリビューションがリリースされるのか今から楽しみである。

画面1　アップデートパッケージをインストール後のEmacs
Project VineからコントリビュートされたEmacs関連のアップデートパッケージにより、Cannaを使った日本語入力が可能になった。

Project Vineによるアップデートパッケージ	Project Vineとの協力によるRed Hat用追加パッケージ
レッドハット社によるアップデートパッケージ	バグフィックスなどで更新されたパッケージ
PostgreSQL 7.0.2	定評のあるオープンソースRDBMSの最新版
Oracle8iのインストールガイド	インストールが難しいOracle8iの導入ガイド
ハンコムワード	日本語文書を作成できる韓国製ワードプロセッサ

表1　Red Hat Linux 6.2Jに追加された収録物
Red Hat Linuxの6.2Jと6.2J改の違いは、これら追加パッケージの有無である。

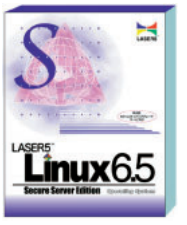
製品名　Red Hat Linux 6.2J改訂版
価格　1万2800円（デラックス）
問い合わせ先　レッドハット株式会社
03-3257-0411
http://www.redhat.com/jp/

Distribution ▶▶▶

LASER5 Linux 6.5 Secure Server Editionリリース

レーザーファイブは、LASER5 Linux Server Editionの新製品「LASER5 Linux 6.5 Secure Server Edition」（以下LASER5 6.5 SSE）を、9月1日から発売した。価格は4万9800円。1年間4回のセキュリティアップデートサービスが付属する。

LASER5 6.5 SSEは、Red Hat Linuxをベースにしたサーバ用ディストリビューションで、各種サーバ機能のデフォルトのパラメータを、より安全な値に設定するなどセキュリティを強化している。ファイアウォールを設置していない環境でのセキュリティを高めるため、パケットフィルタ型ファイアウォール機能を備えている。また、セキュリティやリソースの異常を音声によって知らせる機能を搭載している。



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

Kondara MNU/Linux Web Cluster 2000リリース

デジタルファクトリは、クラスタリングによるWebサーバシステムを容易に構築できる「Kondara MNU/Linux Web Cluster 2000」（以下Web Cluster）を、9月14日から発売する。価格は、2ノード版が12万8000円、ノード数無制限版が24万8000円。CPUにPentiumまたはAlphaを使用したシステムに対応する。

Web Clusterをインストールした複数のマシンで構成したクラスタシステムでは、Webサーバへのアクセスは、各ノードに振り分けられ負荷を分散することができる。障害があったノードを切り離し、残ったノードでシステムを自動的に再構成する機能と、ノード間でファイルリストを交換して、コンテンツの同期が自動的に行われる機能によって、ノードの追加／削除が自由に行える。

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

Oracleデータベースに対応したMiracle Linux、9月末リリース

ミラクル・リナックスは、Oracleデータベースに最適化したディストリビューション「Miracle Linux」をリリースする。9月末日より、中小規模の部門レベルを対象とした「Miracle Linux Standard Edition V1.0」を出荷する。

価格は1サーバあたり5万円。販売は、日本オラクルを経由して各販売パートナーから行われ、サポートは、日本オラクルのサポートセンターを拠点にミラクル・リナックスのサポートスタッフが行う。

Miracle Linuxは、「Oracle8i for Linux Release 8.1.6」に対応し、Oracleデータベースのインストールを支援するためのツールが付属する。

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

Linux 2000Gの登録ユーザー全員に「スペシャルボックス」を提供

ホロンは、Linux 2000Gのユーザー登録者全員に、追加機能を満載した「スペシャルボックス」を、8月18日より発送開始した。

スペシャルボックスには、日本語対応メールソフト「sylpheed」、携帯情報端末PalmとLinuxの連携ソフト「jpilot」、TrueTypeフォントのインストールと自動設定を行う「ttftool」などのアプリケーションが収録されている。

また、最新カーネルやドライバが追加されており、iMac／iBook／G3／G4／PowerBook G3に対応、IntelマシンでUSBキーボード・マウスに対応、プリンタ対応機種の充実などの新機能がある。

株式会社ホロン（http://www.linux2000g.ne.jp/）

Storm Linux 2000 リリース

Stormix Technologies社は、Debian GNU/Linux 2.2（potato）をベースとしたStorm Linux 2000をリリースした。パッケージは2種類あり、「Storm Linux 2000 Starter Edition」は、PartitionMagic Linux Prep Tool、StarOffice 5.2などをバンドルしている。

「Storm Linux 2000 Deluxe Edition」は、Starter Editionに加え、ゲームソフト「Heros of Might and Magic III」、「Flash Player 4.0」、「Acrobat Reader 4.03」などと、StarNet Communications社のWindows用Xサーバ「X-Win32」の1年間ライセンスをバンドルしている。価格は、Starter Editionが19.95USドル、Deluxe Editionが69.95USドル。

なお、Storm Linux 2000 Deluxe Edition日本語版の発売が今年秋に予定されている。

Stormix Technologies Inc.（http://www.stormix.com/）

Vine Linuxよ、いずこへ?

さる7月4日、Vine Linuxの開発を行っているProject Vineが、今後のVine Linuxの販売元を技術評論社からレッドハットに変更するとのアナウンスを行った。また、レッドハットとの協力体制を強化し、本格的に企業ユーザーをサポートしていくとのことだ。
レッドハットとの提携のいきさつと今後の展望をProject Vine代表の鈴木大輔氏と、レッドハット・プロダクトマーケティングマネージャの染谷邦裕氏に聞いた。

――協力関係を結ぶことになった理由は?

鈴木: 会社や学校で大量にVine Linuxが導入され、質問が一度にたくさん来るようになって、技術評論社では手に負えなくなってきた。技術評論社には専任担当者を置いてもらっていたが、サポートが専門ではないという事情がある。

染谷: VineはRed Hat Linuxをベースにしていることもあって、サポートするには共通点が多い。

Vineとは得意な部分とそうでないところで、かみ合う点が大きかった。Red HatはOSのカーネルやglibcを中心に開発しているし、Vineはそれをさらに使いやすくしている。プログラマーから見れば、軽くて小さくてちゃんと動くソフトほど優秀なソフトで、Vineはそういった理念を持ってやっている。逆にいうとレッドハットは、そういうのには手が回らない部分が多い。Vineの開発にはハッカーの人がたくさん参加していて、日本のリソースを食いつぶすのはもったいない。なるべく二度手間にならないようにしたいと思っている。

ディストリビューションとして自分たちの名前を出しても独占していくわけではなく、みんなオープンソースでやっていくわけだから、密な情報交換があったほうがよいだろう。

――Vine LinuxとRed Hat Linuxがより近くなったと思われるが、どこに差を出していくのか?

染谷: レッドハットには、やりたくてもできなかったことがいくつかある。Red Hat Linuxの前に「Official」がついているのは、gnome.orgなどの、いろいろなorgからオフィシャルに出てきたものを使っているためである。日本の中では標準になっていても、世界でオーソライズされていないものは入れることができない。

染谷 邦裕氏
レッドハット株式会社プロダクトマーケティングマネージャ
日本PostgreSQLユーザー会広報担当理事

日本語環境とか、これから取り込んでいかないといけないものがいっぱいある。Vineと一緒にやっていくことで、早くグローバライズしていくという目標があったので、思いきって社内でオーソライズして、Vine Linux Technologyを入れた。

Red Hatは、どうしてもOfficialとして、重厚長大なディストリビューションになってきてしまっているため、「何でもできるが、何にもできない」というようになっている。Vineには、しっかりした操作環境があり、そこに差がある。

――Vine Linuxは1枚目のCD-ROMでインストールした時点で、使いやすい環境ができているのが日本のユーザーに評価されているのでは?

鈴木: Vineは、どちらかというと押しつけているところがある。ハードディスクに再インストールし直しても、自分の環境と同じで、そのまま使えるようになっている。

日本以外では、人のお仕着せが好まれない。日本だと、ある程度使えるようになっているほうが好まれる。特にあまりLinuxに馴染んでいないうちは。Vineのターゲットはどちらかというとそのへんにある。

染谷: Red Hatは逆にクライアントで使うとしたら、かなり敷居が高い。何もカスタマイズしていないから、設定するのにはかなりの能力が必要だ。

――そこが方針の違いでしょうか?

染谷: Red Hatは、グローバライゼーションというか、全部デフォルトに設定されている。

鈴木: 世界的には違うけど、日本では標準なのがそこで、なるべく本当は世界標準に

したいけど、僕らが言ってもなかなか通じない。

染谷：日本のコミュニティで協力してくれる人をみんなでまとめて、大きな声で言ってしまったら一気に入るのではないか。7月より日本のオープンソース界をリードするエンジニアが4名入社したが、現在VineやDebianなど、ほとんどのディストリビューションの人に協力いただいている。

鈴木：僕はふだんVineを作っているけど、いつも話をしているのはDebianの人ばかりだし、そのあたりの良いところをうまく取り入れて、みんな一緒に世界に認めさせよう。レッドハットに入ったことで、レッドハットの名前を利用できるというメリットもできた。

染谷：一番考えているのは、合併とか統合とかではなくて、今はLinuxを普及させていかないといけないということだ。サーバ分野では十分だけど、クライアントとかではLinuxはまだまだだ。

企業（レッドハット）だけでなくて、コミュニティも活躍していて、一緒に連係して行動することで普及が早くなるんじゃないか。実際、鈴木さんはJLA（日本Linux協会）の理事もやられているし。

――Vine Linuxの開発体制は何名で行われているのか？

鈴木：Project Vineは7名、開発版のVineSeedは約90名なので、全部で100名ぐらいだ。VineSeedは、全部を担当する人と、AlphaやSPARC、PowerPCといったアーキテクチャごとの人がいる。

――その人たちのあいだで、今回の件は議論にならなかった？

鈴木：あるように思いますよね。でもなかった。良いほうに解釈している人がほとんどで、売却されたんじゃないかといった悪いほうの意見は1件もなかった。

染谷：オープンソースをやっていれば、別にそんな不思議なことではない。オープンソースだからこそ、優秀な技術者とか、できる人ほどアピールしやすいという面がある。しかし、そういう人に限ってお金が儲からなくて、生活に苦しんでしまう。Red Hatは、そういう人たちに、Linuxで食べていけるように手助けをしている。実際、米国のRed Hatにはカーネルの開発者などのエンジニアがたくさんいる。

鈴木：技術評論社とはケンカ別れしたわけではないし、Vine Linuxのユーザーには引き続き、パッケージが販売されるのだから。

染谷：今回の協力によって、Project Vineの人は開発中のOSに早く触れるようになったというメリットもある。

――では、Project Vineに参加している人たちは、これまでと変わらない？

染谷：レッドハットは、Vineのやり方に口出しはしない。完全に委託販売としている。商売だから、経費がかかる部分は明朗会計でやろうということだ。

鈴木：単に、技術評論社がレッドハットに置き換わっただけ。Project Vineだけでなく、VineSeedやVineユーザーは何も変わらない。

染谷：そうでないと、今使っている人の混乱を招くと思う。

鈴木：元々、売るのが目的ではない。欲しい人がいるからパッケージで出している。

染谷：配布形態として日本ではトラディッショナルだし、FTPサイトに置いておくだけでもよいのだが、ユーザーがそれを望んでいる。

――Vine Linuxは本誌や日刊アスキーLinuxのアンケートではユーザーが多いようだが。

鈴木：パッケージの販売数より、実際に使用している人はかなり多いようだ。今回のリリースを出した日のWebサイトへのアクセスが20万ヒットにもなった。vine-usersのメーリングリストも1万人弱のメンバーがいて、配送に多くの時間がかかっている。最初の頃に加入した人と最近の人では半日も違う。

染谷：Vineは、最初は個人ユースをターゲットでやってきて、気に入ってくれた人が企業内でも利用しようということになった。しかし、企業で利用するためにはサポートが必要で、実際には企業対企業でないと契約できないから導入できないという問題がある。コミュニティ（Vine）側は、そういう意味でレッドハットを利用すればいい。お互いの趣旨を曲げなければ混乱はないはずだ。

――次のVine Linux 2.1は、Red Hat Linux 7.0ベース？

鈴木：バージョン番号からわかるように、7ベースではない。Vineは地雷を踏むことはしたくないので、いま一番安定しているものを使う。そうすると7まではいかないが、6.2ではちょっと古い。そのあいだということになる。

Sendmailとwu-ftpdは、一番セキュリティホールが見つかっているソフトで、次のバージョンでは入れ換える予定だ。これらのソフトはセキュリティなんて、だれも気にしていない時代の代物なので問題が多い。Sendmailに代えて採用するPostFixは、プログラミングミスがあったとしても、セキュリティ上の深刻な問題やバグにならない設計になっている。またライセンスもしっかりしていて、性能もほかの速いといわれているものと比べても遜色ない。

――リリース次期は？

鈴木：リリースロードマップよりも少し遅れそうだ。Vine Linux 2.1のパッケージは11月になるだろう。今度は、Intel版のほかに、PPC、Alpha、SPARCの同時収録を目指している。

（聞き手：編集部　木下）

鈴木 大輔氏
Project Vine代表、日本Linux協会理事
レッドハット株式会社プロフェッショナルサービス

Products

Webブラウザで簡単に設定できるオールインワンLinuxサーバ

Altos SA50

インターネットに接続し、Webサーバやメールサーバとして使うことはもちろん、Windowsファイル共有やDNS、DHCPサーバ、プロキシサーバなどの多くの機能を備えていて、専任の管理者がいなくても利用できるコンパクトサーバである。

製品名	Altos SA50
価格	12万9800円
問い合わせ先	日本エイサー株式会社 TEL 048-290-1816 http://www.acer.co.jp/

日本エイサーから、Linuxを使用したサーバアプライアンスAltos SA50（以下SA50）が発売された。SA50は、縦型のコンパクトなケースの上部に小型の液晶表示部を持ち、インターネットへ接続する場合の基本的なサーバ機能をすべて備えている。

SA50は、CPUにCeleron 500MHz、メモリ64Mバイト、15Gバイトのハードディスクを装備している。オプションのハードディスクを増設し、ミラーリングを行うことでデータの安全性を高めることが可能になっている。

シリアルポートにモデムやTAを接続すれば、ダイヤルアップでプロバイダへ接続することができる。ネットワークポートは、ローカルネットワーク用とグローバルエリアネットワーク用の2ポート用意されている。これらを使ってインターネットに接続する場合に必要な、IPマスカレード（NAT）やパケットフィルタリング機能も備えている。

サポートは、1年間のオンサイトサービスと3年間のパーツ保証で、オプションで3年間のオンサイト保守アップグレードサービスが用意されている。

Webベースのシステム管理

まずローカルネットワーク用ポートからLANに接続する。そして、前面のスイッチ押すと上部のLCDディスプレイに起動中のステータスメッセージが表示される。**写真3**のようにIPアドレスとドメイン名が表示されればサービス開始である。

別に用意したWindows PCに、SA50に付属するセットアップフロッピーを挿入し、Setup.exeを起動する。同一LANに接続されているSA50を探

写真1　Altos SA50のマザーボード
マザーボード上の写真右側には、通常あるべきVGA出力用などの部品が載っていない部分が多い。

写真2　Altos SA50背面コネクタ
キーボード、マウス、VGAモニタなどのコネクタはなく、各種設定はLANで接続した別のPCから行う。

写真3　液晶表示部とAltos SA50内部
16文字×2行のLCDディスプレイにシステムステータスを表示する。LCDへはDOCカードからケーブルで接続されている。

し出してくれるので、それを選びIPアドレスやサーバ名、ドメイン名などを設定する。このとき、特殊なプロトコルを使用しているため、すでに設定されているIPアドレスやネットマスクに関係なく利用できるのは便利な点だ。

次に、設定したIPアドレスを使って、http://<IPアドレス>/admin/というURLで、WebブラウザからSA50のシステム管理ツールを呼び出す。メニューはBasicとAdvanced、Helpに分かれていて、最初はBasicが表示される。Basicにはユーザー管理とファイル共有設定、システムクロック設定の3つの機能があり、この画面を設定するだけでとりあえずファイルサーバとして使うことができる。

Advanced（**画面1**）では、インターネットへの接続、プリンタ、メール、Webサーバ、FTPサーバ、プロキシ・キャッシュサーバ、DNSサーバの詳細設定と、システムログの閲覧、それぞれのサービスの起動／停止の設定が行える。

Linuxをフラッシュ ROMに搭載

DOC（Disk On Chip）カード（**写真4**）上のフラッシュROMにLinuxを搭載しているため、システムの起動が高速であり、PCで構築したLinuxマシンと違って、ハードディスクにインストールしたLinuxシステムファイルを不用意に書き換えてしまうことがない。また、ハードディスクの故障によって、プログラムが起動できなくなってシステムがクラッシュしたりするといった心配もない。

SA50には、キーボードやマウス、VGAモニタ出力のコネクタが付いておらず、telnetで接続することも禁止されていて、必ずLANを経由したWebブラウザで接続して設定することになる。そのため、たとえ管理者ユーザーであっても、SA50の設定機能以外のファイルを見ることはできない。接続できるIPアドレスを制限することもでき、セキュリティが重視されていることがわかる。

初めてインターネットサーバを構築するのなら、SA50を利用して短期間に低コストで導入できるだろう。

画面1　Acerシステム管理マネージャ
WebブラウザでSA50に接続すると、各種機能をメニューで簡単に設定することができる。

写真4　DOC（DiskOnChip）カード
16Mバイトのフラッシュ ROM（DiskOnChip）にLinuxが搭載されている。10/100Base-TとLCDインターフェイスも内蔵。

CPU	Celeron 500MHz
RAM	64Mバイト（最大256Mバイト）
ハードディスク	15Gバイト（2台まで増設可能）
シリアルポート	RS-232C×1ポート
パラレルポート	1ポート
PCIスロット	3スロット（空き1スロット）
ネットワーク	10/100BASE-T×2ポート
OS	フラッシュROMにLinuxプレインストール
LCDディスプレイ	16文字×2行
サイズ（mm）	95（W）×360（D）×300（H）
電源	200W

表1　Altos SA50の主な仕様

OpenBlockS

イーサネットインターフェイスを2つ備え、ローカルルータとして利用できる超小型Linuxサーバが発売された。コンパクトフラッシュカードや、2.5インチハードディスクを追加し、ソフトウェアをインストールすればWebやメールサーバとしても利用できる本格派だ。

製品名	OpenBlockS
価格	4万9800円
問い合わせ先	ぷらっとホーム株式会社 TEL 03-3251-2600 http://www.plathome.co.jp/

OpenBlockSは、ぷらっとホームが販売する超小型Linuxサーバだ。イーサネットポートを2つ備えており、そのままでIPマスカレード／簡易ファイアウォールサーバとして利用することができる。また、オプションのコンパクトフラッシュ（CF）カード、あるいは2.5インチハードディスクを増設することで、Webやメール、DNSなどの各種ネットワークサーバを動作させることができる。

システムの構成

OpenBlockSは、手のひらに載るほどの大きさだが、Linuxカーネル2.2.13が動作する正真正銘のLinuxマシンだ。標準Cライブラリは、glibc 2.1.2で、利用可能なコマンドやツールはサーバ運用に必要なものに限られている。Linuxシステムとしては、非常に簡素なものといえよう。

ハードウェアは、本誌2000年7月号のこのコーナーで紹介したFutureNet eServer-200とほぼ同等で、CPUに組込用のMotorola PowerPC 860Tを採用し、Linuxのシステムソフトウェアを4Mバイトのフラッシュ ROMに収めている。メインメモリは、16Mバイトだ。電源は、付属のACアダプタを利用する。

標準構成で、telnetサーバ、DHCPサーバ、ルータ機能、パケットフィルタ、IPマスカレード機能が利用できる。ただ、残念なことに現在販売されているOpenBlockSにはDHCPクライアント機能がない。このため、DHCPでネットワーク情報を配布するCATVインターネットなどのサービスでは利用できない。ぷらっとホームでは、DHCPクライアント機能が利用できるファームウェアを開発中とのことなので、近い将来この問題も解消するだろう。

2つのイーサネットポートは、10BASE-Tおよび、10BASE-T／100BASE-TXに対応する。また、シリアルポートには、いわゆるRJ-45モジュラジャックを採用しているが、Dsub 9ピンコネクタへの変換ソケットとケーブルが付属している。

システム設定

ネットワークやサーバの設定は、ネットワークを介しWebブラウザで行う。また、telnetやシリアル接続でログインし、コマンドラインから設定することも可能だ。

Webベースの設定ツールでは、ユーザーの追加、IPアドレスなどネットワーク環境の設定、IPマスカレードやパケットフィルタの設定のほか、DHCPやWebなどのサーバ設定をすることもできる。基本的には、PCによるLinuxサーバの設定と同じなの

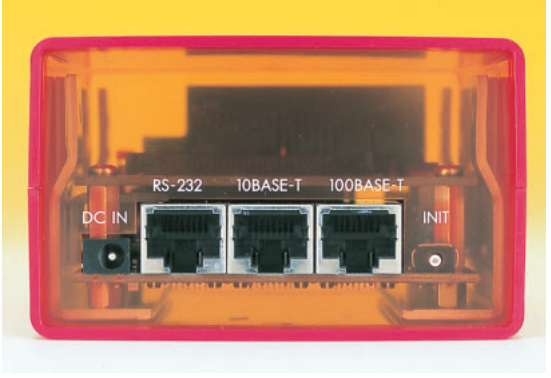


写真1　コネクタ側
左から、電源、シリアル、イーサネット（eth1）、イーサネット（eth0）、リセットスイッチ。

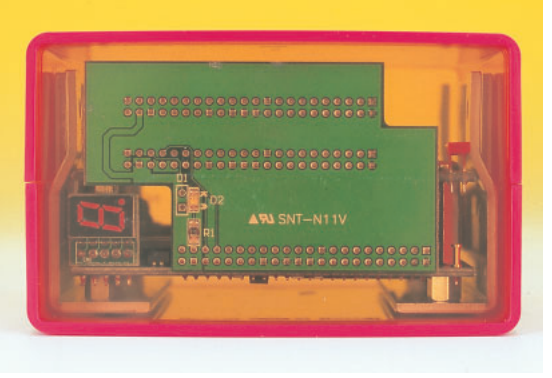


写真2　インジケータ側
サーバの状態を示す7セグメントLED、CFカードやIDEハードディスクのアクセスLED、CFカード装着時に点灯するLEDが並ぶ。

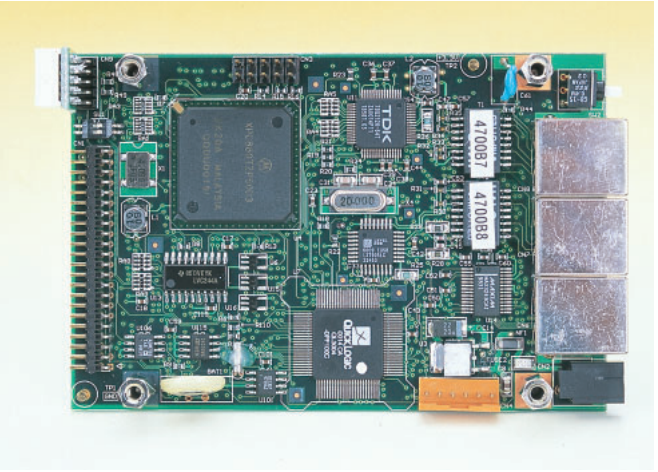


写真3　メイン基板
CPUは周辺装置コントローラを内蔵した組込用PowerPC 860Tだ。アルミの板を通じ、シャーシへ放熱される。

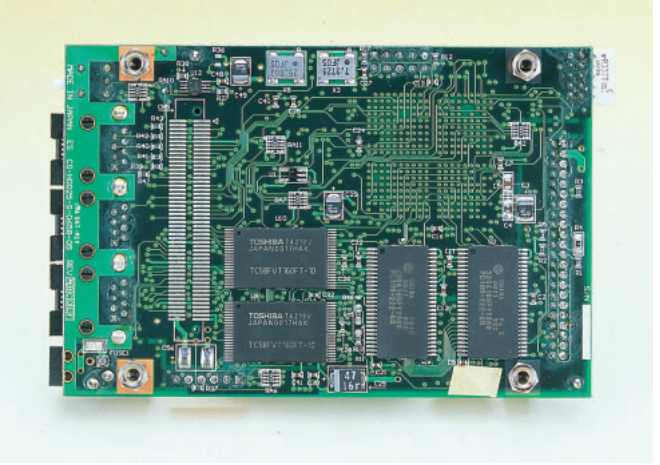


写真4　メイン基板（裏面）
カーネルや各種プログラムは、4Mバイト分のフラッシュROMに書き込まれている。メインメモリとなるRAMは16Mバイトが実装される。

で、Linuxになじみ深い本誌の読者なら迷うことはないだろう。

コマンドラインを利用すれば、より細かい設定ができる。設定ファイルは、PCで動作するLinuxとまったく同じであるのはいうまでもない。

高い拡張性

OpenBlockSは、CFカードスロットと、2.5インチIDEハードディスクコネクタを装備している。標準構成では「NAT BOX」といわれるIPマスカレード機能付きの低価格イーサネットルータとほぼ同じ機能となるが、CFカードやハードディスクをユーザー領域として追加することで、Web、DNS、メール、PPP、FTPサーバとしても動作させることが可能となる。

これらのサーバソフトウェアは、オンボードのフラッシュROMに圧縮されて格納されており、標準構成では利用できないが、拡張ストレージを接続すれば利用可能になる。

Linuxユーザーにイチオシ！

OpenBlockSをNAT BOXとして利用するだけなら、標準構成のままで十分だし、Linuxの知識もあまり必要ない。コンパクトで低消費電力、ファンレスのため騒音も皆無という理想的なNAT BOXとして活躍するだろう。

しかし、本誌の読者なら、CFカードやハードディスクを増設し、本格的なLinuxサーバとして活用してほしい。MacintoshでLinuxPPC環境が利用できるなら、比較的簡単にOpenBlockSで動作するバイナリを作ることができる。ぷらっとホームオープンラボラトリーのWebページ（http://openlab.plathome.co.jp/）には、OpenBlockSの開発環境をLinuxPPC上に構築する方法が解説されているので参考になるだろう。

さらに、Linuxの上級者であれば、クロスコンパイルが可能なgccをビルドすれば、x86プラットフォーム上でOpenBlockSの開発環境を構築できるかもしれない。

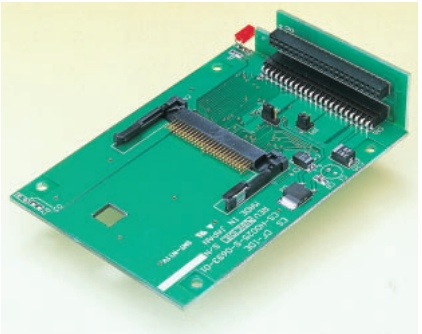


写真5　拡張インターフェイス基板
CFカードスロットと、IDEハードディスクコネクタを標準装備。2.5インチのハードディスクを利用可能だ。

製品名	OpenBlockS
CPU	PowerPC 860T 50MHz
ROM（Flash）	4Mバイト
RAM	16Mバイト
シリアルポート	RS-232C×1ポート
イーサネット	10／100BASE-T×1ポート 10BASE-T×1ポート
拡張インターフェイス	IDEインターフェイス×1 コンパクトフラッシュカードインターフェイス×1
LED	7セグメントLED（電源、ステータス表示）
OS	Linux 2.2.13
Cライブラリ	glibc 2.1.2
ストレージオプション	2.5インチハードディスク コンパクトフラッシュカード
外形寸法（mm）	85（W）×120（D）×50（H）
最大消費電流	2.5A（DC 5V、起動時）

表1　OpenBlockSの主な仕様

『/etcダンジョン』徹底探索!

Linux 設定ファイルガイド

システム起動から、ハードウェア設定まで……
知らなかった/etcの謎が今、解き明かされる。
/etcの謎をすべて解き明かす冒険の書ついに公開!

illustration : Chata Tachibana

/etcディレクトリの内容を知ろう

文：編集部
Text : Linux magazine

Linuxはカーネルを中心にして、デバイスドライバやデーモンなど、数多くのソフトウェアが連携しながら1つのシステムとして動作している。さらには、その上でX Window Systemやユーザーの使うアプリケーションソフトウェアなどが動くのだ。これらのソフトウェアの設定はどのようにして行われているのだろうか。ほとんどのソフトウェアが、設定ファイルを持っており、その中で指定された内容に従って動作する。

設定ファイルの多くは、/etcディレクトリに置かれている。実際に/etcディレクトリを見てみると実に多くのファイルが存在していることがわかるだろう。たとえば、Red Hat Linux 6.2Jでは、/etcディレクトリ以下には、98個のサブディレクトリと1315個のファイルが収められている。

これらのファイルは、システムの中心となるソフトウェアの設定ファイルだったり、一部のアプリケーションソフトウェアが利用する重要なデータファイルだったりする。個々のファイルの役割も、その内容の記述方法もバラバラで、ファイル数もうんざりするほど多いが、/etcディレクトリの内容を

名前	R	K	V	L	T	説明	ページ
CORBA/	○	○	○	○	○	CORBA（Common Object Request Broker Architecture）に関連したファイルを格納する	—
DIR_COLORS	○	○	○	○	○	カラーlsコマンドの表示色の設定ファイル	65
HOSTNAME	○	○	○	○	○	FQDN（Fully Qualified Domain Name）形式でホスト名が書かれているファイル	71
Muttrc	○	○	—	—	—	メールソフトウェアMuttの設定ファイル	80
TextConfig	○	—	○	○	○	コンソール画面の解像度、カーソル形状、フォント情報	68
X11/	○	○	○	○	○	X Window Systemの設定ファイル群を格納するディレクトリ	68
adjtime	○	○	○	○	○	マシンの時刻を設定するhwclockのパラメータファイル	68
aliases	○	○	○	○	—	メールのエイリアス設定ファイル	75
aliases.db	○	○	○	○	—	メールのエイリアスデータベースファイル	75
amd.conf	○	—	—	○	—	amdデーモンの設定ファイル	72
amd.net	○	—	—	○	—	amdデーモンの設定ファイル	72
anacrontab	○	—	—	○	—	24時間稼働しないシステムのためのcron	72
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	73
auto.master	○	○	○	○	○	automountデーモンの設定ファイル	72
auto.misc	○	○	○	○	○	automountデーモンが利用するマップファイル	72
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	65
cdrecord.conf	○	—	—	○	○	cdrecordコマンドの設定ファイル	—
charsets/	○	○	—	—	—	Muttの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	80
codepages/	○	○	○	○	○	Sambaで利用するコードページデータを格納するディレクトリ	79
conf.linuxconf	○	○	○	○	—	linuxconfのモジュールを記述したファイル	68
conf.modules	○	○	○	○	○	modprobeが利用する、ローダブルモジュール設定ファイル（TurboLinuxはmodules.conf）	64
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	73
cron.daily/	○	○	○	○	○	1日に1回自動実行させたいスクリプトを格納するディレクトリ	73
cron.hourly/	○	○	○	○	○	1時間に1回自動実行させたいスクリプトを格納するディレクトリ	73
cron.monthly/	○	○	○	○	○	1カ月に1回自動実行させたいスクリプトを格納するディレクトリ	73
cron.weekly/	○	○	○	○	○	1週間に1回自動実行させたいスクリプトを格納するディレクトリ	73
crontab	○	○	○	○	○	crontabコマンドの設定ファイル	73
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	65
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	65
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	69
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	76
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	80
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	77
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（Red Hat Linuxのみ）	68

押さえれば、Linuxの運用管理は思いのままにできる。そこで今回は、/etcディレクトリの内容を一挙に説明しよう。

解説の対象とするディストリビューションは、Red Hat Linux 6.2J、Kondara MNU/Linux 1.2、LASER5 Linux 6.2、Vine Linux 2.0、TurboLinux Workstation 日本語版6.0だ。/etcディレクトリ以下のファイルは、一部を除きほかのディストリビューションでも大きな違いはない。ただし、Slackware LinuxやPlamo Linuxは、システム起動に関わるファイルの配置や内容が異なるので、これらのディストリビューションについてもあわせて解説する。

解説をするにあたり、次に示す6つのジャンルを設定し、ジャンルごとに重要なファイルやディレクトリを説明していくことにする。

- **システム起動**
- **シェル**
- **システム設定**
- **デーモン**
- **ネットワーク**
- **アプリケーションなど**

このジャンル分けは、あくまでも便宜上行ったものであり、すべてのファイルがこのとおりに分かれるものではない。**表**にファイル一覧と、簡単な説明をまとめたので、わからないファイルを調べる際のインデックスとして活用してほしい。

それでは、/etcのダンジョンに分け入ってみようではないか。/etcディレクトリの内容をマスターすれば、「無敵のLinuxer」の称号はアナタのものだ。

表見出し中のディストリビューション略称
R Red Hat Linux 6.2J
K Kondara MNU/Linux 1.2
V Vine Linux 2.0
L LASER5 Linux 6.2
T TurboLinux Workstation 6.0

名前	R	K	V	L	T	説明	ページ
fdprm	○	○	○	○	○	フロッピーディスクのパラメータファイル	69
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	69
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	63
ftpaccess	○	○	○	○	○	FTPサーバの設定ファイル	76
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	76
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	76
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	76
ftpusers	○	○	○	○	−	FTPアクセスを禁止するユーザを設定するファイル	76
gated.conf.sample	○	−	○	○	−	動的経路設定デーモンgatedの設定ファイル見本	77
gettydefs	○	○	○	○	−	gettyが利用できるシリアルの設定一覧	−
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	−
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	69
group	○	○	○	○	○	ユーザーグループを定義したファイル	69
group-	○	○	○	○	○	groupファイルのバックアップ	69
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	70
gshadow-	○	○	○	○	−	gshadowのバックアップ	70
gtk/	○	○	○	○	○	The GIMP ToolKit（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	−
host.conf	○	○	○	○	○	名前解決の方法、優先順位の設定ファイル（libc5）	74
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	74
hosts.allow	○	○	○	○	○	inetd経由で起動されるサーバのアクセス制御ファイル	76
hosts.deny	○	○	○	○	○	inetd経由で起動されるサーバのアクセス制御ファイル	76
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	77
identd.conf	○	○	○	○	−	個人認証に使われるidentdの設定ファイル	77
im_palette-small.pal	○	○	○	○	○	画像イメージ描画ライブラリimlibのパレット設定ファイル	−
im_palette-tiny.pal	○	○	○	○	○	画像イメージ描画ライブラリimlibのパレット設定ファイル	−
im_palette.pal	○	○	○	○	○	画像イメージ描画ライブラリimlibのパレット設定ファイル	−
imrc	○	○	○	○	○	画像イメージ描画ライブラリimlibの動作設定ファイル	−
inetd.conf	○	○	○	○	○	インターネットスーパーサーバinetdの設定ファイル	75
info-dir	○	○	○	○	○	infoコマンドがデフォルトで表示するファイル	80
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	52
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）	−
irda/	○	−	−	○	−	赤外線通信を行うためのirda-utilsパッケージの初期化スクリプト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	69

名前	R	K	V	L	T	説明	ページ
isdn/	○	—	—	○	—	ISDNカードサポートの設定ファイルを格納。ただし、日本のISDNでは利用不可	—
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	66
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	66
kon.cfg	○	—	○	○	○	日本語コンソールエミュレータkonの設定ファイル	66
krb5.conf	○	—	—	—	—	ケルベロスの設定ファイル	—
ld.so.cache	○	○	○	○	○	共有ライブラリを検索する際に用いられるキャッシュファイル	71
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	71
ldap.conf	○	○	—	○	○	LDAP（Lightweight Directory Access Protocol）モジュールnss_ldapの設定ファイル	75
lilo.conf	○	○	○	○	○	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	—
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	79
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	—
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	80
logrotate.d/	○	○	○	○	○	RPMファイルからインストールしたデーモンのためのlogrotate設定ファイルを格納する	81
ltrace.conf	○	○	—	○	○	ライブラリトレーサの設定ファイル	—
lvs.cf	○	—	—	○	—	Red Hatクラスタリングサービスのデーモンコントロール設定ファイル	—
lynx.cfg	○	○	○	○	○	テキストベースWebブラウザLynxの設定ファイル	—
mail/	○	○	○	○	○	Sendmailが利用するデータベースファイルなどを格納する	—
mail.rc	○	○	○	○	○	mailコマンドの設定ファイル	—
mailcap	○	○	○	○	○	metamail設定ファイル（マルチメディアメールの処理）	—
mailcap.vga	○	○	○	○	○	metamail設定ファイル（マルチメディアメールの処理）	—
man.config	○	○	○	○	○	manコマンドの設定ファイル	—
mc.global	○	○	○	○	○	ファイルマネージャGNU Midnight Commanderの設定ファイル	—
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プレイヤの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-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	—
mime-magic.dat	○	○	○	○	○	mime-magicから作られるデータベースファイル	—
mime.types	○	○	○	○	○	WebサーバApacheが利用するMIME定義ファイル	77
minicom.users	○	○	—	○	○	シリアル通信プログラムminicomのユーザー設定ファイル	—
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	66
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	63
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	78
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	81
named.boot	○	○	○	○	○	DNSサーバBIND 4の設定ファイル	77
named.conf	○	○	○	○	○	DNSサーバBIND 8の設定ファイル	77
news/	○	—	○	○	○	NetNewsサーバINNの設定ファイルを格納する	77
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	—
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	78
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	74
ntp/	○	○	○	○	—	Network Time Protocol（NTP）を利用するためのxntp3パッケージがデータを格納する	78
ntp.conf	○	○	○	○	—	Network Time Protocolサーバxntpdの設定ファイル	78
nwserv.conf	○	○	○	○	—	mars（Netwareエミュレータ）設定ファイル	78
nwserv.stations	○	○	○	○	—	mars用Netwareステーション設定ファイル	78
openldap/	○	○	—	○	—	OpenLDAP（Lightweight Directory Access Protocol）の設定ファイルなどを格納する	78
pam.d/	○	○	○	○	○	PAM（Pluggable Authentication Modules）の設定ファイルを格納する	70
paper.config	○	○	○	○	○	用紙サイズの定義ファイル	—
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	70
passwd-	○	○	○	○	○	passwdのバックアップ	70
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/	○	○	○	○	○	pcmcia（PCカード）の設定ファイルを格納するディレクトリ	69
phhttpd.conf	○	—	—	—	—	HTTPアクセラレータphhttpdの設定ファイル	—
pine.conf	○	—	—	—	○	メール／NetNewsリーダpineの設定ファイル	—
pine.conf.fixed	○	—	—	—	○	メール／NetNewsリーダpine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	—

名前	R	K	V	L	T	説明	ページ
ppp/	○	○	○	○	○	PPPデーモンの設定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	67
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	67
protocols	○	○	○	○	○	プロトコル設定ファイル（変更不可）	—
pwdb.conf	○	○	○	○	○	pwdbの設定ファイル	70
pxe.conf	○	—	—	○	—	リモートブートデーモンpxeの設定ファイル	78
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	52
redhat-releaseなど	○	○	○	○	—	ディストリビューションのリリース名。issueに転記される	—
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	74
rmt@	○	—	○	○	○	/sbin/rmtへのリンク。リモートバックアップ時に用いられる	71
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	71
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	—
screenrc	○	○	○	○	○	画面管理プログラム、screenの設定ファイルテンプレート	68
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	67
security/	○	○	○	○	○	PAM（Pluggable Authentication Modules）で利用するモジュールを格納する	70
sendmail.cf	○	○	○	○	○	Sendmailの動作設定ファイル	78
sendmail.cw	○	○	○	○	—	Sendmailでホストの別名を利用する場合に別名を定義するファイル	78
sendmail.mc	○	○	—	○	—	sendmail.cfを作るためのマクロ定義ファイル。マクロプロセッサm4で利用する	78
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	75
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	71
shadow-	○	—	—	○		shadowのバックアップ	71
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	67
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	71
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	79
smbpasswd	○	○	○	○	○	Sambaのパスワードファイル	79
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	79
smrsh/	○	○	—	○	—	Sendmail用制限シェルで利用できるプログラムを格納する	78
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	79
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	—
squid/	—	○	○	○	○	Webキャッシュプロキシ Squidの設定ファイルを格納する	79
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	60
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	—
termcap	○	○	○	○	○	以前使われていた端末定義データベースファイル	81
up2date.conf	○	—	—	—	—	up2date設定ファイル	—
uucp/	○	○	○	○	○	UUCP（Unix to Unix CoPy）の設定ファイルを格納する	—
vfontcap@	○	○	○	○	○	Vflibで使用するTrueTypeフォントの設定ファイル	—
vga/	○	○	○	○	○	SVGALIBが利用するキーマップやビデオ設定ファイルを格納するディレクトリ	—
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	82
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	○	—	—	○	—	ytalk設定ファイル	—
zlogin	○	○	○	○	○	zshの設定ファイル	—
zlogout	○	○	○	○	○	zshの設定ファイル	—
zprofile	○	○	○	○	○	zshの設定ファイル	—
zshenv	○	○	○	○	○	zshの設定ファイル	—
zshrc	○	○	○	○	○	zshの設定ファイル	—

システム起動

文：竹内充彦
Text : Michihiko Takeuchi

ここでは/etcのなかでも、Linuxの起動に関わる部分を探検してみることにしよう。電源オンからログインまでのLinux起動時の流れについては、ディストリビューションごとに若干の差異はあるものの、おおむね**図1**のような順序になる。図中の①はハードウェアによるチェックだ。②については/etc/lilo.confの項を参照してほしい。

カーネルが読み込まれて、ハードウェアのチェック、ルートファイルシステムの立ち上げ、初期化が完了すると、システムは最初のプロセスinitを実行する。Linuxでは常にたくさんのプロセスが実行されているが、起動時に最初に実行されるのがこのinitである。initが最初に実行されるプロセスであることは、ps axコマンドの結果表示で、常にプロセスIDが1になっていることからも確認できる。このinitの仕組みを知ることで、デフォルトのrunレベルの変更や、端末の起動数、各種デーモンの起動・停止を設定することができるようになる。また、新たなサービスを提供したい場合などにも、どこを書き換えればいいかわかるようになるはずだ。initに関わるファイルやディレクトリとして、以下のものを取り上げる。

```
inittab
rc.d/
sysconfig/  (Red Hat系)
```

さらにここでは、ログイン後に起動するログインシェルが参照する環境設定ファイルとして、以下のものについても簡単に探ることにする。

```
profile
profile.d/
bashrc
csh.cshrc
csh.login
```

なお、最後にハードウェア設定に関するファイルを取り上げる。

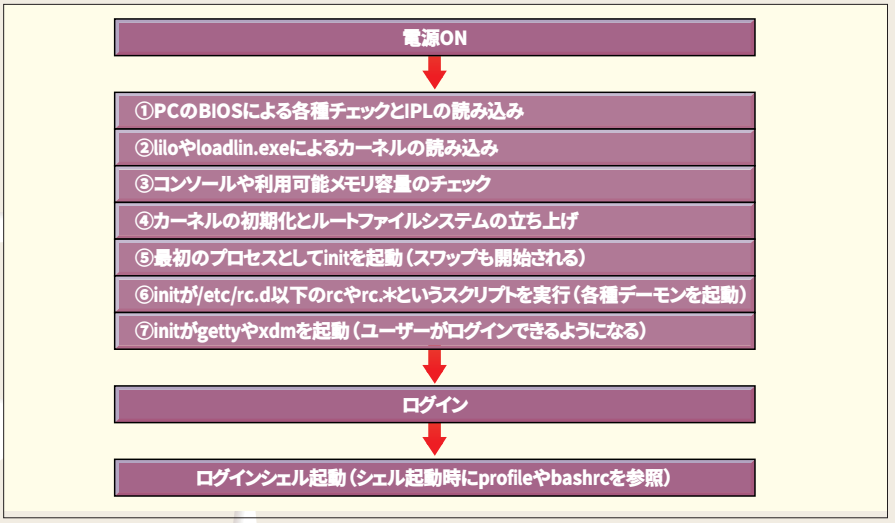


図1　電源ONからログインまでの流れ

/etc/inittabと/etc/rc.d

冒頭で起動時にinitが実行されると書いたが、実はinitが実行されるのは何も起動時だけとは限らない。shutdownコマンドから呼び出されることもあれば、ユーザーが直接実行することもできる。

initは、実行時に/etc/inittabという設定ファイルを参照する。このinittabには、runレベルに応じて実行すべきスクリプトファイルが記述されている。また、Ctrl＋Alt＋Delキーが押されたり、UPS（無停電電源装置）からの停電の通知など、システムがある種のシグナルを受け取ったときのコマンド実行についても記述されている。

このうち、runレベルごとのスクリプトファイルついては/etc/rc.d以下にまとめられている。したがって、init、inittabと、rc.dは切っても切れない関係なのだ（**図2**）。ただし、このinittabとrc.d以下のスクリプトファイルの構成については、ディストリビューションごとに異なる。ここでは、現在、最大派閥のRed Hat系ディストリビューションと、オーソドックスな構成を持つSlackware系ディストリビューションの2つについて、その仕組みを探る。

runレベルとは？

ここで、runレベルについて知っておこう。runレベルとは、Linuxの実行状態を定めたもので、そのレベルは0〜6までと、S（またはs）がよく使われ

る（それ以外にもA～Cが使われることもある）。このレベルは高低という種類のものではない。0は停止（halt）、6は再起動（reboot）、Sはシングルユーザーモードに割り当てられている。1～5、A～C、についてはディストリビューションごとに異なるが、多くの場合、1がシングルユーザーモード、3がキャラクタベースでのマルチユーザーモード、4または5がX Window Systemベースでのマルチユーザーモードに割り当てられている（**表1**）。

たとえばRed Hat系Linuxの場合、xdm（X Display Manager）を利用して、システム起動時からX Window Systemが動作していれば、runレベル5で実行されていると考えていい。X Window Systemの設定に失敗した場合など、キャラクタベースで「login：」プロンプトを表示させたければ、runレベルを3に変更すればいいし、メンテナンス時に他のユーザーがログインしないようシングルユーザーモードに切り替えたければ、runレベルを1に変更すればいい。

runレベルはinittabで定義されている。ここで定義されていないレベルは使えないのだ。逆に、runレベル7～9も定義してしまえば使えるのだが、慣習的に使われていない。

■一時的にrunレベルを変更する

一時的にrunレベルを変更するには、LILOによる“LILO boot：”プロンプトや、shutdownコマンド、initコマンドに引数を指定することで可能だ。

システム起動時にrunレベルを変更したければ、LILOのプロンプトに、lilo.confで指定したLinux起動用のラベル名に続けてrunレベルを指定する。起動用ラベルが“linux”で、runレベル3で起動したければ、

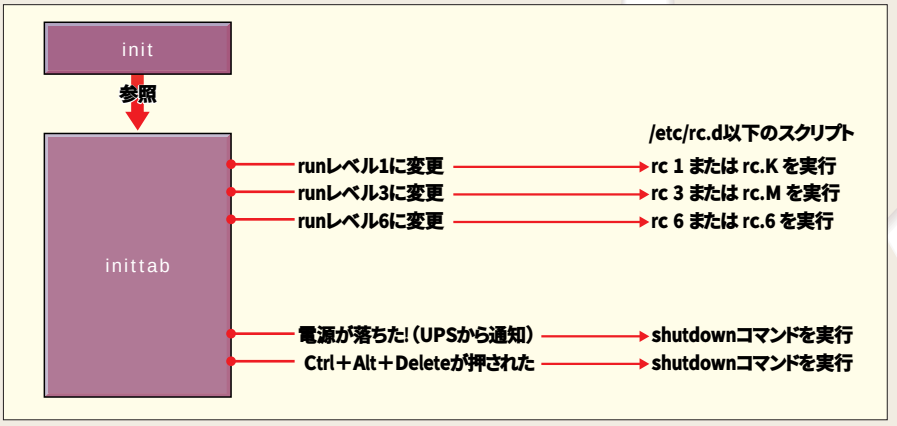


図2　initとinittabとrc.d以下のスクリプトの関係

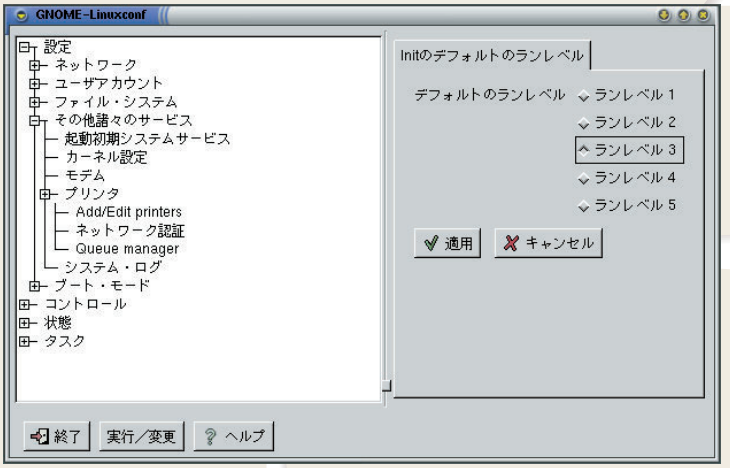
```
LILO boot: linux 3
```

と指定すればいい。

システム利用中にrunレベルを変更したければ、initコマンドかshutdownコマンドを使う。そう、ふだん使っているshutdownコマンドとは、シャットダウンに必要な諸々の処理をしてくれるのは言うまでもないが、結局のところrunレベルの変更をしているのであった。オプションに-hを指定すればrunレベル0、-rを指定すればrunレベル6、-sを指定すればrunレベル1に変更してくれるのだ。

直接initコマンドを使う場合は、引数にrunレベルを指定する。

```
# init 3
```



画面1　linuxconfで定常的にrunレベルを変更する
linuxconfではオプションボタンを選択するだけでデフォルトのrunレベルを変更することができる。

runレベル	Red Hat系	Slackware系
0	停止	停止
1	シングルユーザーモード	シングルユーザーモード
2	シングルユーザーモード（NFSなし）	未定義
3	マルチユーザーモード（テキストベース）	マルチユーザーモード（テキストベース）
4	未定義	マルチユーザーモード（X Window Systemベース）
5	マルチユーザーモード（X Window Systemベース）	未定義
6	再起動	再起動
S	未定義	シングルユーザーモード

表1　代表的なrunレベル

ただし、initコマンドではログイン中のユーザーへの告知等がされないので、誰かがログインしている可能性がある場合はshutdownコマンドを利用したい（initコマンドは、シングルユーザーモードからマルチユーザーモードへの変更に使うとよい）。

initコマンドは引数にrunレベルが指定されない場合inittabに記述されているデフォルトのrunレベルに変更する。これはシステム起動時も同様である。

■定常的にrunレベルを変更する

システム起動時のデフォルトのrunレベルは/etc/inittabに記述されている。inittabの内容は後述するとして、もし、X Window Systemを利用している場合には、linuxconfからデフォルトのrunレベルを変更することができる（**画面1**）。linuxconfのメニューツリーから［起動初期システムサービス］を選び、［デフォルトのランレベル］タブからrunレベルを選べばいい。

/etc/inittabの内容を知る

では、inittabの記述内容はどのようになっているのだろうか。inittabの記述は行単位になっている。基本的な書式は以下のようになる。

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

idフィールドは各行を識別するラベルで、1～4文字のあいだで付ける。initの旧バージョンの制限から伝統的に2文字までにしていることが多い。

runlevelフィールドには、この行を実行すべきrunレベルを指定する。1つのフィールドに複数のレベルを指定することができる。何も指定しないとすべてのrunレベルで実行される。

actionフィールドは、この行の動作を細かく指示するものだ。いくつか指示子があるので後述する。

processフィールドは、実行すべきコマンドやシェルスクリプトをフルパスで指定する。

たとえばinittab中に、

```
my:35:wait:/etc/rc.d/rc.my
```

とあれば、このmy行は、runレベルが3か5に変更されると、/etc/rc.d/rc.myというファイルを、waitという指示で実行するという意味である。

■actionフィールドの指示子

actionフィールドの指示子にはどのようなものがあるだろうか。このフィールドには**表2**のような指示ができる。よく使われるものについて、いくつか具体例を示しながら紹介しておこう。

respawnを指定すると、processフィールドの処理が終了した場合に、すぐにprocessフィールドの処理を再起動する。ユーザーがログアウトした時に、すぐにまた「login：」プロンプトが表示されたり、xdmのログインパネルが表示されるが、それはgettyやxdmといったプロセスが、このrespawnを指定して実行されているからなのだ。

onceは特に変わった動作を指示するものではない。runlevelフィールドに指定したrunレベルで、processフィールドの処理を一度実行するというものだ。waitを指示した行では、processフィールドの処理が終了するまで、次の行の処理を開始しない。たとえばネットワークの初期化が終わらないうちに、サーバサービスのプロセスを実行したりしないようにするためにこの指示を使う。

bootとbootwaitは、runレベル変更

指示子	意　味
respawn	processフィールドの処理が終了したら再起動する。gettyやxdmなどでexitしてもログイン画面になるのはこれを指定しているため
once	指定されたrunレベルになると一度だけ実行される
wait	onceと同じだが、initはprocessフィールドの処理が終了するまで次の処理を開始しない
boot	システム起動中に一度だけ実行される。runlevelフィールドは無視される
bootwait	bootと同じだが、initはこの行のprocessフィールドの処理が終了するまで次の処理を開始しない
off	このactionが指定されている行では何もしない
ondemand	runlevelフィールドに指定したオンデマンドrunレベル（A、B、C）がコールされた場合にprocessフィールドが実行される
initdefault	この行に指定したrunlevelフィールドがデフォルトのrunレベルになる。processフィールドは無視される
sysinit	この行に指定したprocessフィールドは、システム起動時にbootやbootwaitよりも前に実行される。runlevelフィールドは無視されるのですべてのrunレベルで実行される
powerfail	電源供給が断たれる（UPSのバッテリーによる電源供給が開始される）と、processフィールドの処理を実行する
powerwait	powerfailと同じだが、initはこの行のprocessフィールドの処理が終了するまで次の処理を開始しない
powerokwait	電源供給が再開されると、processフィールドの処理を実行する。initはこの行のprocessフィールドの処理が終了するまで次の処理を開始しない
powerfailnow	UPSのバッテリーが空になる直前にprocessフィールドの処理を実行する
ctrlaltdel	initがSIGINTシグナルを受け取る（コンソールでCtrl＋Alt＋Delキーが押される）とprocessフィールドの処理を実行する
kbrequest	キーコンビネーションのシグナルを受け取るとprocessフィールドの処理を実行する

表2　inittabのactionフィールドの指示子

時ではなく、電源を投入しシステムがブートする時に実行される。この指示がある行ではrunlevelフィールドは無視される。つまり、デフォルトのrunレベルを変更しても、常にこの行は実行されるのだ。ブート時にだけ実行すればよい処理があればこれを使う。

offを指示した行は、たとえ他のフィールドがどのような記述になっていても何も実行しない。

ondemandは、runlevelフィールドに指定したrunレベルA～C（これらをオンデマンドrunレベルという）になった場合に、processフィールドを実行する。

initdefaultは、デフォルトのrunレベルを指示するためのものだ。この行のrunlevelフィールドに指定されたrunレベルがデフォルトの（引数なしの場合の）runレベルになる。先ほど、linuxconfからデフォルトのrunレベルを変更したが、キャラクタベースで変更したければ、テキストエディタでこの指示がある行を書き換えればいいのだ。

sysinitを指示した行は、bootやbootwaitを指示した行よりも先に実行される。この行も、runlevelフィールドは無視され、すべてのrunレベルで実行される。ネットワーク等、システムの初期化に使われる。

powerfail、powerwait、powerokwait、powerfailnowはいずれもUPS（無停電電源装置）からのシグナルを受け取ったときに実行される。停電時に正しくシャットダウン処理をするための記述に使う。それぞれの違いについては**表2**を参照してほしい。

ctrlaltdelは、コンソールからCtrl＋Alt＋Delキーが押されたときに実行される。PCではこの組み合わせのキーが入力されると強制的に再起動するようになっている。ご存知のようにLinuxは突然システムがダウンするとファイルシステムが壊れかねない。そこで、キー入力をここでトラップし、正しくシャットダウン処理をするよう指示しておくのだ。

kbrequestもキーコンビネーションを受け取ると実行される指示だ。しかし、このキーコンビネーションのシグナルを受け取るためには、keymapファイルにキーの組み合わせとコードを設定する必要がある。

■Red Hat系のinittab

inittabの行の基本的な設定がわかったところで、実際のinittabファイルを覗いてみることにしよう（**リスト1**）。

2行目のid行では、デフォルトのrunレベルを決めている。**リスト1**では、5に指定されている。Red Hat系なので、X Window Systemベースで起動するということだ。ここを3に書き換えれば、次回からはテキストベースのマルチユーザーモードで起動するようになる。当然だが、ここに0や6を指定してはならない。起動したとたんに停止するか、シャットダウンすることになってしまうからだ。間違って指定した場合は、前述の方法でrunレベルを指定して起動し、この部分を修正すればいい。

5行目のsi行では、システムを初期化するために、/etc/rc.d/rc.sysinitというシェルスクリプトを実行している。rc.sysinitは、キーマップやシステムフ

▼リスト1　Red Hat系の/etc/inittab（抜粋）

```
 1: # Default runlevel.
 2: id:5:initdefault:
 3:
 4: # System initialization.
 5: si::sysinit:/etc/rc.d/rc.sysinit
 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
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
17:
18: # Trap CTRL-ALT-DELETE
19: ca::ctrlaltdel:/sbin/shutdown -t3 -r now
20:
21: # Run gettys in standard runlevels.
22: 1:2345:respawn:/sbin/mingetty tty1
23: 2:2345:respawn:/sbin/mingetty tty2
24: 3:2345:respawn:/sbin/mingetty tty3
25: 4:2345:respawn:/sbin/mingetty tty4
26: 5:2345:respawn:/sbin/mingetty tty5
27: 6:2345:respawn:/sbin/mingetty tty6
28:
29: # Run xdm in runlevel 5
30: # xdm is now a separate service
31: x:5:respawn:/etc/X11/prefdm -nodaemon
```

- 2: デフォルトのrunレベル。通常は3か5。当然だが0や6を指定してはいけない
- 5: rc.sysinitでシステムを初期化。いちばん最初に実行される。すべてのrunレベルに共通
- 7: runレベル0（level0）での処理。スクリプトrcに引数0を渡している
- 8: 以下level6まで同様。ここに定義すればrunレベル7～9も使える。ただし、スクリプトrcも7～9の引数を受け取るように書き換える必要がある
- 16: すべてのrunレベルで実行される
- 19: Ctrl＋Alt＋Delキーをトラップし、shutdownコマンドで再起動されるようにしている
- 22: runレベル2～5においてmingettyを起動しtty1を割り当てる
- 23: 以下tty6まで同様
- 31: runレベル5でログインパネルを表示する。respawnを指定しているのでログアウトするたびにパネルが表示される

ォントの読み込み、ネットワークの起動や、ファイルシステムのチェック等、システムの初期化を行う。

7行目～13行目のl0～l6行は、本項の主要な話題でもあるrunレベルごとのデーモンの起動と停止だ。ここで、注目しておきたいのは、runレベルに応じた引数を/etc/rcに渡していることだ。rcの中身については後述するが、initが受け取ったrunレベルを、ここでさらにrcに渡しているという点に注目しておこう。さらにwaitを指示して、rcの処理が終了するまで次の処理を開始しないようにしている。これにより、たとえばnfsが起動しないうちにX Window Systemを実行してしまうのを防いでいるのだ。

16行目のud行では、/sbin/updateを実行している。updateは、5秒ごとに遅延書き込みバッファの内容をファイルシステムに書き出す。現在のカーネルでは、updateを起動しなくても、30秒ごとにバッファの内容が書き出されるが、この指定により5秒ごとに更新され、安全性が高められている。この行は、runレベルに関係なく実行するようにrunlevelフィールドに何も指定されていない。

19行目のca行では、コンソールからCtrl＋Alt＋Delキーが入力されたときに、それをトラップしている。BIOSレベルでいきなり再起動されては困るので、shutdownコマンドを実行して再起動しようとしているのがわかるはずだ。

21～27行目では、端末を起動している。runレベル2～5のマルチユーザーモード（ただし2と4は使われないが）で、/sbin/mingettyを実行し、tty1～tty6を起動しているのがわかる。gettyは、端末が接続されるとloginを実行する。actionフィールドにrespawnを指示することで、ユーザーがログアウトする（端末の接続が切れる）と、またmingettyを実行し端末の接続を待つようにしているのだ。

31行目のx行は、runレベル5のX Window Systemモードでxdmを実行する。コメントにもある通り、現在ではxdmはサービスが分けられているため、ここではprefdmを実行している。X Window Systemでもログアウトするとすぐに、ログインパネルが表示されるが、この行のactionフィールドにrespawnが指示されているためである。

▼リスト2　Slackware系の/etc/inittab（抜粋）

```
22: # Default runlevel. (Do not set to 0 or 6)
23: id:3:initdefault:
24:
25: # System initialization (runs when system boots).
26: si:S:sysinit:/etc/rc.d/rc.S
27:
28: # Script to run when going single user (runlevel 1).
29: su:1S:wait:/etc/rc.d/rc.K
30:
31: # Script to run when going multi user.
32: rc:2345:wait:/etc/rc.d/rc.M
33:
34: # What to do at the "Three Finger Salute".
35: ca::ctrlaltdel:/sbin/shutdown -t5 -rf now
36:
37: # Runlevel 0 halts the system.
38: l0:0:wait:/etc/rc.d/rc.0
39:
40: # Runlevel 6 reboots the system.
41: l6:6:wait:/etc/rc.d/rc.6
42:
43: # The getties in multi user mode on consoles an serial lines.
44: c1:1235:respawn:/sbin/agetty 38400 tty1 linux
45: c2:1235:respawn:/sbin/agetty 38400 tty2 linux
46: c3:1235:respawn:/sbin/agetty 38400 tty3 linux
47: c4:1235:respawn:/sbin/agetty 38400 tty4 linux
48: c5:1235:respawn:/sbin/agetty 38400 tty5 linux
49: c6:12345:respawn:/sbin/agetty 38400 tty6 linux
50:
51: x1:4:wait:/etc/rc.d/rc.4
52:
53: # End of /etc/inittab
```

デフォルトのrunレベル。通常は3か4。当然だが0や6を指定してはならない

rc.Sでシステムを初期化。最初に実行される。すべてのrunレベルに共通

シングルユーザーモード（runレベル1またはS）での処理。rc.Kを実行している

マルチユーザーモード（runレベル2～5）での処理。ただしSlackwareでは3と4しか用意されていない。rc.Mを実行している

Ctrl＋Alt＋Delキーをトラップし、shutdownコマンドで再起動されるようにしている

runレベル0、すなわちシステム停止時の処理。rc.0を実行している

runレベル6、すなわちシステム再起動時の処理。rc.6を実行している

runレベル1～3、および5においてagettyを起動しtty1を割り当てる

以下、tty5まで同様

tty6のみrunレベル4でも起動していることに注目！

runレベル4、すなわちX Window Systemモードでの処理。rc.4を実行している

■Slackware系のinittab

Red Hat系ディストリビューションのinittabで、ある程度の動作が把握できたので、Slackware系のinittabで、特徴的な部分について説明しておこう。**リスト2**にSlackware 7.1のinittabの抜粋を示す。

この例では、デフォルトのrunレベルが3になっている（23行目）。テキストベースでのマルチユーザーモードだ。

システムの初期化は/etc/rc.d/rc.Sというシェルスクリプトを実行する（26行目）。

Red Hat系との最大の違いは、runレベルごとに用意されるデーモン実行のためのスクリプトだ。Red Hat系では、rcに引数としてrunレベルを渡し、すべてのレベルでrcを実行していたが、

Slackwareではrunレベルごとに（というかモードごとに）シェルスクリプトが用意されている。シングルユーザーモードではrc.Kというシェルスクリプトが（29行目）、マルチユーザーモードではrc.Mというシェルスクリプトが（32行目）、それぞれ実行される。runレベル0と6では、それぞれrc.0とrc.6というシェルスクリプトが実行される（38行目、41行目）。

43～49行目では、やはり端末が接続される準備をしているのだが、tty6だけは、runレベル4でも起動されるようになっている。これはinittabにもコメントされているが、これまではrunレベル4はX Window Systemのみのモードにしていたが、ttyを1つ起動してもロードアベレージが1にも満たないことがわかり、なにかと便利なのでこうしてあるとのことだ。気にしないで欲しいとも書いてある（しかし、Red Hat系では6つも起動しているけど、誰も気にしていないと思うのだが……）。

/etc/rc.d以下を探検する

Red Hat系ディストリビューションも、Slackware系ディストリビューションも、runレベルに応じて、inittabの設定からrc.d以下のシェルスクリプトを実行していることがわかった。では、このrc.dとはいったい何であろう。

rcとはRunlevel Changeの略で、文字通りrunレベルの変更に関わるシェルスクリプトだ。rc.dはそのうちデーモン（daemon）の起動と停止を制御するためのシェルスクリプトが収められている。

デーモンとは、バックグラウンドで動作し、サービスを提供するプログラムだ。WebサーバのhttpdやSambaのsmbdなどはわかりやすい例だろう。initから呼び出されたシェルスクリプトがいかにしてデーモンを起動するか、ここで解明すべく探っていこう。

Red Hat系の/etc/rc.d以下

リスト1でもわかるように、Red Hat系のinittabでは、すべてのrunレベルでrcというシェルスクリプトを実行していた。ただしrunレベルを引数として一緒に渡している。つまり、rcは受け取った引数をもとにして、さらにそこから実行するシェルスクリプトを選択しているのだ。

はじめにrc.d以下の構成を見てみよう。**図3**のようになっている。シェルスクリプトは3つあり、rc、rc.local、rc.sysinitとなっている。rcとrc.sysinitはinittabに設定があったので、それぞれの条件が満たされれば実行される。rc.localは、initのすべてのシェルスクリプトが実行し終わってから実行されるシェルスクリプトだ。システムに用意されていない、独自の設定をしたい場合は、できるだけrc.localに記述するようにしたい。ディストリビューションごとの独自拡張部分も、このrc.localに記述されている。

rc.d以下には7つのディレクトリがある。rc0.d～rc6.dには、rcから実行されるシンボリックリンクファイルが収められている。ディレクトリ名rc？.dの？の部分はrunレベルを示している。

実は、このシンボリックリンクのファイル名にちょっとした仕掛けがある。これらのリンクの実体は、すべてinit.d以下にあるシェルスクリプトなのだが、別名を付けてある。すべてのリンクに共通なのは、名前がKかSで始まり、それに数字が続いていることだ。

Kで始まるものは、そのrunレベルで停止するデーモン、Sで始まるものは、そのrunレベルで起動するデーモンのために用意されている。続く数字はシェルのワイルドカード展開を利用して、起動と停止の順序を調節するために付けられている。

rcの一部を参照するとわかりやすい

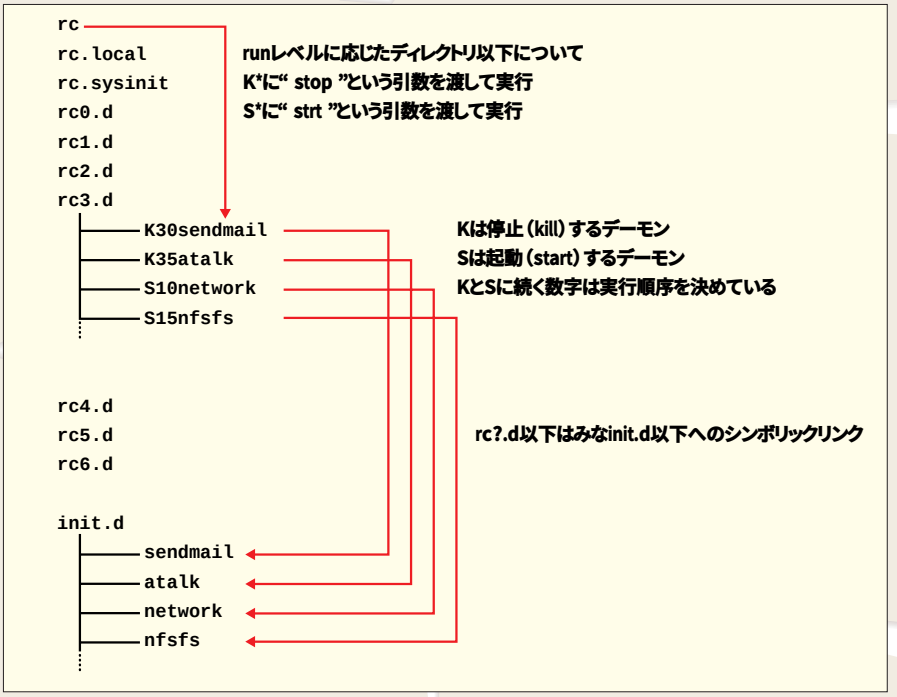


図3　/etc/rc.d以下の構成（Red Hat系）

のでリスト3に示す。リストにはないが、rcは受け取った引数から、変更するrunレベルを変数runlevelに代入する。そして41行目で、ターゲットとなるrc$runlevel.dというディレクトリが存在するかどうかをチェックしている。変数runlevelが「3」ならば、rc3.dが存在するかどうかだ。存在すればthen以下が実行される。

43行目ではfor文にin節を使い、ターゲットディレクトリ内のKで始まるファイル名をシェル機能でワイルドカード展開している。展開されたファイルを順に変数iに代入し、すべてについて59行目のdone文まで実行する。65行～100行は、デーモンの起動で、S＊について同様の処理をしている。

ここでちょっとした工夫がある。lsコマンドを使えばわかるように、シェルの機能による＊の展開の結果はASCIIコード順に並べられてしまう。シンボリックリンクのファイル名の先頭に、起動のSと停止のKしか設定しなかった場合を考えてみよう。atalkはSatalkとKatalk、networkはSnetworkとKnetworkになる。これだと、Snetworkよりも先にSatalkが実行されてしまうことになる。ネットワークサービスが起動する前に、AppleTalkのようなネットワーク上で提供されるサービスが起動するはずもなく、破綻してしまうだろう。

これを解決しているのがSとKに続く数字である。そう、数字の順に実行されるように調節しているのだ（ナント原始的な！）。これでS10networkは必ずS15atalkより先に実行されるというわけだ。さらに言えば、停止時も同様で、smb（Samba）を停止してからnetworkを停止しなければならない。この場合もKsmbより先にKnetworkが実行されては困るので、K10smbとK90networkのようにしておかなければならない。

この理屈がわかっていれば、GUIによるrunレベルエディタのようなツールが、何をしているかもわかるだろう（画面2）。

さて、起動と停止のそれぞれの繰り返し処理の中で、58行目と98行目に注目しよう。ここで、展開されたシンボリックファイルにオプションを付けて実行している。たとえばディレクトリrc3.dにあるS10networkと、rc0.dにあるK90networkは共に、ディレクトリinit.dにあるnetworkを参照している。つまり、実体であるnetworkにオプションを付けて実行することで、起動も停止もできるようになっているのだ。init.d以下のシェルスクリプトはみなこのようにオプションを指定できる。その多くは、start、stop以外にも、restart、statusというオプションも指定可能だ。restartはいったん停止してから再起動する。statusは現在の実行状態を表示する。試しに、

```
# /etc/rc.d/init.d/smb status
```

と指定すれば、現在Sambaのデーモンが実行中か停止中かがわかるだろう。

Red Hat系の場合、rc？.d以下にSで始まるシンボリックファイルがあるかどうかでデーモンの起動が決まる。これがわかれば、いろいろと応用がきく。たとえば、システムに問題が見つかって当面Webをサービスしない場合など、S85httpdの名前をs85httpd等に変更しておくだけで、起動されなくなる。先頭を小文字のsにしておくことで、起動項目であることを忘れずに済むだろう。問題がクリアされたら、元通り大文字に戻せばいいのだ。

このinit.dに用意されているシェルスクリプトは、実際には/sbinや/usr/sbinにあるコマンドをデーモンとして起動している。Red Hat系の行き届いているところは、統一されたオプションを備えるこうしたシェルスクリプトをすべて用意してくれているところだ。ユーザーは1つ覚えてしまえば、ほかのすべてに応用できるというわけだ。

▼リスト3　/etc/rc.d/rcのデーモンの起動と停止部分（Red Hat系、抜粋）

```
 40: # Is there an rc directory for this new runlevel?
 41: if [ -d /etc/rc.d/rc$runlevel.d ]; then
 42:     # First, run the KILL scripts.
 43:     for i in /etc/rc.d/rc$runlevel.d/K*; do
   :                 :
   :                （中略）
   :                 :
 57:         # Bring the subsystem down.
 58:         $i stop
 59:     done
   :                 :
 65:     # Now run the START scripts.
 66:     for i in /etc/rc.d/rc$runlevel.d/S*; do
   :                 :
   :                （中略）
   :                 :
 97:         # Bring the subsystem up.
 98:         $i start
 99:     done
100: fi
```

$runlevelには新しいrunレベルが代入されている。-dはファイル名が存在し、かつそれがディレクトリであれば真となる。then以降が実行される

K＊を展開しマッチするファイル名を順に変数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

for文に戻り繰り返し

Slackware系の/etc/rc.d以下

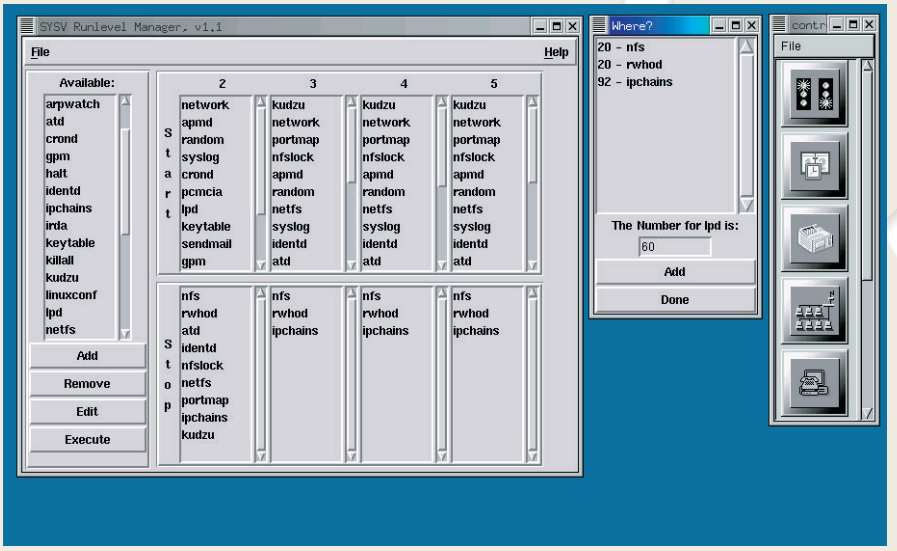
Slackware系のrc.d以下はRed Hat系と比べると、サブディレクトリもなく、雑然としているように見える。しかし、各シェルスクリプトの内容はRed Hat系よりも直感的に何をしているかが把握でき、いくつもの手がかりを手繰り寄せていくような煩わしさはない。古くからUNIX系OSを使っているユーザーにはわかりやすいだろう。rc.d以下のファイルとその内容を**表3**に一覧しておく。

rc.0～rc.6と、rc.K、rc.M、rc.Sはinittabに設定があったので、それぞれの条件が満たされれば実行される。そして、これらのファイルからその他のrc.＊ファイルが実行されるというシンプルな仕組みである。rc.0がrc.6のリンクになっているが、rc.6の内部でrunレベルに基づきhaltとrebootを分けている。

Red Hat系のようにすべてのデーモンに統一的なシェルスクリプトが用意されているわけではなく、/sbinにあるコマンドを直接起動しているものも多い。

では、具体的にどうなっているのだろうか。すべての処理がrc.Mやrc.4に集約されているので全体を紹介することは避けるが、Slackwareのrc.＊からデーモンを起動する部分は**リスト4**のように記述されている。－xで実行ファイルが存在するかどうかを確認し、メッセージを表示した後にデーモンを起動する。なんともシンプルである。

起動したくないデーモンがあったらどうすればいいだろうか。**リスト5**は、



画面2　runレベルエディタ（LASER5 Linux）
runレベルごとデーモンの起動順序、停止順序を指定する欄がある。

ファイル	内容
rc.0	runレベル0で実行される。システムを停止。実態はrc.6へのリンクファイル
rc.4	runレベル4で実行される。X Window Systemモードに変更
rc.6	runレベル6と0で実行される。runレベルに応じてシステムを停止もしくは再起動
rc.K	runレベル1かSで実行される。シングルユーザーモードに変更
rc.M	runレベル3で実行される。マルチユーザーモード（テキストモード）に変更
rc.S	システムを初期化
rc.atalk	netatalkを起動
rc.cdrom	CD-ROMドライブにCD-ROMを見つけると自動的にマウント。デフォルトでは起動されないようになっている。chmod 755で起動するようになる
rc.font	コンソールで利用するスクリーンフォントを設定
rc.font.sample	rc.fontの雛形ファイル
rc.gpm	マウスとカット＆ペーストを提供するgpmを起動
rc.httpd	httpd（Apache）を起動
rc.ibcs2	x86系CPUで動作する他OSのアプリケーション実行環境を提供するモジュールiBCS（Intel Binary Compatibility Specification）の組込み
rc.inet1	ネットワークを起動
rc.inet2	ネットワークを起動。inetdをはじめ、sshd、named、routed、yp等のデーモンを起動
rc.local	initが最後に起動。rcの独自拡張に使う
rc.modules	カーネルに拡張ドライバのモジュールを組み込む
rc.netdevice	イーサネットカード等、ネットワーク機器のドライバモジュールを組み込む
rc.news	INND（Netnewsサーバ）を起動。/usr/lib/news/bin/rc.newsへのリンクファイル
rc.pcmcia	PCMCIAカードコントローラのドライバを起動
rc.samba	Samba（smbd、nmbd）を起動
rc.serial	シリアルポートを初期化
rc.sysvinit	SystemV版UNIXとの互換を取るためのスクリプト。商用アプリケーション等で利用されることを考えて用意されている

表3　Slackware系の/etc/rc.d以下のファイル

▼リスト4　rc.Mファイルのメールデーモン起動部分（Slackware系、抜粋）

```
110: # Start the sendmail daemon:
111: if [ -x /usr/sbin/sendmail ]; then
112:   echo "Starting sendmail daemon (/usr/sbin/sendmail -bd -q15m)..."
113:   /usr/sbin/sendmail -bd -q15m
114: fi
```

実行形式のファイルが存在すればthen以降を実行する
起動中である旨のメッセージ
デーモンを起動する（いきなり/usr/sbinのコマンドを実行している）

rc.Mのnetatalkの部分である。デフォルトでコメントアウトされている。このようにコメントアウトしておけばいいわけだ。もちろんnetatalkを起動したければ#を外せばいい。あぁ簡単！

しかし、Slackware系では、デーモンの実行状態を確認したり、シェルスクリプトに/sbinや/usr/sbinのコマンドを指定したりするのに若干のスキルが要求される（大したものではないが）。このあたりがSlackwareユーザーのこだわりでもあり、カッチョイイところでもある。

以上のようにrc.d以下は、Red Hat系とSlackware系で異なる構成を持っている。どちらも一度わかってしまえば、難しいことはない。あとは、好みの問題だろう。

/etc/sysconfig (Red Hat系)

Red Hat系には/etc/sysconfigというディレクトリがある。ここには、主にinit.d以下のシェルスクリプトが参照する、ハードウェア定義情報や、デーモン起動時のオプションなどが、個別のファイルとして置かれている。

Slackwareでは、ここにあるファイルの役割の多くをrc.d以下のシェルスクリプトにマージしているため、このディレクトリは存在しない。

/etc/sysconfig以下のファイル

sysconfigにあるファイルとディレクトリを眺めてみよう。ファイルの内容を**表4**に示す。いろいろなファイルが置かれているので、すべてを詳しく解説することは避けるが、pcmciaを例にとり、具体的にどのようなファイルが置かれているのかを明らかにしておこう。

■コンフィグレーションの例

pcmciaは、Linuxをノートパソコンにインストールする際に必ずお世話になる。ファイルにはカードドライバに関する設定が記述されているのだ。**リスト6**に示す。

PCMCIAを利用する場合は、カードドライバを有効にするスイッチにyesを指定する。何も指定しないと無効になる。PCICにはPCMCIAのコントローラチップを指定する。もちろんドライバが用意されていなければならない。PCIC_OPTSには、コントローラに指定するオプションを列挙する。PCカードに割り当て可能なIRQ番号と、カードの抜き差しを検出するポーリング間隔を指定している。CORE_OPTSには、ドライバ自体に対するオプションを指定する。

このファイルと同様の設定は、Slackware系で言うと、/etc/rc.d/rc.pcmciaというファイルの中に記述するようになっている。

シェルの環境設定ファイル

シェルの環境設定ファイルとは、シェルやアプリケーションの設定を行うためのファイルだ。多くの場合、シェル変数や環境変数の設定、シェルそのものの振る舞いの設定、コマンドプロンプトの表示形式、コマンド検索パスなどを記述する。これらのファイルを変更することで、Linuxがより使いやすくなる。シェルの環境設定ファイル

▼リスト5　rc.Mファイルのnetatalk起動部分（Slackware系、抜粋）

```
56: # Start netatalk. (a file/print server for Macs using Appletalk)
57: #if [ -x /etc/rc.d/rc.atalk ]; then
58: #  /etc/rc.d/rc.atalk
59: #fi
```

行頭の#ですべてコメントアウトされている

▼リスト6　/etc/sysconfig/pcmcia（Red Hat系）

```
1: PCMCIA=yes
2: PCIC=i82365
3: PCIC_OPTS="irq_list=4,6,10,11 poll_interval=100"
4: CORE_OPTS=
```

PCMCIAドライバを有効にする
PCMCIAコントローラチップを指定
PCMCIAコントローラチップを指定カードに割り振るIRQと、カードの抜き差しを検出するためのポーリングの間隔
コアオプションを指定する

ファイル／ディレクトリ	内容
apmd	電源管理に関する設定情報（apmdへのオプション指定）
clock	時刻に関する定義
console/	コンソールに関する定義ファイル（キーマップ等）
hwconf	ハードウェア全般の定義（I/Oコントローラチップやビデオチップ等）
i18n	国情報、使用する言語の情報
init	init実行時の各種設定
keyboard	キーボードの型（日本語106型キー等）
mouse	マウスの種類（3ボタン、PS2マウス等）
network	ホスト名、IP Forward等
network-scripts/	ネットワークデバイス、ネットワークのダウン・再起動時の各種設定
pcmcia	PCMCIAドライバへのオプション情報
sendmail	Mailデーモンのキューイング時間等
soundcard	サウンドカードの設定情報

表4　sysconfig以下のファイルとディレクトリ

もまた、/etcディレクトリに置かれている。ディストリビューションや、システムにインストールするシェルの種類にもよるが、一般的には以下のファイルが見つかるはずだ。

```
profile
csh.cshrc
csh.login
```

また、システムによっては以下のファイルやディレクトリも見つかるかもしれない。

```
profile.d/*
bashrc
```

ここではこれらのファイルについて簡単に解説しておこう。

bashの環境設定ファイル

ユーザーがログインすると、/etc/passwdに記述されているログインシェルが起動する。Linuxでは、通常はログインシェルとしてbashが使われている。bashはユーザーの利用に合せて、いろいろと環境設定できるようになっている。bashが参照する環境設定ファイルには以下のものがある。

```
~/.bash_profile
~/.bashrc
~/.bash_logout
```

.bash_profileは、bashがログインシェルとして起動されたときに参照される。bashrcは、ログイン後に新たにbashを起動したときに参照される。.bash_logoutは、ログアウトするときに参照される。

bashは、Bourn Again SHellから命名されており、UNIXに古くからあるBourn Shell、すなわちshの多くの機能をサポートしている。そのためshからの移行もたやすく、.bash_profileは.profileでも代用可能だ（ふだんshを使っているユーザーは滅多にいない。むしろkshすなわちKorn Shellからの移行をたやすくしたはずだ）。シェルスクリプトの書式も踏襲されているため、そのまま使える。cshやtcshはログイン時に.loginというファイルを参照するが、この概念を取り入れて、.bash_profileの代わりに.bash_loginというファイルを作成しても同じく機能する（ただし、cshとbashはシェルスクリプトの書式が異なるのでそのまま使うことはできない）。

bashはログイン時に、まず.bash_profileを探し、なければ.bash_loginを、これもなければ.profileを探す。ユーザーのホームディレクトリにこれらの該当ファイルがまったくない場合、/etc/profileを参照する。

また、もし/etc/bashrcというファイルがあれば、それは.bashrcの雛形だ。これを自分のホームディレクトリにコピーして、さらに自分専用の環境を盛り込んでいけばいい。

ディレクトリ/etc/profile.d以下には、ユーザーログイン時に実行されるシェルスクリプトが収められている。このディレクトリ内のシェルスクリプトはprofile内から実行される。

bashの環境設定

もしも、自分のホームディレクトリに上記のファイルのいずれもない場合には、.bash_profileと.bashrcを作成しよう。ここでは、bashの環境設定ファイルの内容について簡単に触れる。シェル環境のカスタマイズで、最もポピュラーなのが、コマンド検索パスの変更と、プロンプト文字列の変更、そしてコマンドのエイリアス機能だろう。

■コマンド検索パス

まずは、コマンド検索パスからだ。コマンド検索パスにディレクトリを登録しておけば、シェルのコマンドライン入力の手間が省ける。登録したディレクトリまでのパスを省略することができるのだ。単にlsと入力したときに、/bin/lsが実行されるのは/binがコマンド検索パスに登録されているからにほかならない。

コマンド検索パスを登録するには、シェル変数PATHに検索させたいパスを代入すればいい。複数あれば、コロン（：）で区切って指定する。シェルは、先頭のディレクトリから順に検索していくことに注意しよう。同じ名前のコマンドが別のディレクトリに複数存在した場合、どれが実行されるかは、PATHへの登録順序次第となる。

```
PATH="/usr/local/bin:/usr/bin:/bin:
/usr/X11R6/bin:/usr/games"
```

■プロンプト文字列の変更

複数のディレクトリを行き来したり、複数のホストにログインしたりする場合、現在使っている端末がどのホストに接続されているか、カレントディレクトリがどこであるかをきちんと把握しておきたい。また、システム管理者の場合は、現在一般ユーザーでログインしているのか、スーパーユーザーでログインしているのかにも注意しなければならない。こんなときに便利なのが、コマンドプロンプト文字列の変更だ。ここに、現在のユーザー名、ホスト名、カレントディレクトリを表示させるといい。

bashでは、プロンプト文字列表示用に、プロンプト変数PS1からPS4までが用意されている。ただし、PS3とPS4はシェルプログラミングとデバッグ用なのでここでは割愛する。PS1は通常のプロンプトで、PS2はコマンドラインが完成していないのにEnterキーが押された場合にだけ表示される。ただし、コマンドラインが完成しているかどうかをシェルが検出する方法は限られている。PS2に設定したプロンプトにお目にかかれるのはクォーテーションの途中で改行した場合だ。たとえばPS1が「$」でPS2が「>」のとき、echoコマンドの途中で改行すると以下のようになる。

```
$ echo "This is
> test"
This is
test
$
```

さて、プロンプト文字列の設定をしよう。設定方法は簡単で、PS1に文字列を代入すればいい。

```
PS1="myprompt=> "
```

これで、「myprompt=> 」という文字列が表示される。では冒頭のように状態表示をさせるにはどうすればいいだろう。実は、プロンプト変数には通常の文字列以外に、いろいろな情報に置き換える特殊文字が用意されている。その主なものを**表5**に示す。

これを踏まえて、たとえばPS1を以下のように設定する。

```
PS1="\u@\h:\w\$ "
```

ユーザーfoo-nがホストmyhostの/etcで作業中ならば、

```
foo-n@myhost:/etc$
```

という表示になるのだ。さらにここでsuを使ってスーパーユーザーになると、

```
root@myhost:/etc#
```

と表示が変わる。これでホストやディレクトリを間違える危険も減るだろう。

文字	置き換え内容
\d	今日の日付を表示。「Fri Aug 25」（8月25日金曜日の場合）
\H	ホスト名。「myhost.subdomain.domain.com」
\h	最初の.までのホスト名。「myhost」（サブドメイン、ドメインを省略）
\n	改行
\s	シェルの名前（実行されているシェルの起動時の名前）
\T	現在の時刻を12時間表示の時：分：秒で表示する。「02:30:10」
\t	現在の時刻を24時間表示の時：分：秒で表示する。「14:30:10」
\@	現在の時刻をAM／PM表示付きの時：分で表示する。「02：30pm」
\u	現在のユーザー名を表示
\v	bashのバージョン番号を表示
\V	bashのリリース番号を表示
\w	カレントディレクトリを表示。「/usr/local/bin」（ただしホームディレクトリは~で表示）
\W	カレントディレクトリのベース名を表示。「bin」（/usr/local/binの場合）
\#	現在のコマンドのコマンド番号を表示
\!	現在のコマンドのヒストリ番号を表示
\$	rootなら#を、それ以外なら$を表示
\xxx	文字を文字コードで指定（xxxは8進数）
\\	「\」を表示

表5　bashのプロンプト変数に利用できる特殊文字

■エイリアス

コマンド文字列に別名を付けるのがエイリアス機能だ。よく使う長いコマンドライン文字列を短縮登録できるのだ。

たとえばWindowsパーティションと頻繁に行き来をする場合、

```
cd '/mnt/windows/My Documents'
```

などという長いコマンドラインを何度も打つのは煩わしい。そこで、

```
alias cdwin='cd /mnt/windows/My Documents'
```

としておけば、「cdwin」と入力するだけで、コマンドラインを置き換えてくれるのだ。「=」の前後にスペースを空けてはいけない。

当然だがコマンドラインはシェルが解釈した後にシステムに渡される。したがって、aliasにコマンド名を登録してオプションまで含めてしまうこともできる。たとえば、rmコマンドに確認のオプション-iを付けたい場合、

```
alias rm='rm -i'
```

と指定しておくといい。そうしておけば、単に「rm」と実行しても、-iオプションが付いて、削除時の確認がされるのだ。

aliasを複数登録する場合は、1行1行並べて記述するようにする。

```
alias rm='rm -i'
alias sl='ls'
  :
```

aliasの設定は、ログインシェルとは別にbashを起動した場合、そちらのシ

ェルには引き継がれない。したがって、.bashrcに設定しておくべきである。そうしておけば、起動したシェルで常にaliasが有効になる。

しかし、それではログインシェルで使えないという問題が起きてくる。では、.bash_profileにも同様の設定をしなければいけないのだろうか？　bashの環境設定ファイルはsourceコマンドを使っていつでも読み込ませることができる。ユーザーがコマンドラインから実行するなら以下のようする。

```
$ source .bashrc
```

これで.bashrcの設定を直ちにbashに反映させることができるのだ。これを.bash_profileの最後に記述しておけば、ログイン時にも.bashrcの設定が有効になる。

環境をカスタマイズすれば、どんどん使いやすくなるのがbashだ。これを機に、いろいろ設定してみて欲しい。

cshとtcshの環境設定ファイル

chs.cshrcとcsh.loginは、cshとtcshで参照される環境設定ファイルだ。最近のディストリビューションでは/bin/cshは/bin/tcshにリンクされているので、両者は一緒だったりする。したがってtcshとしておこう。tcshは、スクリプトの書式がCに似ていることもあり、UNIXでは根強い人気を持つシェルだ。しかし、Linuxユーザーにはあまり馴染みがないかもしれない。

tcshもログイン時とシェル実行時に、それぞれ環境設定ファイルを参照する。通常は、以下のファイルを参照する。

```
/etc/csh.login
~/.login
/etc/csh.cshrc
~/.tcshrc
~/.cshrc
```

tcshはスクリプトの書式がshと異なるため、bashのように.profileを使いまわすことはできない。それぞれのファイルの参照タイミングは次のようになる。まず、ユーザーログイン時には/etc/csh.loginと/etc/csh.cshrc、~/.ogin、~/.tcshrc、~/.cshrcが、シェル実行時には/etc/csh.cshrcと~/.tcshrc、~/.cshrcが参照される。

したがって、ユーザーが自分の環境を設定したければ、.loginと.tcshrcを用意すればいい。bashのところで紹介した機能をtcshでも設定してみよう。

コマンド検索パスの設定は以下のようになる。複数のディレクトリはスペースで区切る。

```
set path = ( /usr/local/bin /usr/bin
/bin /usr/X11R6/bin /usr/games )
```

プロンプト文字列は変数promptに文字列を代入する。やはり特殊文字が用意されている。主なものは**表6**に示すので参照してほしい。これらを使うことによってプロンプトを便利に使うことができる。特殊文字を使った例を示しておこう。

文字	内容
%#	rootなら#を、それ以外なら>を表示
%/	カレントディレクトリを表示。「bin」（/usr/local/binの場合）
%~	%/と同じだが、ホームディレクトリは~に置き換える
%h、%!	コマンドヒストリ番号を表示
%M	ホスト名。「myhost.subdomain.domain.com」
%m	最初の.までのホスト名。「myhost」（サブドメイン、ドメインを省略）
%T	現在の時刻をAM／PM付きの時：分で表示。「02：30pm」
%t、%@	現在の時刻を24時間制で表示。「14：30」
%n	ユーザー名を表示する
%D	今日の日付を表示。「07」（2000年9月7日の場合）
%W	今月の月を綴りで表示。「Sep」（2000年9月7日の場合）
%w	今月の月を数字で表示。「09」（2000年9月7日の場合）
%y	今年の年を下2桁で表示。「00」（2000年9月7日の場合）
%Y	今年の年を4桁で表示。「2000」（2000年9月7日の場合）
%l	シェルが起動している端末番号（tty??）を表示
%B	太字表示の開始と終了（%B〜%Bというように文字列を囲む）
%U	下線表示の開始と終了（%U〜%Uというよに文字列を囲む）
%%	%を表示

表6　tcshのプロンプト変数に利用できる特殊文字

```
set prompt = "%n@%m:%~%# "
```

結果は一般ユーザーのときのプロンプト文字列がbashの「$」と異なり「>」になるが、あとは同じである。

エイリアスも書式が若干異なるが機能は一緒だ。tcshでは「=」を使わずスペースで区切る。以下のように記述すればいい。rmコマンドの例ならば、

```
alias rm 'rm -i'
```

と設定する。

tcshは、デフォルトのシェルではないし、システムに用意されている（たとえば/etc/rc.d以下の）スクリプトファイルとも書式が異なるため、よほどの理由がなければ、わざわざ使うユーザーはいないかもしれない。

しかし、古くからUNIXを利用しているユーザーには、馴染みのあるシェル環境である。

ハードウェアに関する設定

fstab
mtab

fstabは、ファイルシステムを構成するパーティションやリムーバブルディスクなどの、ディレクトリツリー上におけるマウントポイントが記されているファイルだ。ファイルシステムの設計図といえる。またmtabは、現在マウントされているファイルシステムの一覧が記されている。マウントには、mountコマンドが用いられる。

起動時には、fstabに記されている順にパーティションがマウントされ、1つのディレクトリツリーが作られていくので、fstabの先頭行は、必ず“／”（ルート）パーティションを記述する。

fstabの書式は、1項目につき6つのフィールドがある。1番めのフィールドには、デバイス名を指定する。NFSやSMBファイルシステムを用いて、リモートマシンのディスクをマウントする場合は、ホスト名などを指定する。

2番めのフィールドには、マウントポイントを指定する。スワップパーティションは、マウントされないので、“ none ”または“ swap ”と記述する。

3番めには、ファイルシステムの種類を記述する。Linuxはネイティブのext2ファイルシステム以外にも、多くのファイルシステムをサポートしており、ここで正しく設定することで自由に読み書きができる。

4番めには、各ファイルシステム固有のオプションを指定する。defaultsを指定すると、読み書き可能、非同期アクセス、一般ユーザーによるマウントの禁止などをまとめて選択したことになる。noautoは、明示的に指定しないかぎりマウントできないようにするもので、リムーバブルメディアのドライブには必要だろう。roは読み込み専用を表す。userは一般ユーザーによるマウントを許可するものだ。ほかにもさまざまなオプションが存在するが、一般的にはこの程度を知っていれば十分だ。詳細は、mountやfstabをmanコマンドで調べてほしい。

5番めには、dumpコマンドがそのファイルシステムをダンプする必要があるかどうかを指定する。0だとダンプする必要がないと判断されるので、ハードディスク上のパーティションには1、それ以外には0を指定しておけばいい。

最後の6番めには、fsckコマンドで整合性をチェックするかどうかを指定する。0を指定したパーティションは、チェックされない。1以上の整数を書いておけば、数字の小さい順にチェックが行われる。ルートパーティションは最初にチェックされる必要があるので、必ず1を指定する。最近のfsckは、並列処理ができるようになっているため、ルート以外のハードディスク上のパーティションにはすべて2を指定しておけば、複数のドライブを同時にチェックするなど適切に処理が行われる。リムーバブルメディアなどには0を指定しておけばいい。

conf.modules
modules.conf

Linuxは、ハードウェアのデバイスドライバや、ファイルシステム用ドライバなどを必要に応じてカーネルに組み込むことができる。この機能は、カーネルモジュールのオンデマンドローディングと呼ばれるものだ。RPM系のディストリビューションでは、modutilsというパッケージで提供されている、depmod、modprobe、insmodなどのコマンドによって実現されている。

conf.modules／modules.confは、depmod、modprobeコマンドの設定ファイルである。Red Hat、Kondara、LASER5、Vineでは、conf.modulesが、TurboLinuxではmodules.confという名前になっている。またカーネル2.4からはmodules.confがデフォルトになるようである。各ディストリビューションともカーネル2.4を採用するバージョンから、順次modules.confに変わっていくと思われる。

conf.modulesは、さまざまな設定が行えるが、実際にはaliasを指定して、使用しているハードウェア用のドライバを自動的に組み込むことが、一番多いだろう（**リスト8**）。

▼リスト7　fstab

	デバイス名	マウントポイント	ファイルシステム	オプション		
1:	/dev/hda5	/	ext2	defaults	1	1
2:	/dev/hda8	/home	ext2	defaults	1	2
3:	/dev/hda1	swap	swap	defaults	0	0
4:	/dev/fd0	/mnt/floppy	ext2	noauto,user	0	0
5:	/dev/cdrom	/mnt/cdrom	iso9660	noauto,ro,user	0	0
6:	proc	/proc	proc	defaults	0	0
7:	none	/dev/pts	devpts	mode=0622	0	0

▼リスト8　conf.modules ネットワーク、サウンドカードなどが指定されている

```
1: alias eth0 eepro100
2: alias parport_lowlevel parport_pc
3: alias sound-slot-0 esssolo1
```

シェル

文：編集部
Text : Linux magazine

GNOMEやKDEといったデスクトップ環境下でも、コンソールを開いてbashなどのシェルを使い、効率良く作業を進めるのがUNIXスタイルだ。広義のシェルに関するファイルを調べていこう。

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シェルによっては、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。DIR_COLORSの変更を反映させるには、DIR_COLORSを引数に指定して、dircolorsコマンドを実行すればよい。ただし、dircolorsコマンドは、カラー表示設定のコマンドを出力するだけなので、実際には以下のように入力する。

```
# eval `dircolors /etc/DIR_COLORS`
```

“ 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（画面1）。毎回オプションを付けるのはめんどうなので、シェル起動時に読み込まれるファイル（~/.bashrcなど）に、

```
alias ls='ls --color'
```

と指定しておこう。

なお、複数の人が使うマシンで、奇抜な配色にしてしまうと、ひんしゅくを買うおそれがある。その場合は、DIR_COLORSファイルを~/.dir_colorsにコピーし、編集後に、

```
#eval `dircolors ~/.dir_colors`
```

とするようにしよう。

bashrc

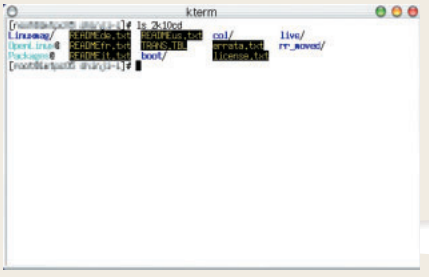
Linuxの標準シェルであるbash起動時に、全ユーザー共通の設定項目を記すためのファイルである。bashが自動的に読み込むわけではなく、各ユーザーの~/.bashrcの先頭で実行しているだけだ。

profileがログイン時のみ実行されるのに対し、~/.bashrcはシェルが起動するたびに実行される。

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である。

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



画面1
DIR_COLORSを書き換えて、lsを阪神タイガースファン専用（？）に設定する。

▼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

```
33: # string consists of one or more of the following numeric codes:
34: # Attribute codes:
35: # 00=none 01=bold 04=underscore 05=blink 07=reverse 08=concealed
36: # Text color codes:
37: # 30=black 31=red 32=green 33=yellow 34=blue 35=magenta 36=cyan 37=white
38: # Background color codes:
39: # 40=black 41=red 42=green 43=yellow 44=blue 45=magenta 46=cyan 47=white
40: NORMAL 00       # global default, although everything should be something.
41: FILE 00         # normal file
42: DIR 01;34       # directory
43: LINK 01;36      # symbolic link


41: FILE 33;40      # normal file
```

このように変更すると、画面1のような表示になる

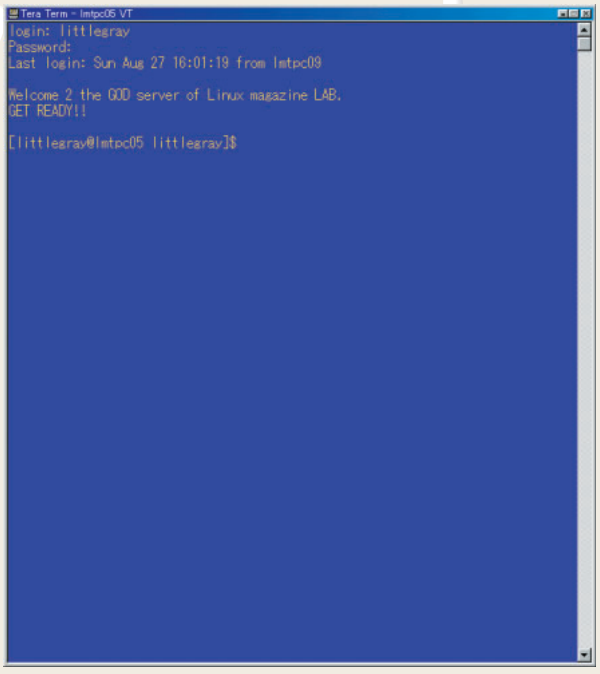
ーするか、~/.cshrc、~/.loginの先頭で実行するようにすれば、全ユーザー共通の環境を設定できる。

リスト2の例では、13～18行で新たに作られるファイルのデフォルトパーミッションを、「グループ名とユーザー名が同じで、ユーザーIDが14より大きい」場合にパーミッションが“ 775 ”、そうでなければ“ 755 ”に設定している。これは、1人が1グループに設定されている一般ユーザーは、本人とグループのメンバー（実際には本人しかいない）にフルアクセスを許可し、それ以外の場合には、本人だけにフルアクセスを許すように設定されているということだ。

また28行めからは、profile.d/以下のスクリプトを実行している。profile.d/以下には、各種のパッケージ用の初期設定スクリプトがあり、その中の拡張子が“ csh ”であるものを順次実行するようになっている。

Linuxの標準シェルは、sh系であるbashだが、このようにprofile.d/以下にcsh用の初期設定スクリプトを集めることによって、ログインシェルがsh／csh系を問わず環境設定を行えるようになっている。

なお、これらシェルの設定ファイルの記述方法は、今回の特集の範囲を越えるものなので、必要に応じて文献などを参照してほしい。



画面2
ログインに成功すると、motdの内容が表示される。

▼リスト2　csh.login ~/.loginのテンプレート

```
13  [ `id -gn` = `id -un` -a `id -u` -gt 14 ]
14  if $status then
15          umask 022
16  else
17          umask 002
18  endif


28  test -d /etc/profile.d
29  if ($status == 0) then
30          set nonomatch
31          foreach i ( /etc/profile.d/*.csh )
32                  test -f $i
33                  if ($status == 0) then
34                          source $i
35                  endif
36          end
37          unset i nonomatch
38  endif
```

デフォルトパーミッションの設定

profile.d/ ディレクトリ内のスクリプトを実行

issue
issue.net
motd

ローカルまたはリモートからログインする前後に表示されるメッセージは、これらのファイルで指定する。

issueは、ローカルから、issue.netは、telnetでリモートからログインする際にプロンプトの前に表示されるメッセージが含まれている。通常は、ディストリビューション名、カーネルバージョン、ホスト名などが書かれている。

motdは、“ Message Of The Day ”の略語で、正しいユーザー名とパスワードを入力して、ログインに成功した直後に表示されるメッセージが含まれている。Red Hat系ディストリビューションでは、デフォルトでは何も書かれていない。**画面2**のようにウェルカムメッセージを表示するのに利用してもいいし、「メンテナンスのため、××日にシステムが停止する」といったユーザー全員に伝えるべき情報を表示するのにも適している。わざわざメールで知らせるよりも、motdを利用するほうがスマートなやり方といえる。

kon.cfg

日本語コンソールエミュレータであ

るkonの設定ファイルだ。ディスプレイの表示範囲など、ハードウェアに関する設定が書かれているので、ユーザーの手で書き換える必要はない。

profile
profile.d/

profileは、Linuxの標準シェルであるbashがログイン時に実行するファイルで、システム全体に影響がある環境設定を行うファイルである。同じRed Hat系ディストリビューションでも、微妙に差があるが、設定される項目そのものは、ほぼ一緒だ。

リスト3の9行めからは、新たに作られるファイルのデフォルトパーミッションを指定している。「グループ名とユーザー名が同じで、ユーザーIDが14より大きい」場合にパーミッションが“775”、そうでなければ“755”に設定される。これは、1人が1グループに設定されている一般ユーザーは、本人とグループのメンバー（実際には本人しかいない）にフルアクセスを許可し、それ以外の場合には、本人だけにフルアクセスを許すように設定されているということだ。

また26行めでは、各種の環境変数を設定しており、28行めからはprofile.d/以下のスクリプトを実行している。profile.d/以下には、各種のパッケージ用の初期設定スクリプトがあり、profileから起動されるようになっている。たとえばlang.sh（リスト4）は、先頭で/etc/sysconfig/i18nというファイルを呼び出している。i18nの内容は、

```
LANG="ja_JP.eucJP"
```

であり、ここで環境変数“LANG”を初期設定している（Red Hat Linux 6.2）。なお、TurboLinuxには、/etc/sysconfig/i18nがないが、turbocentro-cfg.xmlで同様の設定を行っている。

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である。デフォルトは、tty1からtty8までが記されており、rootユーザーのログインは、ローカルのコンソールからだけに制限されている（**リスト5**）。

以前のバージョンでは、telnetでログインする際のターミナル名としてttyp0、ttyp1……、が用いられていたので、securettyにこれらのターミナル名を追加することで、rootユーザーでもリモートログイン可能にすることができた。

しかし最近のバージョンでは、pts/0、pts/1……というようにターミナルの名称が変更されており、これらをsecurettyに加えてもrootユーザーのtelnetはできないようになっている。もっとも一般ユーザーでログインした後に、suコマンドを使えば同じことができるので、問題にはならない。これは、rootがリモートからログインするというセキュリティ意識の薄い行為をできないようにする、フェイルセーフ的な措置といえる。

shells

ログインシェルとして指定可能なシェルがフルパスで記されている。chshコマンドでログインシェルを変更する時に参照される。

▼リスト3　profile 全ユーザーに共通の項目を設定する

```
 9: if [ `id -gn` = `id -un` -a `id -u` -gt 14 ]; then
10:         umask 002
11: else                                          デフォルトパーミッションの設定
12:         umask 022
13: fi


26: export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC   環境変数の設定
27:
28: for i in /etc/profile.d/*.sh ; do
29:         if [ -x $i ]; then
30:                 . $i
31:         fi
32: done
```

▼リスト4　profile.d/lang.sh “LANG”環境変数の設定

```
 3: if [ -f /etc/sysconfig/i18n ]; then
 4:     . /etc/sysconfig/i18n
 5:     if [ -n "$LANG" ] ; then
 6:        [ "$LANG" = "C" ] && LANG="en_US"
 7:        export LANG
 8:     else
 9:       unset LANG
10:     fi
```

▼リスト5　securetty rootのログインをローカルのコンソールに限定

```
1: tty1
2: tty2
3: tty3
4: tty4
5: tty5
6: tty6
7: tty7
8: tty8
```

システム設定

文：編集部
Text : Linux magazine

このセクションでは、システム設定に関するファイルを解説していく。もっとも、システム設定という言葉は非常に範囲が広く、/etcディレクトリすべてのファイルに対してシステム設定と言うこともできるだろう。

しかし、前のセクションでシステム起動とシェルまでを解説しているので、ここではシステム起動後に関連するファイルをのみを中心に集めている。なお、デーモンに関しては別のセクションとしてページを割いている。

画面表示関連

screenrc

VT100／ANSI端末エミュレーション機能を持つ画面管理プログラム、screenの設定ファイルの雛形である。通常、ユーザーのホームディレクトリに.screenrcとしてカスタマイズしたファイルを用意する。screenはバックスクロールや、ウィンドウ間でのカット＆ペーストなど、豊富な機能を持つが、日本語表示に対応していないので日本ではあまり使われていないし、そんなプログラムが存在していたことすら知らない管理者も多いだろう。

TextConfig

コンソール画面の解像度や、フォント、カーソル形状などを変更するSVGATextModeコマンドが参照する設定ファイルだ。MS-DOSの時代に使われていた、Hi-TextやV-Textと同等の機能を実現する。ただし、konなどの漢字コンソールエミュレータには影響しないため、日本語環境では使うことができない。konで同様の設定を行うファイルは、「kon.cfg」だ。

X11

X Window Systemの各種設定ファイルが置かれている。

また、X Window Systemの設定ファイルをはじめ、グラフィカルログイン画面の設定やログインの認証方法などが記述されたファイルが置かれている。多くのウィンドウマネージャの設定ファイルなども保存されている。

ハードウェアの設定

adjtime

ハードウェアクロックの取得と設定を行うhwclockが使うファイルで、時刻をより正確に合わせるためのパラメータが記述されている（**リスト1**）。ハードウェアクロックは不正確であるため、誤差を少なくするために使用されるものだ。1行目には3つのパラメータが記述されており、順に次のような意味を持っている。

- **1日あたりの時刻のズレ（浮動小数点10進）**
- **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）**

▼リスト1　/etc/adjtimeファイルの内容

```
1: 7.724213 967395300 0.000000
2: 967395300
3: LOCAL
```

最後のゼロは、clockコマンドとの互換性のために用意されているもので、以前は時刻合わせを行った際に切り捨てられた秒数を格納していた。

2行目は1行目の2つめのパラメータと同じく、1970年1月1日　0時0分0秒UTC（Universal Time Coordinated：世界協定時）からの経過秒数が格納されている。

3行目は、ハードウェアクロックがUTC（UTC）かローカルタイム（LOCAL）かを示す文字列が格納される。

これらに格納される値は、hwclockコマンドが設定するため、システム管理者が設定する必要はない。

もっとも、adjtimeを使って時刻を正確に保つのであれば、タイムサーバから正確な時刻を取得するのが確実だ。

conf.linuxconf

Linux設定ツールである、linuxconfが参照する設定ファイル。linuxconfのインストール済みのモジュールなどが記述されているが、特に設定する必要はない。

fb.modes

ディスプレイフレームバッファのデータベース。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フレームバッファのモードが記述されている。各解像度における、ドット数や周波数などが記述されている。fb.modesはX Window Systemが参照するものではないので、通常のユーザーにはあまり関わりがないだろう。なお、Red Hat Linuxのみ存在する。

fdprm

フロッピーディスクのパラメータが定義されたファイル。パラメータはsetfdprmコマンドが参照する。特別なドライブを使用しているのでなければ、通常変更する必要はない。

filesystems

mountコマンドでファイルシステムの形式を指定しなかった場合や、ファイルシステムのタイプにautoを指定した場合で、ファイルシステムが自動認識できなかった場合に、このファイルに記述されているファイルシステムの形式でマウントしようと試みる。パラメータの指定方法は、単にファイルシステムが羅列されているだけだが（**リスト2**）、「nodev」を付けたファイルシステムの形式は除外される。通常使われるファイルシステムは自動認識されるため、filesystemsに新たにファイルシステムを加える必要はほとんどないだろう。

gpm-root.conf

標準マウスハンドラである、gpm-rootの設定ファイルだ。コンソール画面でのマウスの機能を設定する。もっとも、X Window Systemが一般的に使われるような環境では設定する必要はない。

isapnp.gone

プラグ＆プレイISAカード設定ツールである、isapnptoolsで使用されるファイル。isapnp.goneには、プラグ＆プレイで認識できないリソースを指定する。isapnptoolsでは、指定したリソースを避けてリソースを割り当てる。isapnp.goneに記述するパラメータは、COMポートやパラレルポートなどのマザーボードの標準的なリソースや、プラグ＆プレイに対応していないISAカードのリソースなどを記述する。しかし、多くの場合、isapnp.goneに記述する必要はないだろう。

pcmcia

pcmcia（PCカード）用の設定ファイルが置かれているディレクトリだ。このディレクトリには、configというファイルがあり、ここに各PCカードとドライバの対応が記述されている。拡張子optsファイルには、PCカードのコンフィグレーションを記述する。PCカードのコンフィグレーションについては、PCカードごとに異なるのでここでは割愛する。

ユーザー／グループに関する設定

default/

useraddコマンドで作成するユーザーアカウントについて、デフォルトの設定が記述されたuseraddファイルが格納されている。useraddファイルは**リスト3**のような内容になっている。

「GROUP」は、ユーザーが標準で属するグループIDを指定する。ただし、useraddコマンドはユーザーIDとグループIDを同じにするようにRed Hatが仕様を変更してるため、この値はRed Hat系ディストリビューションでは、「-n」オプションを付けた場合のみ有効となる。

「HOME」は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。ユーザーのホームページは通常、/homeの下に作成するので変更する必要はない。

「INACTIVE」はパスワードの有効期限後にアカウントを無効にする期限を1日単位で指定する。「−1」はアカウントを無効にしない、「0」はパスワードの有効期限とともにアカウントを無効にする指定だ。

「EXPIRE」はパスワードの有効期限を、INACTIVEと同様に1日単位で指定する。

「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。通常は/bin/bashのままで構わない。

パスワードの有効期限とパスワードの有効期限後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。

「SKEL」はユーザーのホームディレクトリにコピーするスケルトンファイルが格納されたディレクトリを指定する。通常は/etc/skelディレクトリに格納されているファイルがユーザーのホームディレクトリにコピーされる。これらのディレクトリには、「.bash_profile」や、「.Xdefaults」などが含まれている。

group
group-

groupファイルは、グループを定義しているファイルだ。グループはファイルの読み書き属性や、実行属性で利用されるものだ。各フィールドは**リスト4**のようになっている。

ユーザーリストに列記されているユーザーが、そのグループに属している

▼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
```

▼リスト3　/etc/default/useaddファイルの内容

```
1: GROUP=100
2: HOME=/home
3: INACTIVE=-1
4: EXPIRE=
5: SHELL=/bin/bash
6: SKEL=/etc/skel
```

ことになる。現在ではシャドウパスワードが一般的に使われているため、暗号化されたパスワードフィールドは、暗号化されていることを表す「x」のみが格納されている。

group-ファイルは、groupファイルのバックアップで、修正前の内容が保存されている。

グループパスワードは、newgrpでグループIDを変更する場合に必要となる。

gshadow
gshadow-

gshadowファイルは、グループパスワードの暗号化されたファイルで、**リスト5**のような形式になっている。

このファイルはセキュリティのために管理者以外は参照できないようにしておく。

pam.d/
security/

pam.d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）の設定ファイルが置かれてる。最近のディストリビューションの多くは、ユーザーの認証やサービスへのアクセス認証にPAMを使っている。PAMはLinuxだけのものではなく、多くのUNIXベースのシステムで使われている。PAMを使うメリットは、モジュール化されているために、認証手順が完全にカプセル化されることだ。これにより、プログラマーは簡単な手順で安全な認証手続きを実装することができる。loginやsuコマンドもPAMを使って認証が行われている。

pam.dディレクトリにある設定ファイルには、認証に必要なモジュールと手順が記述されている（**リスト6**）。ここに記載された手順とモジュールによって認証が行われることになる。

PAMは非常に高機能で、設定ファイルを変更することで、ユーザーがログインできる時間帯を指定したり、ディスクの容量制限を行うquotaと同様の機能が実現できるが、不用意に変更してしまうと、ログインできなくなったり、セキュリティホールができてしまう可能性があるので十分注意する必要がある。PAMについてすべての情報を提供することは不可能なので、詳しい情報は、ドキュメントパッケージを参照してほしい。

securityディレクトリには、PAMで使われるモジュールが格納されている。

passwd
passwd-

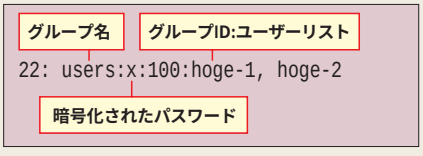
passwdには、ユーザーごとに、ユーザー名（ログイン名）、暗号化されたパスワード、ユーザーID、グループID、フルネーム（またはコメント）、ホームディレクトリ、標準シェルが順に書き込まれている（**リスト7**）。現在では、groupファイルと同様にシャドウパスワードが一般的に使われているため、暗号化されたパスワード欄には暗号化を表す「x」のみが格納されている。このため、現在はユーザー情報のみが格納されているファイルだといえる。このファイルはテキストファイルで構成されているため、テキストエディタで編集することもできるが、不用意に変更してしまうと、ユーザーがログインできなくなってしまうなどの問題が発生するため、注意が必要だ。なるべくpasswdコマンドを使うようにしたい。直接編集する場合は、vipwコマンドを使用する。

passwd-ファイルはgroup-ファイルと同じくバックアップだ。

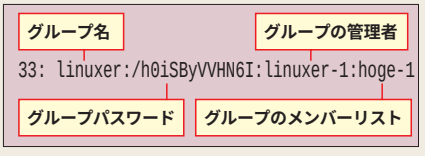
pwdb.conf

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。libpwdbは、アプリケーションが容易にユーザー情報を取得するためのライブラリである。PAMは、そのライブラリを使用してユーザー情報を取得している。pwdb.confでは、ユーザーやグループ情報を取得するためのデータベ

▼リスト4　/etc/groupファイルの内容



▼リスト5　/etc/gshadowファイルの内容



▼リスト6　/etc/pam.d/loginファイルの内容

```
#%PAM-1.0
auth       required  /lib/security/pam_securetty.so
auth       required  /lib/security/pam_pwdb.so shadow nullok
auth       required  /lib/security/pam_nologin.so
account    required  /lib/security/pam_pwdb.so
password   required  /lib/security/pam_cracklib.so
password   required  /lib/security/pam_pwdb.so nullok use_authtok md5 shadow
session    required  /lib/security/pam_pwdb.so
session    optional  /lib/security/pam_console.so
```

▼リスト7　/etc/passwdファイルの内容

```
45: hogehoge:x:515:100:Hoge Linao:/home/hogehoge:/bin/bash
```

ースの集合を記述する。しかし、特別な場合を除いてこの設定ファイルを変更する必要はない。

shadow
shadow-

shadowは、暗号化されたパスワードに関する情報が格納されている。以前のUNIXシステムでは、/etc/passwdに暗号化されたパスワードが記述されていたが、このファイルはシステムにログインできるユーザーであれば誰でも参照できるため、パスワードの安全性に問題があった。そこで、rootユーザーだけが参照できるshadowにパスワード情報を格納できるように拡張された。

また、アカウントの有効期限などが設定できるようにも拡張されている。

shadowの内容は、「:（コロン）」に区切られた形式で、**表1**のような順序になっている。

このファイルはセキュリティのために、rootユーザー以外は参照できないようにする。また、記述を間違えると、ユーザーがログインできなくなってしまう可能性もあるため、ファイルの内容に関して、エディタなどで直接編集すべきではない。passwdコマンドを使うようにしたい。

shadow-ファイルはご想像通りバックアップだ。

カラム	内容
1	ログイン名
2	暗号化されたパスワード
3	前回パスワードを変更した日（1970年1月1日 UTCからの経過日）
5	パスワードが変更可能になるまでの日数
6	パスワードの変更期限までの日数
7	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
8	パスワードの有効期限が過ぎてから、アカウントが無効化されるまでの日数
9	アカウントが使用不可になるまでの日数（1970年1月1日 UTCからの経過日数）

表1　/etc/shadowファイルの書式

skel/

useraddでも触れたが、ユーザーアカウントを作成した際に、ユーザーのホームディレクトリ以下に自動的にコピーされるファイルが格納されている。.bash_profileなどをシステムに合わせて変更しておき、このディレクトリにコピーしておくといいだろう。

その他

ld.so.cache
ld.so.conf

これらのファイルは動的リンク（ダイナミックリンク）されたライブラリを読み込むために使われるファイルである。Linuxのバイナリファイルによっては、実行時に動的ライブラリを読み込む必要がある。動的リンクされたプログラムは、ライブラリを内部に持つプログラムに比べて、サイズを小さくすることができる。

ld.so.confは、ダイナミックライブラリを検索するディレクトリを記述している。通常は、「/usr/lib/sconv」「/usr/X11R6/lib」などが含まれている。このファイルは、特殊なライブラリを必要としないかぎり変更する必要はない。

ls.so.cacheは、ライブラリのキャッシュファイルだ。このファイルはバイナリファイルであるため、編集することはできない。

HOSTNAME

マシン固有のホスト名がFQDN（Fully Qualified Domain Name）形式で格納されている。たとえば、「hoge@ascii.co.jp」といったものだ。このファイルはLinuxの起動時に毎回自動的に作成されるものであるため、管理者が変更する必要はない。

rmt

rmtは、/sbin/rmtへのシンボリックリンクになっている。rmtはリモート磁気テーププロトコルモジュールで、rdumpコマンドなどで、リモートバックアップが実行された際に呼び出される。

rpc

リモートプロシージャで使用されるプログラムのサーバの名前と番号と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システムにアクセスするためのもので、分散コンピューティング環境で使用されるシステムだ。

Column

/etcダンジョンに登場する隠れキャラ！？

/etcを探検していても、ふだんはなかなかお目にかかれないファイルがある。一時的に作成されるファイルがそれだ。たとえば、shutdownコマンドは指定時刻の5分前（5分以内の指定なら即時）に/etc/nologinというファイルを作成する。内容は、間もなくシャットダウンするのでログインできないという旨のメッセージだ。このファイルがあると、ログインしようとしたユーザーの端末にこのメッセージが表示され、root以外のユーザーは新規にログインできなくなるのだ。

この/etc/nologinは管理者が自分で作成することもできる。何かの理由で他のユーザーをログインさせたくない場合には、ここに一般ユーザーへの説明を書き込んでおけばいい。メンテナンスをする場合に便利だ。もちろん、自分で作成した場合は、あとで消しておくのを忘れずに。でないと一般ユーザーは永遠にログインできなくなってしまう。

実は、この仕組みも本文中で解説しているPAMが行っているのだ。

デーモン

文：編集部
Text : Linux magazine

UNIX系のOSでは、多数のデーモンプロセスが動作しており、さまざまな機能を実現している。もっとも、その大部分はネットワーク系のサービスを提供するものであり、それらに関してはネットワークの項目を参照していただきたい。ここでは、ネットワークサービス以外のデーモンに関係するファイルを説明する。

amd.conf
amd.net

これらは、ファイルシステムのオートマウントを実現するamdデーモンの設定ファイルだ。今回調査したRed Hat系ディストリビューションでamdがインストールされるのは、Red Hat Linux 6.2とLASER5 Linux 6.2だけだ。それ以外のディストリビューションでは、代わりに、同等の機能を持ったautomountが用意されている。

anacrontab

anacrontabは、Red Hat Linux 6.2とLASER5 Linux 6.2に用意されているanacronというコマンドの設定ファイルだ。名前から想像がつくように、cronに類似したコマンドだ。連続運転していないマシン用のcronと考えればいいだろう。

anacrontabの書式は、繰り返しの周期（日単位）、実行時の遅延時間（分単位）、名前、コマンドの4つをスペースで区切って、1行に書くようになっている。**リスト1**の例では、cron用の設定を流用して、実行頻度が1日、1週間、1カ月のコマンドを同じ頻度で繰り返すように設定している。

anacronコマンドは、前回に起動した日時を記録しており、今回の起動時間との差を取った結果をanacrontabに記されている繰り返しの周期と比較する。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（分）だけ待ったのち、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、

```
# run-parts /etc/cron.daily
```

が実行される。run-partsは、ディレクトリ以下のスクリプトを順次実行するコマンドだ。

anacronコマンドは、マシンの起動時に実行するようにしておくといいだろう。

auto.master
auto.misc

ファイルシステムのオートマウントを実現するautomountデーモンの設定ファイルだ。

直接コマンドラインからautomountを起動することもできるが、ほかのデーモンプロセスと同様の制御スクリプトが、/etc/rc.d/init.d/autofsとして用意されているので、これを利用するか、ntsysvコマンドやturboservice（TurboLinux）を用いて起動させればよい。

auto.masterは、マウントポイント（/misc）、マップファイルの名前（auto.misc）を記述し、オプションで自動アンマウントまでの時間などを指

▼リスト1　anacrontab cronの設定をそのまま利用している

```
 1: # /etc/anacrontab: configuration file for anacron
 2:
 3: # See anacron(8) and anacrontab(5) for details.
 4:
 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、auto.misc 自動マウント用の設定を行う

```
 1: # $Id: auto.master,v 1.2 1997/10/06 21:52:03 hpa Exp $
 2: # Sample auto.master file
 3: # Format of this file:
 4: # mountpoint map options
 5: # For details of the format look at autofs(8).
 6: /misc   /etc/auto.misc  --timeout 60


 1: # $Id: auto.misc,v 1.2 1997/10/06 21:52:04 hpa Exp $
 2: # This is an automounter map and it has the following format
 3: # key [ -mount-options-separated-by-comma ] location
 4: # Details may be found in the autofs(5) manpage
 5:
 6: cdrom          -fstype=iso9660,ro      :/dev/cdrom
 7: floppy         -fstype=auto            :/dev/fd0
```

定できる。**リスト2**の例では、automountでマウントするファイルシステムは、/misc以下に配置され、デバイスとしては、/etc/auto.miscに指定したCD-ROMとフロッピーディスクが利用できる。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されるようになっている。

auto.miscは、自動マウントしたいデバイスのキー、ファイルタイプなどのオプション、そしてデバイスファイル名を1行ずつ記す。それぞれのデバイスは、auto.masterで指定したマウントポイント下の、キーサブディレクトリとしてアクセスできる。

リスト2を例にとれば、/dev/cdrom（通常の構成では/dev/hdcへのリンクになっていることが多い）で表されるドライブにCD-ROMを入れ、コマンドラインから、

```
$ ls /misc/cdrom
```

とすることで、自動的にCD-ROMがマウントされ、リストが表示される。いちいちmount／umountをする必要がないので、なかなか便利に使える。

そのほか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のパーティションもNFSやSMBファイルシステムを利用して、自動マウントさせることが可能であり、工夫しだいでいろいろな使い方が考えられる。

at.allow
at.deny

UNIX系OSには、時間を指定してプログラムを実行するためのatコマンドがある。atはさまざま方法で時間の指定が可能だ。

at.allowとat.denyは、atを利用できるユーザーを指定するための設定ファイルだ。atは、以下の手順でユーザーのパーミッションを確認する。

(1) /etc/at.allowがあれば、そこに書かれているユーザーだけがatを利用可能
(2) /etc/at.allowがなくて/etc/at.denyがあれば、at.denyに書かれていないユーザーだけがatを利用可能
(3) どちらもなければ、スーパーユーザのみが利用可能

空の/etc/at.denyだけがあれば、(2)の規則により、すべてのユーザーがatを利用できる。今回調べたディストリビューションではこの状態がデフォルトになっており、誰でもatが利用できる。ユーザーによって利用を制限したければ、上記の規則にしたがって、at.allowかat.denyを作成しよう。

cron.*/
crontab

UNIX系システムで、定期的に同じプログラムを何回も実行したい時には、crontabコマンドを利用するといい。cron.d、cron.hourly、cron.daily、cron.weekly、cron.monthlyの各ディレクトリ内のファイルとcrontabファイルは、このcrontabコマンドに関連する設定ファイルだ。

crontabファイルの書式は、日時の指定、ユーザー、コマンドをスペースで区切って、1行に書くようになっている。日時は、分／時／日／月／曜日の5つをスペースで区切って順番に並べていく方法で指定する。具体的な数値で指定する以外に、“,”（カンマ）で区切って複数の数字を並べたり、“／”で増分値を指定することもできる。また「任意」を表す“＊”も使用できる。

以下、デフォルトのcrontabファイル（**リスト3**）を実例として見てみよう。7行めは、「時」以降がすべて“＊”になっているので、毎時1分にroot権限で/etc/cron.hourlyを引数にしてrun-partsコマンドを実行することを、また8行めは、毎日4時2分に/etc/cron.dailyコマンドを引数にしてrun-partsコマンドを実行することを表している。同様に9、10行めもそれぞれ毎週、毎月のコマンド実行を指定していることがわかる。同様に、月に2回の頻度で実行したいコマンドがあるなら、11行めを追加し、/etc/bi-weeklyディレクトリを作って、その中に実行したいコマンドを置けばいい。

繰り返し作業の周期は、たいてい「毎週」や「毎日」なので、これらのディレクトリにコマンドを置くほうが、crontabファイルに直接記述するよりも合理的だ。

▼リスト3　crontabファイル　11行めは、あとから加えたもの

```
 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
11: 32 4 1,15 * * root run-parts /etc/cron.bi-weekly
```

月2回実行するコマンドは、ここに入れる

ネットワーク関連

文：編集部
Text : Linux magazine

Linuxはネットワークとの親和性が高いOSだ。/etcディレクトリには、ネットワーク設定に関する数多くのファイルが収められている。

ひとくちにネットワークの設定といっても、Linuxのシステムがネットワークを扱うための設定や、各種ネットワークデーモンの設定など、その種類は多岐に渡る。このセクションではそれらについて、ジャンルごとに内容を見ていくことにしよう。

Linuxシステムのネットワーク設定ファイル

hosts

IPアドレスとホスト名の変換にはDNS（Domain Name System）というしくみが使われることが多いのはみなさんもご存じのとおりだ。しかし、小規模なLANでは、DNSサーバを設置しないことも多い。このような場合は、hostsファイルにIPアドレスとホスト名の対応を列挙する（**リスト1**）。

hostsファイルを利用した名前解決を行うには、後述するhost.confやnsswitch.confでDNSを参照する前にhostsファイルを参照するように設定する必要がある。

hostsファイルによる名前解決の欠点は、すべてのマシンが最新情報を記述したhostsファイルを持たなければならないことだ。すなわち、ホスト名やIPアドレスの変更があった場合には、すべてのマシンのhostsファイルを更新しなければならない。ある程度の台数がつながったLANでは、DNSサーバを導入したほうが手間がかからないだろう。

resolv.conf

標準Cライブラリは、DNSなどを利用してIPアドレスとホスト名を変換するしくみを持つ。これをリゾルバ（Name Resolver）といい、ネットワークアプリケーションはリゾルバの機能によって名前解決の結果を得ている。

resolv.confは、リゾルバの設定ファイルで、**リスト2**のような形式になっている。2～3行目ではDNSサーバのIPアドレスを指定している。また、1行目では省略されたドメイン名に対し、付加するドメイン名を指定している。この例では、pc01と指定するとpc01.lm-network.gr.jpの名前解決を行うようになる。

host.conf
nsswitch.conf

host.confは、名前解決をする時に、どのような順番で問い合わせをするかを指定する。インストール直後は、**リスト3**のように記述されているはずだ。1行目で問い合わせの順番を指定する。この例では、まずhostsファイルを参照し、次にDNSを参照する。BINDというのは、UNIX系OSで標準的なDNSデーモンの名前だ。

2行目では、hostsファイルで、1つのホスト名に複数のIPアドレスが対応付けられている時に、問い合わせに対して2番目以降のIPアドレスも返す設定だ。

host.confは、libc5という標準Cライブラリで使われている設定ファイルだが、現在、ほぼすべてのディストリビューションがlibc5に代わりglibc2を採用している。glibc2では、host.confが果たしていた役目はnsswitch.confというファイルに移されている。互換性を維持するため、設定変更は両方のファイルで行ったほうがよいだろう。

nsswitch.confでは、システムデータベースと、NSS（Name Service Switch）を設定する。旧来のUNIXでは、システムデータベースはローカルマシンにあるpasswdなどのファイルを利用していたが、NIS（Network Information Service）などを利用してシステムデータベースをサーバで一元管理することも多くなってきた。このような場合に、システムデータベースの情報源と参照する優先順位をnsswitch.confで設定す

▼リスト1　hostsの例

```
1: 127.0.0.1  localhost.localdomain   localhost
2: 192.168.3.101      pc01
3: 192.168.3.102      pc02
4: 192.168.3.103      pc03
5: 192.168.3.104      pc04
6: 192.168.3.105      pc05
```

▼リスト2　resolv.confの例

```
1: domain lm-network.gr.jp
2: nameserver 192.168.3.20
3: nameserver 192.168.3.22
```

▼リスト3　host.confの例

```
1: order hosts,bind
2: multi on
```

る。また、名前解決のためのネームサービスの参照順もこのファイルで設定する。**リスト4**では、33～35行目でパスワード、シャドウ、グループのシステムデータベースを参照する順序を設定し、38行目ではネームサービスの参照順を設定している。

ファイルの書式については、man snnwitch.confを参照してほしい。

ldap.conf

LDAP(Lightweight Directory Access Protocol)をNSS(Name Service Switch)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。

nss_ldapを導入すると、nsswitch.conf内で指定するサービス名に、filesやnisなどに加え、ldapを指定できるようになる。

services

インターネットサービスのサービス名とポート番号、プロトコルの種類の対応を定義するファイル。RPMファイルでネットワークデーモンをインストールする場合は、そのデーモンに必要な行がインストール時に追加されることがほとんどなので、通常は手で書き換える必要はない。

1024より小さいポート番号を利用する場合は、root権限が必要になるので、自分でサービスを追加する場合は注意してほしい。

aliases

aliases.db

aliasesは、メールアカウントの別名を定義するファイルだ。たとえば、

```
info: taro, hanako
```

と定義しておくと、メール配送プログラムsendmailはinfo宛てのメールをtaroとhanakoに配送する。Red Hat Linux 6.2Jでは、**リスト5**のようなaliasesファイルがインストールされる。これによりpostmasterなどに宛てられたメールはrootに配送されることになる。**リスト5**の末尾にあるように、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ようにしておくと便利だろう。

注意してほしいのは、sendmailはaliasesではなく、これをもとに作られるバイナリファイルのaliases.dbを参照している点だ。従って、aliasesを書き換えたら、aliases.dbファイルを更新しなければ動作に反映されない。aliases.dbファイルの更新にはnewaliasesコマンドを使用する。rootユーザーになり、引数なしで実行すればよい。

```
# newaliases
/etc/aliases: 14 aliases, longest
10 bytes, 152 bytes total
```

inetd.conf

LinuxやUNIXでは、さまざまなネットワークサーバデーモンを動作させることができる。これらのデーモンは、起動されると外部から接続要求があるまでスリープ状態で待機する。しかし、数多くのデーモンを起動すると、待機中もメモリなどのリソースを消費してしまう。そこで考えられたのがinetd(インターネットスーパーサーバ)だ。

inetdは、接続要求を見張るのが仕事で、外部からの接続要求を受けると、対応するサーバデーモンを起動する。これにより、inetdから起動できるネットワークサーバデーモンは常に起動しておく必要がなくなるのだ。

接続要求に対し、どのデーモンを起動すればよいのか、また、どのように起動すればよいのかを指定するのがinetd.confだ。

Linuxディストリビューションを利用する場合、ユーザーがエントリを追加することは稀なので、内容を簡単に解説する。inetd.confは、**リスト6**のような書式になっている。1行につき1つのサーバサービスの内容を記述し、"#"以降はコメントとなる。すなわち、行頭に"#"のあるサービスは起動されない。セキュリティを確保するためには不必要なサービスはコメントアウト

▼リスト4　nsswitch.confの例(一部)

```
33: passwd:     files nisplus nis
34: shadow:     files nisplus nis
35: group:      files nisplus nis
36:
37: #hosts:     db files nisplus nis dns
38: hosts:      files nisplus nis dns
```

▼リスト5　aliasesの例

```
 1: MAILER-DAEMON:  postmaster
 2: postmaster:     root
 3: bin:            root
 4: daemon:         root
 5: games:          root
 6: ingres:         root
 7: nobody:         root
 8: system:         root
 9: toor:           root
10: uucp:           root
11:
12: manager:        root
13: dumper:         root
14: operator:       root
15:
16: decode:         root
17:
18: root:           hiro
```

しておこう。

サービス名には、/etc/servicesで指定されるサービス名を記述する。35行目を見ると、FTPサービスへの接続要求に対しては、root権限で/etc/sbin/tcpdが起動されることがわかる。tcpdへの引数はin.ftpd -l -aだ。ほかの行を見てもtcpdが起動されるものがほとんどで、tcpdを経由して引数で指定されるネットワークサーバデーモンが起動されることになる。

tcpdは、tcp_wrappersというセキュリティパッケージに含まれるデーモンで、リクエストの接続元によってアクセス制限をしたり、syslogデーモンを使ってログを取ることができる。

アクセス制限の指定は、後述のhosts.allow、hosts.denyという2つのファイルで行う。

hosts.allow
hosts.deny

tcp_wrappers（tcpd）によるアクセス制御の設定ファイル。inetdから起動されるネットワークサーバデーモンへのアクセス制御に使用される。hosts.allowには接続を許可するホストを、hosts.denyには拒否するホストをそれぞれサービスごとに記述する。hosts.denyには、

```
ALL : ALL
```

を記述しておき、接続を許可するサービス、ホストをhosts.allowに追加するとよい。たとえば、hosts.allowに、

```
ALL : 127.0.0.1, 192.168.3
```

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192.168.3.0～192.168.3.255のホストだけが利用できるようになる。

ネットワークサーバアプリケーションの設定ファイル

dhcpc/

DHCPクライアントデーモンdhcpcdを使用している場合、DHCPサーバから取得したネットワーク情報はこのディレクトリに保存される。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、通常はpumpという別のDHCPクライアントデーモンが使われるため、このディレクトリは空のままとなる。

また、TurboLinux 6.xでは、dhclientというDHCPクライアントデーモンが採用されており、Red Hat系と同様このディレクトリは使用されない。

ftpaccess
ftpusers
ftphosts
ftpconversions
ftpgroups

FTPサーバwu-ftpdの設定ファイル群。ftpaccessがwu-ftpdの動作の詳細を設定するファイルだ。

ftpusersには、FTPでのアクセスを許可しないユーザーを列挙する。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などが設定済みだ。

ftphostsでホスト、ユーザーごとのアクセス許可／禁止を設定できる。書式はman ftphostsに書かれているので、必要なら調べてほしい。

▼リスト6　inetd.confの例（一部）

サービス名　ソケットタイプ　プロトコル　フラッグ　ユーザー　プログラム　引数

```
35: ftp         stream  tcp     nowait  root            /usr/sbin/tcpd  in.ftpd -l -a
36: telnet      stream  tcp     nowait  root            /usr/sbin/tcpd  in.telnetd
37: #
38: # Shell, login, exec, comsat and talk are BSD protocols.
39: #
40: shell       stream  tcp     nowait  root            /usr/sbin/tcpd  in.rshd
41: login       stream  tcp     nowait  root            /usr/sbin/tcpd  in.rlogind
42: #exec       stream  tcp     nowait  root            /usr/sbin/tcpd  in.rexecd
43: #comsat     dgram   udp     wait    root            /usr/sbin/tcpd  in.comsat
44: talk        dgram   udp     wait    nobody.tty      /usr/sbin/tcpd  in.talkd
45: ntalk       dgram   udp     wait    nobody.tty      /usr/sbin/tcpd  in.ntalkd
46: #dtalk      stream  tcp     wait    nobody.tty      /usr/sbin/tcpd  in.dtalkd
47: #
48: # Pop and imap mail services et al
49: #
50: #pop-2      stream  tcp     nowait  root            /usr/sbin/tcpd  ipop2d
51: #pop-3      stream  tcp     nowait  root            /usr/sbin/tcpd  ipop3d
```

wu-ftpd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ことができる。このときに使われるプログラムと拡張子の対応はftpconversionで定義する。通常は書き換える必要はない。

なお、TurboLinux 6.xでは、これらのファイルは/etc/wu-ftpd以下に配置される。

exports

NFS（Network File System）デーモンを実行すると、特定のディレクトリをネットワークを介してほかのマシンからマウントできるようになる。その際に、マウントを許可するディレクトリと許可するマシン、許可属性をこのファイルに記述する。たとえば、**リスト7**のように記述すると、/home/shareディレクトリをpc01、pc02というマシンから読み書き可能な属性でマウントでき、pc03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。さらに、/home/ftpディレクトリをpc01から読み書き可能な属性でマウントできる。

インストール直後は空のファイルが用意されているので、必要に応じて書き換えよう。

gated.conf.sample

gatedは動的なネットワーク経路設定を行うデーモンだ。同様のデーモンにroutedがあるが、routedがRIP（Routing Information Protocol）のみを使うのに対し、gatedはOSPF（Open Shortest Path First Protocol）などにも対応する。そのgatedの設定を行うファイルがgated.confで、gated.conf.sampleはそのひな形である。

インターネットに接続している場合、不用意にroutedやgatedを動作させるとプロバイダのネットワークなどに悪影響を及ぼすことがあるので注意しよう。

dhclient-script
dhclient.conf.sample

TurboLinuxで採用されているDHCPクライアントデーモンdhclientの設定スクリプトがdhclient-scriptだ。通常はユーザーが変更する必要はない。また、dhclient.conf.sampleはdhclientの設定ファイルの見本だ。dhclientの動作を細かく設定したい場合はこれをひな形にしてdhclient.confファイルを作成する。

httpd/

WebサーバデーモンApacheが使用するディレクトリ。Red Hat系ディストリビューションやTurboLinuxでは、このディレクトリの下にconf、logs、modulesというディレクトリが作られている。

confディレクトリにはApacheの設定ファイルhttpd.conf、srm.conf、access.confが収められている。logs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。

Apacheをソースからビルドした場合や、Plamo Linux、Slackware Linuxでは、/usr/local/apacheディレクトリ以下にApacheに関連するすべてのディレクトリが作られる。

mime.types

WebサーバデーモンApacheが利用するMIMEタイプの定義ファイル。MIMEタイプは、

メディアタイプ／メディアサブタイプ　拡張子

の形式で割り付ける。通常使われるであろうMIMEタイプは設定済みなので、書き換える必要はほとんどない。

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デーモンidentdの設定ファイル。

named.boot
named.conf

DNSサーバデーモンBINDの設定ファイル。named.confがBIND 8の、named.bootがBIND 4の設定ファイルで、両者で書式は異なる。

現在のディストリビューションでは、BIND 8がインストールされているが、BIND 4に慣れているユーザーのためにnamed.bootをnamed.confに変換するPerlスクリプトが用意されている。named.bootを書き換えたら、

```
# named-bootconf < named.boot > named.conf
```

を実行すれば、named.bootからnamed.confが作成される。

news/

NetNewsサーバINNが利用するディレクトリ。各種の設定ファイルなどが

▼リスト7　exportの例

```
1: /home/share pc01(rw) pc02(rw) pc03(ro)
2: /home/ftp   pc01(rw)
```

収められている。

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。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だ。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ので、特にNIS＋では、劇的なパフォーマンス向上が見込まれるという。

ntp/
ntp.conf

NTP（Network Time Protocol）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可能だ。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サーバデーモンが含まれており、他の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。ntpディレクトリは、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で、ntp.confはxntpdの設定ファイルだ。

Linuxマシンの時計を合わせることが目的であれば、xntpdを起動してNTPサーバにする必要はない。xntp3パッケージにはntpdateというNTPクライアントソフトウェアも用意されているので、NTPサーバの名前を引数にしてこれを実行すれば時刻設定をすることができる（**画面1**）。

また、LinuxマシンをNTPサーバにして、フリーのNTPクライアントソフト「桜時計」などをWindowsマシンで動作させれば、LAN内のマシンで時計を同期させることが可能だ（**画面2**）。

nwserv.conf
nwserv.stations

mars_nwe（MARtin Stover's Net Ware Emulator）パッケージを使うと、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。LinuxマシンをNetwareサーバに見せかけることができるわけだ。

nwserv.confは、mars_nweの設定ファイルだ。このファイルで共有するディレクトリなどを設定する。いっぽう、nwserv.stationsには、mars_nweが動作するLinuxマシンがプライマリファイルサーバとなるマシンを登録する。

openldap/

LDAP（Lightweight Directory Access Protocol）アプリケーションと開発ツールを提供するフリーのパッケージOpenLDAPの設定ファイル群を収めたディレクトリ。

LDAP(Lightweight Directory Access Protocol）は、インターネットを介した利用も可能なディレクトリサービスにアクセスするためのプロトコルだ。

pxe.conf
mtftpd.conf

Red Hat Linux 6.2J、LASER5 Linux 6.2には、PXE（Preboot eXecution Environment）サーバパッケージが含まれている。PXEとは、コンピュータの起動、OSのインストール、診断などをネットワーク越しにリモートから行えるようにする技術だ。PXEに準拠したBIOS ROM搭載のネットワークカードがあれば、OSをインストールしていないマシンでもネットワークブートさせることができる。

PXEサーバにLinuxのブートイメージを用意すると、クライアントマシンへLinuxのリモートインストールを行うことが可能になる。

PXEサーバデーモンの設定は、pxe.confファイルで行う。また、ブートイメージを転送するのに必要なMTFTPデーモンの設定はmtftpd.confで行う。MTFTPデーモンの代わりに、TFTPデーモンを利用することも可能だ。

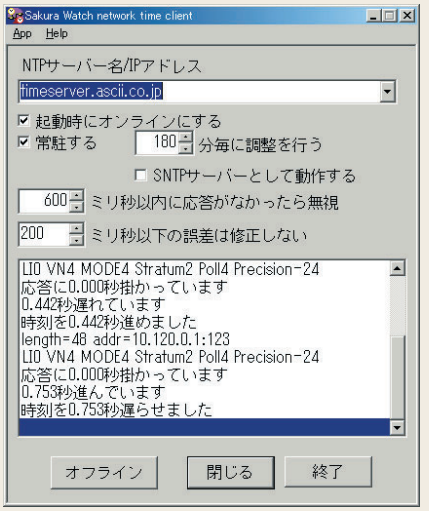
sendmail.cf
sendmail.cw
sendmail.mc
smrsh/

メール配送デーモンSendmailの設定ファイル群。

Sendmailの動作を決定するメインの設定ファイルがsendmail.cfだ。このファイルは、書式が極めて難しいことで有名だ。そのため、よく使われる設定項目を集め、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

```
# ntpdate timeserver.ascii.co.jp
27 Aug 01:24:36 ntpdate[16119]: adjust time server 10.120.0.1 offset 0.000948 sec
```

画面1　ntpdateコマンドの実行例



画面2　WindowsのNTPクライアントソフトウェア「桜時計」

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。

Sendmailは、シェルのコマンドを呼び出して実行することがあるのだが、通常のシェルを利用させるとセキュリティホールになりかねない。そのため、Sendmailから利用するための制限シェルsmrshを利用する。smrshディレクトリには、smrshから利用するコマンドをコピーするかシンボリックリンクを張る。smrshは、このディレクトリにあるコマンドしか実行できないので、セキュリティが確保できる。

smb.conf
smbpasswd
smbusers
lmhosts
codepages/

Windowsとファイル／プリンタを共有するためのデーモンSambaの設定ファイル群。

Sambaの動作を決めるのがsmb.confだ。ここで共有するディレクトリやプリンタの設定を行う。

smbpasswdは、Sambaを利用するユーザー名と符号化したパスワードが記述されている。また、smbusersは、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だ。

lmhostsはSambaがNetBIOS名（Widnowsネットワークでのホスト名）の解決に利用するホスト名、IPアドレスの定義ファイルだ。hostsファイルとほぼ同じ書式で書かれている。

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る。

snmp/

SNMP（Simple Network Management Protocol）に対応するパッケージucd-snmpで利用するディレクトリ。SNMPデーモンの設定ファイルsnmpd.confが置かれているので、SNMPデーモンを利用する場合はこのファイルを自分の環境に合わせて書き換える。

squid/

Webキャッシュサーバsquidの設定ファイルを格納するディレクトリ。squidの設定は、ここにあるsquid.confで行う。初期状態では、localhost、すなわちLinuxマシン自身からしか利用できないようになっている。

Squidは、比較的設定も簡単で、導入効果も大きなサーバアプリケーションだ。**リスト8**を参考にして、ほかのマシンからも利用できるようにして試してみるとよいだろう。この例では、IPアドレスが192.168.3.1～192.168.3.255のホストからもSquidを利用できるようにしている。

Squidを起動したら、クライアントマシンのWebブラウザでプロキシサーバを指定する。サーバ名はLinuxマシンを、ポートは3128などsquid.confで設定したものにする。

ssh/

TurboLinux 6.xでは、ssh（Secure Shell）を実装したフリーのパッケージOpenSSHがインストールされる。sshは、データを暗号化したうえでリモートログインやファイルの転送を行うパッケージだ。telnetやrlogin、rcpの代わりに使うことで安全にリモートマシンを利用することが可能になる。

OpenSSHには、クライアントとサーバ両方のプログラムが含まれており、このディレクトリ以下にあるssh_configがクライアントの、sshd_confがサーバの設定ファイルだ。

▼リスト8　squid.confの例（一部）

```
 45: http_port 3128                                  ポート番号。8080などもよく使われる
                :
                :
585: ftp_user taro@lm-provider.ne.jp                 FTPサーバに渡すメールアドレスを設定
                :
                :
1153: acl all src 0.0.0.0/0.0.0.0
1154: acl manager proto cache_object
1155: acl localhost src 127.0.0.1/255.255.255.255
1156: acl mynetwork src 192.168.3.0/255.255.255.0    mynetworkという名前でLANを登録（この行追加）
                :
                :
1195: http_access allow localhost
1195: http_access allow mynetwork                    mynetworkからのアクセスを許可する（この行追加）
1197: http_access deny all
```

アプリケーションなど

文：編集部
Text : Linux magazine

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。

アプリケーションソフトウェアの設定ファイル

Muttrc
charsets/

Muttは、動作が軽快なテキストベースのメールソフトウェアだ（**画面1**）。メール一覧をスレッド表示したり、メールにファイルを添付することもできるなど、機能は十分で、telnetなどでリモートログインして使えることはテキストベースゆえの強みでもある。

そのMuttの設定ファイルがMuttrcだ。ほとんどの項目がコメントアウトされたかたちで用意されているので、自分の環境に合わせて書き換えよう。

charsetには、各国語のキャラクタセット定義ファイルが用意されている。この中のファイルをユーザーが書き換える必要はない。

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 2000
```

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、引数で指定されたダンプレベルより小さなダンプレベルのバックアップ記録がdumdatesファイル中に見つかれば、そこに記録された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。

Linux用のdumpコマンドは、ext2ファイルシステムにしか対応していないので注意してほしい。dumpコマンドで取ったバックアップは、restoreコマンドでリストアする。

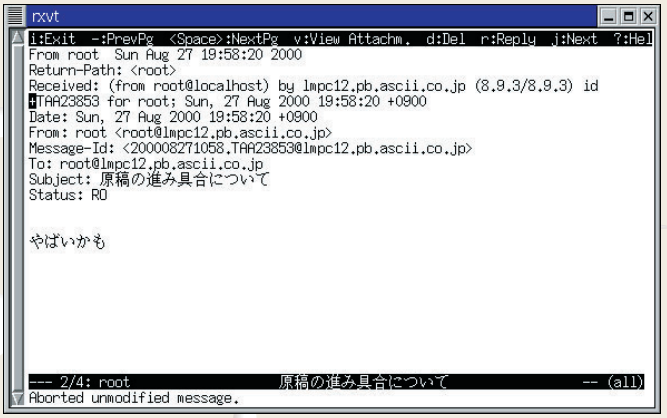
info-dir

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。Linuxディストリビューションには、テキストベースのtexinfoリーダinfoが含まれており、

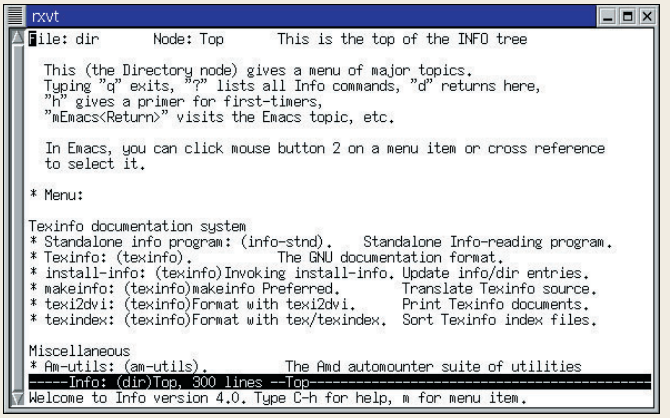
```
$ info ls
```

のようにしてドキュメントを読むことが可能だ。

infoは、texinfoのハイパーテキストに対応しており、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（**画面2**）。



画面1　Muttの画面



画面2　引数なしでinfoを起動し、info-dirファイルを表示

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ログファイルを分割し、古いログを圧縮／削除する。

WebサーバデーモンのApacheなど、RPMファイルからデーモンパッケージをインストールすると、logrotateの設定ファイルがlogrotate.dディレクトリにインストールされる。Red Hat系ディストリビューションやTurboLinuxでは、logrotate.dディレクトリ以下にインストールされるファイルをlogrotate.confから呼び出すことで、さまざまなデーモンのログを処理できるようにしているのだ。

Red Hat Linux 6.2Jの/var/logディレクトリを見てみると、ログは1週ごとに分割され、古いログにはファイル名の末尾に.1、.2などの数字を付けて4週間分保存されているのがわかる（**画面3**）。

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。

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、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1**）。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。

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ふつうだったのだ。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みなさんもご存じのとおりだ。

```
# ls -p /var/log/
boot.log    cron.4          maillog.4      phhttpd/     spooler    wtmp.1
boot.log.1  dmesg           messages       samba/       spooler.1  xferlog
boot.log.2  fax/            messages.1     savacct      spooler.2  xferlog.1
boot.log.3  htmlaccess.log  messages.2     secure       spooler.3  xferlog.2
boot.log.4  httpd/          messages.3     secure.1     spooler.4  xferlog.3
canna/      lastlog         messages.4     secure.2     squid/     xferlog.4
cron        maillog         netconf.log    secure.3     usracct
cron.1      maillog.1       netconf.log.1  secure.4     uucp/
cron.2      maillog.2       news/          sendmail.st  vbox/
cron.3      maillog.3       pacct          snmpd.log    wtmp
```

画面3　Red Hat Linux 6.2Jのログディレクトリ

▼リスト1　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）

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なのだ。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。

Red Hat Linux 6.2Jのtermcapを見てみると、ファイルの末尾にkonのエントリが追加されている（**リスト2**）。環境変数TERMに“kon”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。

▼リスト2　termcapファイルにあるkonのエントリ

```
14626: kon|kanji on console:\
14627:  :am:eo:mi:ms:ut:xn:xo:\
14628:  :co#80:it#8:li#25:\
14629:  :&7=^Z:@7=\E[4~:AL=\E[%dL:DC=\E[%dP:DL=\E[%dM:\
14630:  :F1=\E[23~:F2=\E[24~:F3=\E[25~:F4=\E[26~:F5=\E[28~:\
14631:  :F6=\E[29~:F7=\E[31~:F8=\E[32~:F9=\E[33~:FA=\E[34~:\
14632:  :IC=\E[%d@:K2=\E[G:al=\E[L:bl=^G:cd=\E[J:ce=\E[K:\
14633:  :ch=\E[%i%dG:cl=\E[H\E[J:cm=\E[%i%d;%dH:cr=^M:\
14634:  :cs=\E[%i%d;%dr:ct=\E[3g:cv=\E[%i%dd:dc=\E[P:dl=\E[M:\
14635:  :do=^J:ei=\E[4l:ho=\E[H:ic=\E[@:im=\E[4h:k1=\E[[A:\
14636:  :k2=\E[[B:k3=\E[[C:k4=\E[[D:k5=\E[[E:k6=\E[17~:\
14637:  :k7=\E[18~:k8=\E[19~:k9=\E[20~:k;=\E[21~:kB=\E[Z:\
14638:  :kD=\E[3~:kI=\E[2~:kN=\E[6~:kP=\E[5~:kb=^H:kd=\E[B:\
14639:  :kh=\E[1~:kl=\E[D:kr=\E[C:ku=\E[A:le=^H:mr=\E[7m:\
14640:  :nd=\E[C:nw=^M^J:r1=\Ec:rc=\E8:sc=\E7:sf=^J:sr=\EM:\
14641:  :st=\EH:ta=^I:u6=\E[%i%d;%dR:u7=\E[6n:u8=\E[?6c:\
14642:  :u9=\E[c:up=\E[A:vb=200\E[?5h\E[?5l:ve=\E[?25h:\
14643:  :vi=\E[?25l:tc=klone+sgr:tc=klone+color:\
14644:  :hs:es:ts=\E[?T:fs=\E[?F:ds=\E[?H\E[?E:
```

wgetrc

wgetは、HTTPとFTPに対応した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

▼リスト3　wgetrcの例（一部）

```
 28: # You can lower (or raise) the default number of retries when
 29: # downloading a file (default is 20).
 30: tries = 25                                  リトライ回数の上限（コメントを外して書き換える）
                :
                :
 65: # You can set the default proxy for Wget to use.  It will override the
 66: # value in the environment.
 67: http_proxy = proxy.lm-provider.gr.jp:8080   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）
 68: ftp_proxy  = proxy.lm-provider.gr.jp:8080   FTPプロキシとポートの指定（新たにこの行を加える）
 69:
 70:
 71: # If you do not want to use proxy at all, set this to off.
 72: use_proxy = on                             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）
```

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3**）。

このように、非常に強力な機能を持つwgetにも欠点はある。それは、コマンドラインオプションが複雑なことと、Webブラウザとの連携が考慮されていないことだ。これらの問題を解消するためには、GUIのフロントエンドプログラムを使えばよい。

ここでは、GNOMEで動作するGTransferManagerを紹介する（**画面4**）。GTransferManagerのWebページ（http://gtm.sourceforge.net/）からRed Hat Linux 6.x用のRPMファイルやソースコードを入手できる。メニューからwgetの設定ができるので、環境に合わせて設定すれば準備は終了。あとは、Netscape NavigatorからGTransferManagerのウィンドウへ、マウスを使ってリンクをドラッグ&ドロップすればよい。実際のダウンロードは、バックエンドで起動されたwgetが行ってくれる。たくさんのファイルをダウンロードしたい人には必須のソフトウェアだといえよう。

各ディストリビューションに固有のファイル

パッケージ管理システムにRPMを利用しているディストリビューション間では、/etcディレクトリに置かれるファイルに大きな違いはない。しかし、ディストリビューションごとに専用の環境設定ツールを持っていたり、提供されるソフトウェアが違っているため、特定のディストリビューションにだけ含まれる設定ファイルもある。

ここでは、そのようなファイルをいくつか紹介する。

●Kondara MNU/Linux

mph.conf

mph-conf.conf

Kondara MNU/Linuxが備える環境設定ツール群mphの設定ファイルだ。

odbc.ini

odbcinst.ini

PostgreSQl、MySQL、MiniSQLのための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。

ethereal/

ネットワークパケットをダンプするツールetherealの利用するディレクトリ。ネットワーク機器のMACアドレスとメーカー名が書かれたmanufファイルが収められている。

●TurboLinux

turbocentro-cfg.xml

turbopkgrc

turbowmcfgrc

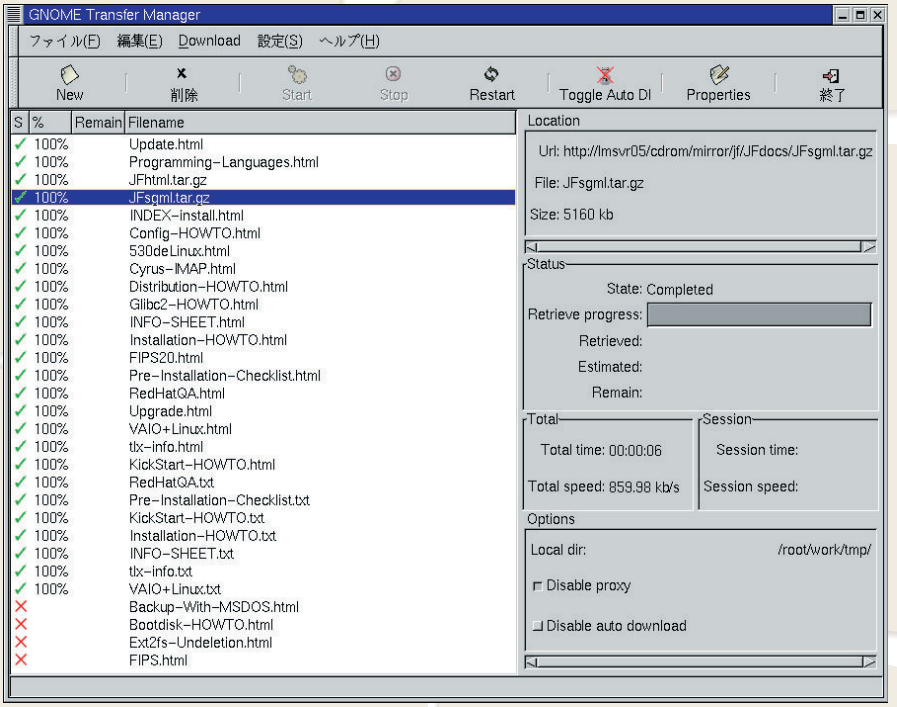
TurboLinuxは、Turbo Toolsと呼ばれる、環境設定ツール群を提供する。これらのファイルはそれらの設定ファイルである。

atalk/

TurboLinux 6.xには、Macintoshとネットワークでつなぎ、ファイルやプリンタを共有するnetatalkが含まれている。このディレクトリにはnetatalkの設定ファイルが収められている。

ssh/

フリーの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。ほかのディストリビューションを使用している場合でも、プロジェクトのWebサイト（http://www.openssh.com/）からOpenSSHを入手可能だ。



画面4　wgetのGUIフロントエンドGTransferManager

/etcディレクトリとのつきあい方

文：編集部
Text : Linux magazine

ここまでで見てきたとおり、/etcディレクトリにはたくさんの設定ファイルがあり、しかもその働きや書式はまちまちだ。Linuxのコミュニティやディストリビューターは、システムの設定をより簡便に行えるよう、GUIの環境設定ツールを開発し提供している。

GUI環境設定ツールの意義

Linuxconf（**画面1**）に代表されるGUIの環境設定ツールは、さまざまな設定を一括して、しかも統一されたユーザーインターフェイスによって行えるのが特長だ。これによって、/etcディレクトリのみならず、さまざまなディレクトリに散らばっている設定ファイルを個別に編集する苦しみから解放される。

Kondara Projectでは、システムを運用するにあたり、特に重要となる設定に焦点を絞ったmphというツールを提供している（**画面2**）。カーネルの再構築設定ができるなど、いままでにないアプローチは、今後が楽しみだ。

TurboLinuxも、Turbo Toolsという独自の設定ツール群を提供しており、Linuxに不慣れなユーザーの強い味方として評価を得ている。

しかし、環境設定ツールは万能ではない。設定ファイルを直接編集するのに比べ、細かな設定ができなかったり、メニュー項目が膨大になりすぎて、かえって全体の見通しが悪くなる危険性をはらんでいるのだ。また、設定ツールに頼りすぎるとディストリビューションを替えたとたんにシステム設定のしかたがまったくわからない状況に陥る可能性もある。

これらのツールも活用しつつ、各種設定ファイルの内容も理解できるようになることがLinuxをうまく使いこなす秘訣だといえるだろう。

今後の/etcディレクトリ

Linuxを始めとする、UNIX系OSのファイル配置の標準化も進められている。先頃、Filesystem Hierarchy Standard（FHS）2.1が発表され（http://www.pathname.com/fhs/）、Kondara Projectがいち早くこれに対応することをを表明した。

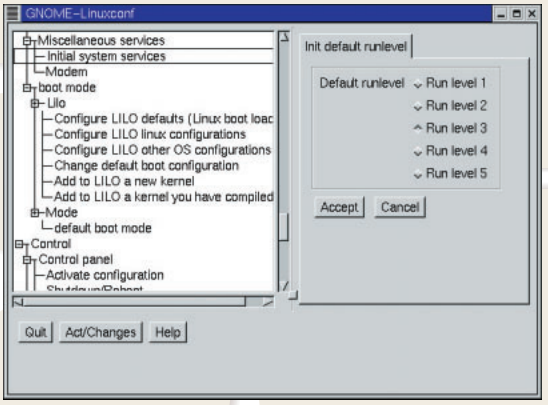
現在、RPMを採用しているディストリビューションの多くは、Linux Filesystem Structure（FSSTND）に準拠したディレクトリ配置を採用しているが、FHS 2.1では、/etcディレクトリ以下のサブディレクトリ配置が変更されている。今後、FHS 2.1への移行がどのように進行するのか注目したい。

/etcディレクトリとのつきあい方

/etcディレクトリにある設定ファイルの多くがテキストファイルだ。ファイルの内容や書き方について、詳細なコメントが付けられているものも少なくない。興味を持ったらlessなどのテキストビューアで中をのぞいてみよう。内容がわからないときはmanやinfo、/usr/docディレクトリにインストールされたドキュメントが役に立つはずだ。

個々のファイルとじっくりつきあってみれば、Linuxへの理解も深まるし、より便利な環境が手に入るだろう。

ただし、ファイルを書き換える場合は、元のファイルをバックアップしておくことをお勧めする。



画面1　環境設定ツールの代表格Linuxconf

画面2　Kondara MNU/Linuxの統合環境設定ツールmph

最新Linuxカーネルを使う！

はじまる、はじめる カーネル2.4

文：カーネル2.4調査委員会
Text：Kernel 2.4 Investigation Commitee

2.2から2.4へ……。1999年1月の2.2.0以来、ほぼ2年ぶりのリリースとなる次期安定版カーネルは、どのような先進性をもって姿を現わすのか？　最新テストバージョンをもとに、その機能を解説／検証し、既存Linux環境への実際の導入にチャレンジする。

カーネル2.4の新機能

どんな選択肢があるのか

開発版の2.3系カーネルは2.3.51をもって機能追加が終了となり、2.3.99-pre1以降、カーネル2.4.0のテストフェーズへと移行した。これが今年の3月のこと。その後、意欲的にテスト／デバッグが行われ、5月にはlinux-2.4.0-test1がリリース。原稿執筆時点ではtest6までリリースされている。

すでにテストフェーズに移り、ベースとなる機能面では「コードフリーズ」状態にある。そこで、本特集ではカーネル2.4の新機能を紹介し、あわせて2.4へのアップグレード方法、新機能の実際について解説していくことにする。

カーネルコア

カーネル2.4のコア部分に追加される機能は、ハイエンドなサーバ用途においてアドバンテージをもたらすものが多い。プロセス管理やスケジューラのアルゴリズムの変更による高負荷状態でのパフォーマンスの改善、マルチプロセッサ環境でのスケーラビリティの向上、64Gバイトまでのメモリサポート、Raw I/Oデバイス[*1]やLVM（後述）のサポートなどがそれにあたる。これらは、Linuxのビジネス環境への浸透を後押しすることになるはずだ。

ほかにも、ファイルキャッシュやネットワークソケットの処理の効率化、あらゆるデバイスドライバの機能改善といったブラッシュアップが施されており、企業レベルだけでなくパーソナルユースの場面においても、この新しいLinuxカーネルは多くの利点をユーザーに提供するはずだ。

ハードウェアサポート

ネットワークカードやサウンドカード、IDE／SCSIコントローラなど、多くの分野で数多くの製品が新たにカーネルレベルでサポートされるようになった。限られた誌面で、そのすべてを個別に取り上げることはもちろんできないが（サポートリストはLinuxHardware.netなどを参照）、ハードウェア関連での注目される変更点をなるべく多く紹介していくことにしよう。

■ハードウェアアーキテクチャ

Linuxはこれまでにも、PowerPCやAlphaなど、さまざまなCPU向けに移植されてきた。カーネル2.4では、新たに対応プラットフォームとして、Intelの64ビットプロセッサItanium（IA64）、IBMのメインフレームS/390、Windows CE向けのアーキテクチャとして知られている日立のSuperHが追加されている。

それぞれの移植作業は各プロジェクト（**表1**を参照）で別途進められているが、ベース部分はカーネルのソースツリーに組み込まれている。

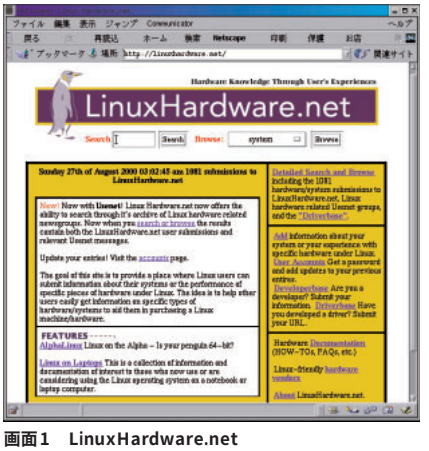
■USB

USBはこれまでにも、2.3系から反映されたパッチ（バックポートパッチ）や別個のプロジェクトで開発されたモジュールによって、2.2系のカーネルを採用したディストリビューションで一部（USBキーボードなど）が利用可能であった。カーネル2.4では、これらの成果が統合されて、USBインターフェイスの制御機能およびデバイスが標準でサポートされるようになった。

USBホストコントローラ規格については、Intelが提唱するUHCI（Universal Host Controller Interface）とCompaq/Microsoft/National Semiconductorが提唱するOCHI（Open Host Controller Interface）の双方ともサ

サイト	URL
LinuxHardware.net	http://linuxhardware.net
LinuxSH	http://linuxsh.sourceforge.net/
IA-64 Linux Project	http://www.linuxia64.org/
LINUX for S/390	http://oss.software.ibm.com/developerworks/opensource/linux390/

表1　本文中で紹介したプロジェクトのURL



画面1　LinuxHardware.net

ポートしている。

USBデバイスのサポートも充実している。プリンタ、スキャナ、オーディオ、モデム、シリアルコンバータ、ストレージデバイスなどだ。加えて、キーボード、マウス、ジョイスティックといったHID（Humam Interface Devices）と呼ばれるデバイス群もカーネルレベルでサポートされている。製品個別では、KodacのDX-2XXシリーズ（デジカメ）、Diamond MultimedeiaのRio500（MP3プレーヤ）などのドライバが追加された。

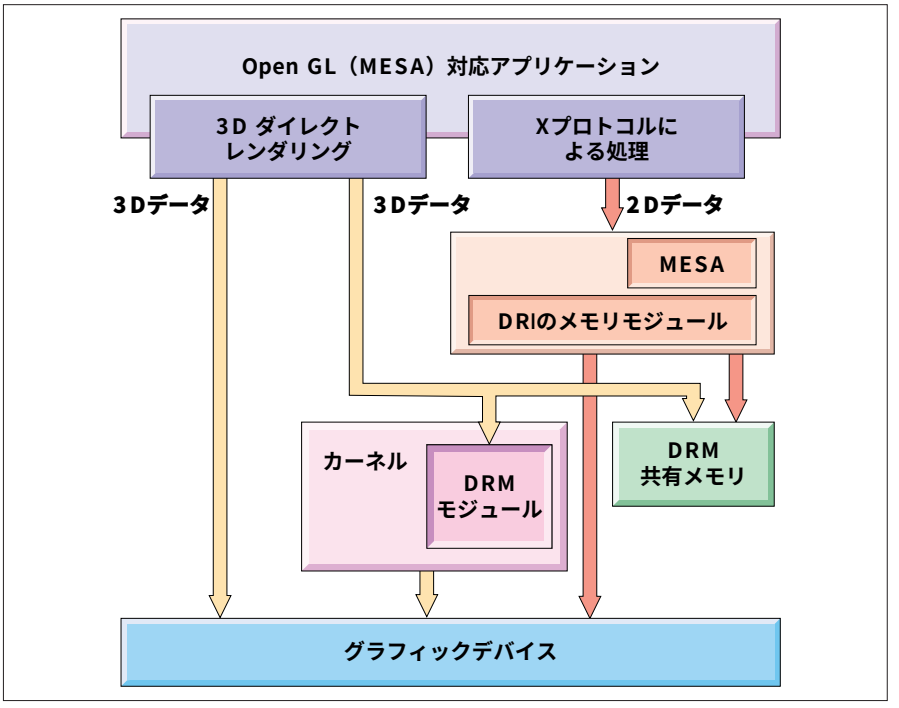


図1　DRIとDRMの動作概念

■PCMCIA

これまでカーネルとは別に開発・配布が行われてきたPCMCIA（PCカード）のサポート（pcmcia-cs）が、カーネルに統合された。これには、PC CardBusへの対応も含まれている。コアのコントロール部分だけでなく、ネットワークカード、無線LAN、IDE／SCSIアダプタなど、一部のデバイスもカーネルレベルでサポートされている。

PCMCIAの完全なサポート（今回サポート外となったデバイスや管理ツール）には、引き続きカーネル外部のデバイスドライバモジュールとプログラムが必要となる。詳細は、pcmcia-csプロジェクトのWebサイト（http://pcmcia-cs.sourceforge.net/）を参照にしてほしい。

■IEEE1394

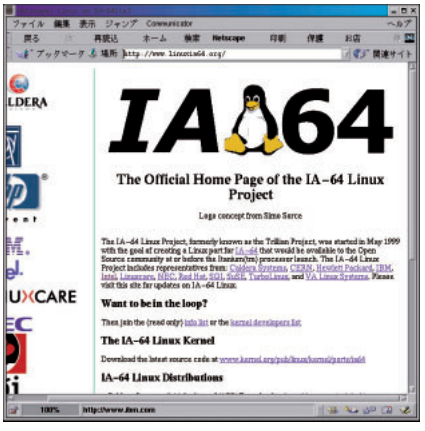
一般に「FireWire」または「i.Link」として知られているIEEE1394もサポートされている（インターフェイスのコア部分と一部のデバイスのサポートが組み込まれている）。執筆時点の最新テストバージョン（linux-2.4.0-test6）においては、「EXPERIMENTAL（試験的なサポート）」となっているが、正式リリースにも含まれるものと思われる。

■ビデオカード

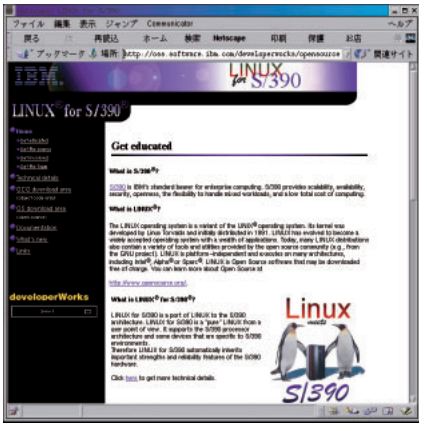
ビデオカードそのものについては特筆すべき変更点はないが、XFree86

Glossary

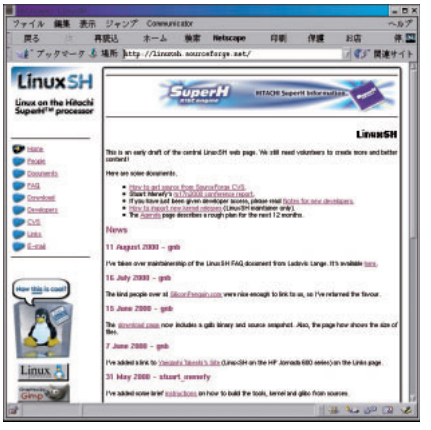
＊1 Raw I/Oデバイス
カーネル内部でのバッファリングを使用せずに直接アクセスできるデバイス。ディスクアクセスのタイミングをアプリケーション側で制御できるので、分散したデータに交互にアクセスするデータベースシステムなどにおいて、最適なパフォーマンスを実現するために使用される。



画面2　The IA-64 Linuxプロジェクト



画面3　IBMのLinux for S/390サイト



画面4　LinuxSHプロジェクト

4.0で導入されたDRI（Direct Rendering Infrastructure）に関連する新しい機能が追加された。DRM（Direct Rendering Manager）と呼ばれる、グラフィックス処理の制御機能がそれだ。

DRIはXでの3Dグラフィックスのレンダリングを高速化するためのメカニズムで、DRMはそのカーネル側でのサポート機能にあたる（**図1**）。DRMは、複数のプロセスからのグラフィックデバイスへのアクセスを制御することで、高速なレンダリングを安定して継続できるように動作する。

■ACPI

直接のハードウェアサポートではないが、パワーマネージメント方式としてACPI（Advanced Configuration and Power Interface）が追加されている。もちろん、この機能を利用するには、PC側（BIOS）でACPI機能がサポートされていることが前提となる。なお一部の機能は、現段階では「EXPERIMENTAL」となっている。

デバイスファイル機構の変更

Devfs（Device File System）は、Linuxにおけるデバイスファイルの管理・制御の機構を一新する機能だ。従来、ハードディスクやフロッピーなどのブロックデバイス、コンソールやシリアルポートなどのキャラクタデバイスへのアクセスは、/devディレクトリ以下の特殊なデバイスファイル（/dev/hda、/dev/fd0など。デバイスノードとも呼ばれる）へのリード／ライトによって実現されていた。これらのデバイスファイルはインストール時に作成され、標準で作成されないデバイスについてはユーザー（システム管理者）がmknodコマンドなどで作成する必要があった。インストール時に作成される標準的なデバイスファイルも、実際にはシステムに存在しないデバイスのものがあり、無駄にディスクスペースを消費するなど、必ずしも効率的ではなかった。

一方、Devfsは動的なデバイスファイルの生成を実現しており、システムに実在するデバイスについてのみデバイスファイルが作成される。このとき、ルートファイルシステム上に直接デバイスファイルが作成されるのではなく、メモリ上のファイルシステム（Devfs）にエントリが登録される。実際には、登録されたデバイスエントリの情報を持つDevfsのネームスペースが/devディレクトリにマウントされるのだ。

デバイスファイルの識別に使用されていたメジャー／マイナー番号による方式もDevfsでは使用されない。デバイスは、Devfsのネームスペースに登録されたエントリ情報に基づいて識別される。ネーミング規則も変更されている。**図2**にIDEドライブの例を示すので、参考にしてほしい。ネーミング規則の変更によって発生する従来方式との非互換性の問題は、devfsdデーモンによって解消されている（devfsdはデバイスファイルの登録やパーミッション管理も行う）。このデーモンはカーネルのソースには含まれていないので、別途Webサイト（http://www.atnf.csiro.au/~rgooch/linux/＝Devfsの主開発者であるRichard Gooch氏のサイト）からダウンロードする必要がある。

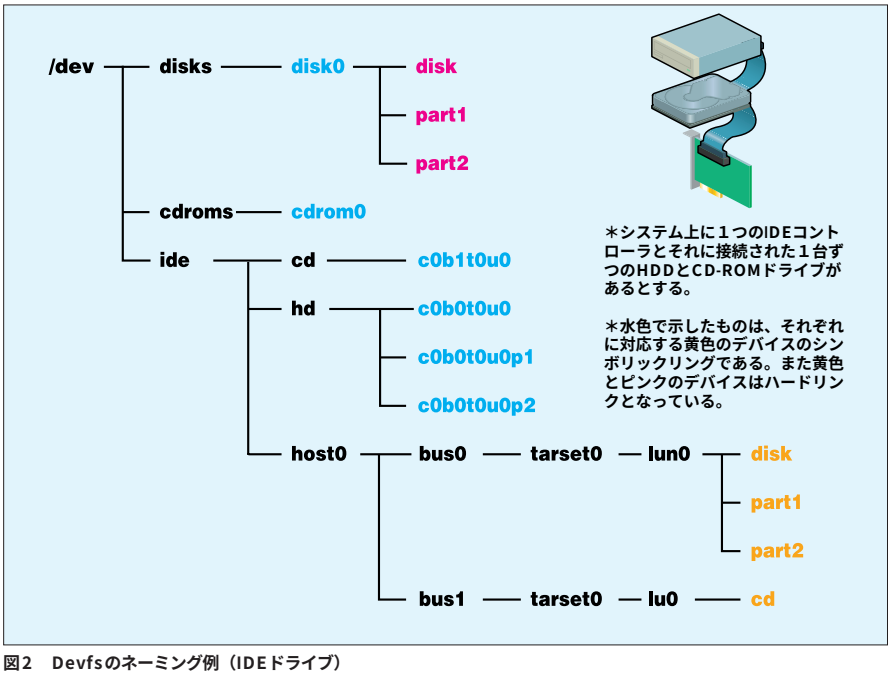


図2　Devfsのネーミング例（IDEドライブ）

ファイルシステム

噂されていたジャーナリング機能[*1]を持つファイルシステム（ReiserFS、XFS）のサポートは、今回は見送られたようだ。ただし、それぞれに開発が進んでおり、Kondara MNU/Linuxのように、ディストリビューション独自の実装によりこれらのファイルシステムが採用される可能性はある。

Linuxをはじめとする多くのオペレーティングシステムでは、複数のフォーマットの異なるファイルシステムを

扱うための仕組みとして「仮想ファイルシステム（VFS）」と呼ばれる機能を持っている。カーネル2.4では、このVFSにも変更が加えられ、ext2で扱える最大ファイルサイズの2Gバイトという制限も取り払われている。

対応ファイルシステムも追加され、SCO UnixWareで採用されているBFS、SGIのIRIXの1世代前のファイルシステムであるEFS、DVDのファイルシステムであるUDFなどがサポート対象として追加された。また、リードのみサポートされていたOS/2のHPFSが完全にサポートされるようになっている。前述のDevfsも新しいファイルシステムのひとつである。

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

ユニーク&ユースフル

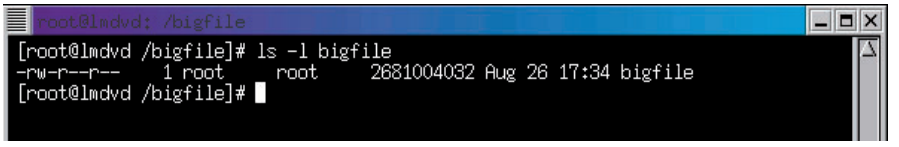
このほかにも、ユニークな機能や「使える」機能が多くある。そのうちのいくつかを紹介しよう。

■LVM

LVM（Logical Volume Manager）は、複数のディスクデバイス上のパーティションをひとつの仮想的なボリュームとして扱うことを可能にする機能だ。この論理ボリュームは、パーティションの追加や削除、サイズ変更を行え、より柔軟なディスク管理が実現できる（**図3**）。データベースサーバなど、使用領域の拡張が必要なケースで有効な機能だ。

新しく追加したドライブ上のパーティションを、既存の論理ボリュームに追加してサイズ変更すれば、既存のデータに手を加えずにディスク領域を拡張できる。これはパーソナルユースでも、うれしい機能だ。単独では使いでのない容量の小さな古いディスクドライブを再利用したい場合などにも使えるだろう。

LVMの機能を利用するためのツールはカーネルとは別に配布されている。LVM用ツールは、Linux LVMのサイト（http://linux.msede.com/lvm/）からダウンロードできる。



画面5　2Mバイトオーバーのファイルをlsで表示

■kHTTPd

kHTTPはカーネルに統合されたHTTPサーバ機能である。今やWebブラウザは、すべてのクライアントコンピュータに必須ともいえるアプリケーションだ。異なるプラットフォーム間でのデータ交換も手軽に行える。また、ユーザーもWebベースのインターフェイスに慣れている。つまり、Web（HTTP）をベースとしたサービスは非常に有効なデータ配信の手段だということだ。

kHTTPdは、CGIなどの機能はサポートしていないが、LAN環境でのデータの受け渡しなどで有効なソリューションになりえるだろう。カーネルモードで動作するため、高速な処理処理も期待できる。kHTTPdは単純なデータ転送のみを担当し、通常のWebサーバの補助的な役割を果たすといったことも可能だ。

Glossary

＊1 ジャーナリング機能
ファイルシステムに対する変更をログ（このログがジャーナルと呼ばれる）に記録し、障害時のデータ復旧を支援する機能。サービスのダウンタイムを短縮する効果がある。

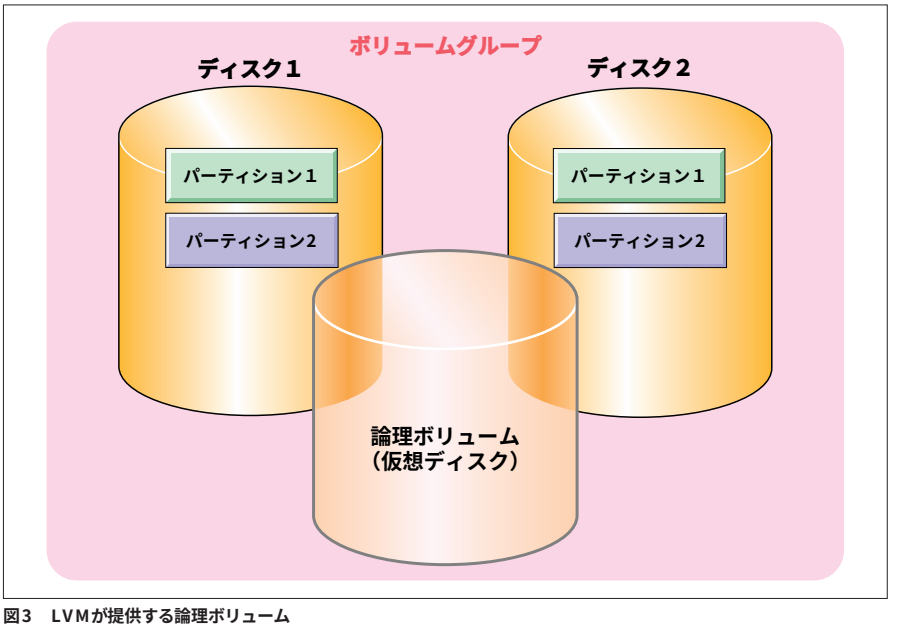


図3　LVMが提供する論理ボリューム
ディスク1とディスク2上のパーティションを1つの「ボリュームグループ」として定義し、そのボリュームグループ内に「論理ボリューム」を仮想的なレイヤとして作成できる。

Column

カーネルって何だ？

Linux magazineに限らず、多くのコンピュータ系雑誌で、カーネルという単語は特に注釈もなく用いられている。だが世間一般では、「カーネル」といわれて思い浮かぶのは、鶏肉を食材にしたファーストフードチェーンのほうであろう。ちなみにそちらのほうは、発音は同じだが“colonel”と綴り、「大佐」という意味を持つ。もちろん、Linuxの“kernel”とは別ものだ。

オペレーティングシステム（OS）のカーネルは、「縁の下の力持ち」的存在であり、ユーザーが当たり前のように感じている機能の大部分を提供している。一般的なOSカーネルが持つ機能は、以下のようなものだ。

●プロセス管理

最近のOSは、複数のプログラムを同時に動作させることができるようになっている。システム内で動作しているプログラムは、プロセスと呼ばれる単位で管理されている。実際には、各プロセスを非常に短い間隔で切り替えることで、ユーザーからは同時に動作しているように見えるようになっている。

カーネルは、各プロセスにどの程度の時間を割り当てるか、どのような優先順位で動作させるかを決定している。

●プロセス間通信

UNIX系のOSでは、複数のプロセスが協調して働くことが多い。シェルプロンプトからなにげなく使っているパイプ（|）や、signalコマンドによってシグナルを送り、プロセスを停止したり再起動するのがその例だ。

このようなプロセス間通信の機能は、OSごとにさまざまな種類があるが、いずれもカーネルによって提供されている。

●メモリ管理

もっとも基本的なメモリ管理は、プロセスからの要求に応じてメモリを割り当てたり、終了したプロセスに割り当てていたメモリを「フリー」の状態に戻す（解放）といったことだ。また物理的なメモリが不足した場合の、仮想記憶（スワップイン／アウト、ページイン／アウト）もカーネルによって制御されている。

●デバイスドライバ

PCに接続されるハードウェアデバイスは、非常に種類が多い。これらのハードウェアを共通した方法でアクセスできるようにするため、用意されているプログラムがデバイスドライバである。

多数のデバイスドライバをカーネルに組み込んでしまうと、カーネルサイズが巨大になってしまい、メモリが無駄に使われてしまうことになる。Linuxでは、後述するカーネルモジュールという手法でこの問題を回避している。

●ファイルシステムサポート

PCで用いられる記憶装置は、ハードディスク以外にもCD-ROMやフロッピーディスク、MOなど多岐にわたる。そして、データはLinux用のext2やWindows用のFAT、NTFSといったファイルシステムに記録されている。またハードディスクでは、複数のファイルシステムがパーティションで区切られている場合がある。

だが、どの記憶装置でもlsコマンドでファイルリストを表示し、cpコマンドでファイルをコピーすることができる。このように記憶装置やパーティション形式の差異を吸収し、同じようにアクセスできるようにするのが、ファイルシステムサポートの役目だ。

●ネットワーク

今やPC用のOSで、ネットワーク機能を持たないものなど考えられない。デファクトスタンダードであるTCP/IPをベースにした各種プロトコルやサービスが、カーネルによって提供されている。

これらの機能は、システムコールという仕組みによって、アプリケーションプログラムから利用される。システムコールは、カーネル内部に存在するサブルーチンの集合体で、UNIX系OSでは、C言語のライブラリとしてシステムコールへのインターフェイスが提供されている。

LinuxであれWindowsであれ、システムが順調に動作しているかぎりは、OSやカーネルの存在を意識する機会はそんなに多くない。というよりむしろ、自らの存在をユーザーに意識させずに、作業に専念させてくれるのがOSであり、カーネルなのだ。

モノリシックカーネルとマイクロカーネル

初期のUNIXは今ほど複雑ではなかった。そもそも“UNIX”という名前は、“MULTICS”という名の高機能だが複雑なOSへのアンチテーゼとして付けられたものなのだ。ところがOSに要求される機能が増えるにつれて、カーネルも肥大化／複雑化が進み、その結果、新しい機能の付加や改良が困難になってしまった。このような巨大で複雑なカーネルをモノリシックカーネルという（“monolithic”は、「一枚岩の」などの意味。時々「モノシリック」という誤記を雑誌などで見掛ける。「物知り」なカーネルなのだろうか？　ちょっと恥ずかしい間違いだ）。

そこで、カーネルに組み込まれている要素から、必要最小限な機能だけを取り出して、ごく小さなカーネルを作り、それ以外の機能はサーバプロセスとしてカーネルの外で動作させるというシステムが考え出された。この小さなカーネルをマイクロカーネルと呼ぶ。マイクロカーネルを採用したOSとしては、Windows NT／2000、Machなどがある。またMacintosh用のMkLinuxは、マイクロカーネル構造をとっている。

●どっちがいいの？

マイクロカーネル、モノリシックカーネルのそれぞれに、長所／短所がある。

マイクロカーネルは、複雑になり過ぎて機能の追加が困難になったモノリシックカーネルの反省から生まれたため、機能ごとに分割された構造になっている。一部の機能を改良したり、新機能を追加したりする際にも、手を加えるべき部分がはっきりしており、容易に実現できる（もちろん現実の実装の際には、そんな理想どおりにはいかないことは、Windows NT／2000がいつまでたっても問題

を抱えていることからも明らかだが)。またサーバプロセスが分割されているため、マルチプロセッサシステムとの親和性が高く、性能を上げやすいと言われている。

構造的には優れているマイクロカーネルだが、性能の点ではモノリシックカーネルよりも劣っている。これは、システムコールの処理が実際に行われる際の流れを考えてみれば、すぐにわかることだ。

モノリシックカーネルでは、システムコールの要求を受け取ったカーネルが内部で処理を行い、結果を返す。それに対してマイクロカーネルでは、カーネルがシステムコールの要求を受け取ってからサーバプロセスへと送り、サーバプロセス内部で処理される。処理の結果は、逆の流れでシステムコールを発行したプログラムへ戻される。

また、サーバプロセスは、一般のプログラムと同じユーザーモードと呼ばれるモードで動作しているが、ユーザーモードとカーネルモードの切り替えはかなり時間のかかる処理であり、システムコール発行のたびに切り替えが多発するマイクロカーネルが、性能面でモノリシックカーネルに劣るのは当然のことなのだ。

もっとも、CPUの動作クロックが1GHzを越えている現在では、少々の性能低下は、まったく問題にはならないだろう。

●Linuxはモノリシックカーネル

しかしLinus Torvalds氏が最初のLinuxカーネルを作り、インターネットを通じて世界中に発表した'91～'92年は、今とは事情が違った。x86 CPUの主力はi386であり、i486はまだ市場に出回っていなかった。それに彼がLinuxを作ったそもそもの動機は、「x86の勉強のため」だったのだ。Torvalds氏がLinuxを作る際に、実装が容易で、性能が出せるモノリシックカーネルを選んだのは、妥当な判断だったと言えるだろう。

だが、この時点で、モノリシックカーネルのOSを作ることに賛成していない人々もいた。その中でもっとも有名なのが、Andrew Tanenbaum氏だ。彼は、MINIXという教育用のOSを作ったことでも知られている計算機科学の教授である。Tanenbaum氏が、Net Newsのcomp.os.minixに投稿した“ LINUX is obsolete ”(LINUXは時代遅れ)と題した文書がきっかけで、かなり激しい論争が起きたのは有名な話だ。この論争の経過は、今でもインターネット上で検索すれば見つけられる。また「オープンソースソフトウェア」(オライリー・ジャパン)には、日本語訳が掲載されている。

モジュール

Linuxカーネルは、モノリシックではあるが、ローダブルモジュールがサポートされている。そのため、デバイスドライバや各種のファイルシステムに対応したドライバは、必要になった時点でメモリ上に読み込まれるように設定できる。多くのドライバを組み込んだ不必要に巨大なカーネルを利用したり、ハードウェア構成に合わせてカーネルを再構成したりせずに、いろいろなマシンに適したシステムを作ることが可能になっている。

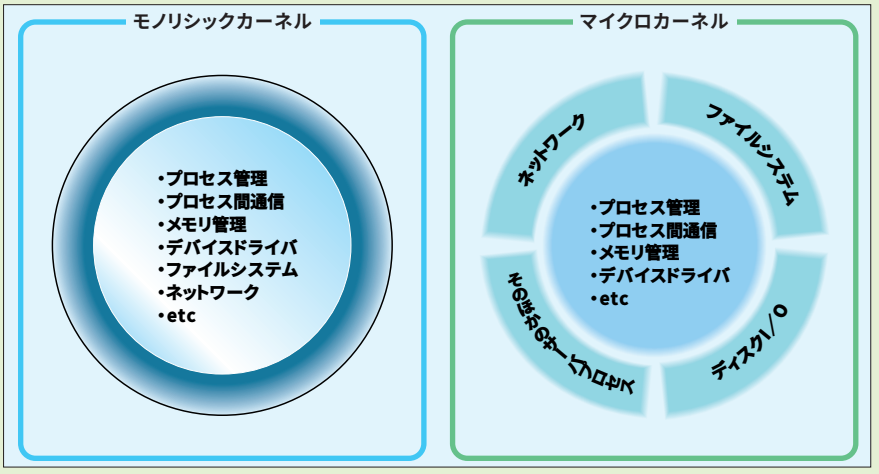
ローダブルモジュールが使えれば、新しいハードウェアを追加する際にいちいちカーネルを再構成する必要もなくなるため、「Linux対応」ハードウェアを用意しやすくなっていると言える。ハードウェアベンダーは、ローダブルモジュールとその組み込み方を説明したドキュメントを用意すればいいからだ。ハードウェアの追加にカーネルの再コンパイルが必須では、こうはいかなかっただろう。

Linuxカーネルの今後

「x86の勉強をするため」に作られ、小人数で使うつもりだったLinuxカーネルは、わずか10年で世界中に広まり、有力なベンダーによって採用されるに至った。

2.2から2.4への進化は、2.0から2.2の進化と比較して明らかに加速していることがわかる。世界中のプログラマーやベンダーの協力があればこそだろう。

Linuxの主要な部分は、ほぼ完成の域に達しており、大幅に変化することはないという意見もあるが、新しい技術を取り込みつつ、これからも進化は続いていくだろう。



モノリシックカーネルとマイクロカーネル
昔から使われているモノリシックカーネルに対し、マイクロカーネルは、より洗練された構造を持つ。

MkLinuxのWebサイト
PowerPC用Linuxの主力がLinuxPPCに移ったとはいえ、今でも開発は行われている。

カーネル2.4をインストールしよう

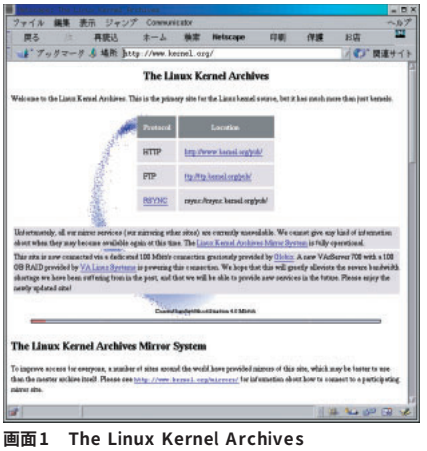
いよいよ今秋にもリリースされようかというカーネル2.4。ここまで紹介してきたように、あんな機能やこんな機能が満載である。満載であるからには、使わなきゃソンっていっても、アップデート方法を知らなきゃはじまらない。ということでカーネル構築のノウハウを解説していこう。

まずは下調べ

なにしろ、カーネルはオペレーティングシステムの根幹となる部分である。うっかり変な設定でインストールしようものなら、システムが起動しなくなる可能性だってあるのだ。そこで重要になってくるのが、事前の調査。何はともあれ、カーネルのソースに含まれるドキュメントから目を通していこう。

■カーネルソースを入手しよう

Linuxカーネルの配布元のご本家は、The Linux Kernel Archives（http://www.kernel.org/。**画面1**）である。



画面1　The Linux Kernel Archives

もちろん、世界中のLinuxユーザーのために各国にミラーサーバが用意されている。日本国内では、RingServer Project（http://www.ring.gr.jp/）の各サーバを利用するのがよいだろう。それぞれのURLは表1を参照のこと。

執筆時点での最新ベータバージョンは、8月10日にリリースされた2.4.0-test6であった（アルファバージョンとしてはtest7/pre6が最新）。今回はこのtest6リリースを使ってテストすることにしよう。今月号の付録CD-ROMに収録しているので、そちらも活用してもらいたい。

カーネルソースはtar＋gzipまたはtar＋bzip2形式で配布されている。とりあえずドキュメントを見るだけなので、どこのディレクトリに展開してもいいのだが、後々ことも考えてディストリビューションの設定に合わせることにした（標準の「/usr/src/linux」というカーネルソースのパスは、カーネルとは別に配布されるローダブルモジュールのコンパイル時にデフォルトで参照されることがある）。なお、ベースのディストリビューションにはLASER5 Linux 6.2を使用した。

既存のカーネルソースが収められているパス「/usr/src/linux」のlinuxディレクトリは「/usr/src/linux-2.2.14」のシンボリックリンクになっている。LASER5以外のディストリビューションでも、同じようなディレクトリ構成になっているはずだ。リンクなので「rm linux」として削除してもよいのかもしれないが、Linux自体を消すみたいで縁起が悪そうな気がしたため、一応リネームすることにした。

```
# cd /usr/src
# mv linux linux-save
```

続いてtarボールを/usr/srcにコピーして展開する。アーカイブ形式に合わせて以下のいずれかのコマンドを実行しよう。

```
# tar xvzf linux-2.4.0-test6.tar.gz
# bzip2 -dc linux-2.4.0-test6.tar.
bz2 | tar xvf -
```

/usr/src/linux以下にソースツリーが展開されるので、リネームしてリンクを作成する。これで、元のディレクトリ構成と同じ形になる。

```
# mv linux linux-2.4.0-test6
# ln -s linux-2.4.0-test6 linux
```

サイト	URL
カーネル2.4	
The Linux Kernel Archives	http://www.kernel.org/pub/linux/kernel/
RingServer Projects	http://www.ring.gr.jp/archives/linux/kernel.org/kernel/
modutil	
The Linux Kernel Archives	http://www.kernel.org/pub/linux/utils/kernel/modutils/
RingServer Projects	http://www.ring.gr.jp/archives/linux/kernel.org/utils/kernel/modutils/

表1　Linuxカーネル関連のURL

■README読むべし

展開先のlinuxディレクトリにあるREADMEファイルを読むと、linux/Documentationディレクトリ以下に各種ドキュメントが収められていることがわかる。また、Documentation/00-INDEXファイルに各ドキュメントの簡単な説明がリストされているので参考にしよう。

インストールに必要となるコンパイラやツールのバージョンは、Documentation/Changesファイルに記載されている。test6のChangesに載っていた最低限必要なバージョンは、**表2**のとおりだ。LASER5では、以下のとおりであった（pcmcia、PPP、ISDNに関してはテスト環境にないので省略）。

●gcc　2.91.66
●binutil　2.9.5

コンポーネント	バージョン	チェックコマンド
Gnu C	2.7.2.3	gcc --version
binutils	2.9.1.0.22	ld -v
util-linux	2.10g	chsh -v
modutils	2.3.13	insmod -V
e2fsprogs	1.18	/sbin/tune2fs --version
pcmcia-cs	3.1.13	cardmgr -V
PPP	2.4.0b1	pppd --version
isdn4k-utils	3.1beta7	isdnctrl 2>&1|grep version

表2　カーネル2.4のソフトウェア要件

●util-linux　2.10f
●modutil　2.3.10
●e2fsprogs　1.18

うーむ。微妙に推奨環境に合致していない。Changesには、それぞれのツールの配布元URLも記載されていて参考になるのだが、いちいちダウンロードしてコンパイルしていくのもなんだか面倒なので、とりあえず調査を進めることにしよう。

Changesを詳しく見たところ、util-linuxについては致命的な問題はなさそう。Cコンパイラ（gcc）は、2.95に「既知の問題」があり、pgccではカーネルのコンパイルに障害があると書かれている。2.91系については、特に触れられていない。さらに、カーネル開発のメーリングリストを検索したところ、Linus Torvalds氏もカーネルのコンパイルにgcc-2.91.66を使っているとのご本人の投稿を発見。とりあえず、gccのバージョンも現行のままでいくことにしよう（ただし、完全にステイブる）。実際に開発に携わる人々のメールなので、ハイブローすぎて理解不能なことも確かに多いが、なかには役に立つ話題もあるはず。本文中で紹介したLinusさんのメールもここで発見したものだ。

リンクサイトでは、LinuxLinks.comのカーネルのコーナー（http://www.linuxlinks.com/Kernel/）がお勧め。カーネル関連だけで、300近いサイトのリンクがある。

Column

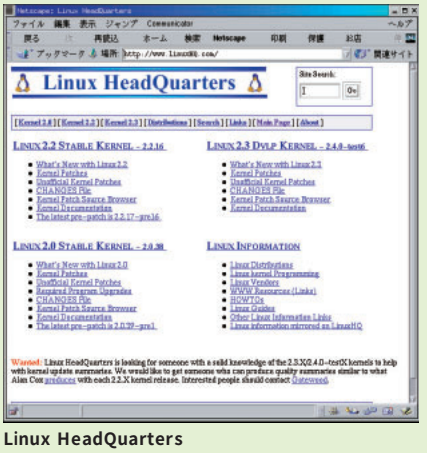
Linuxカーネルの情報サイト

もともとインターネットを通じて開発されてきただけに、Linuxに関する情報やドキュメントはネット上のあちこちにあふれている。カーネルについても、多くのWebサイトが専門に情報を発信している。

カーネル情報の発信サイトとしてはKERNELNOTES.ORG（http://kernelnotes.org/）が有名だが、現在、主な内容は「Linux HeadQuarters」に移行しているようだ。

Linux HeadQuartersは、安定版の2.0系／2.2系、開発版の2.3系（2.4.0-testXも含む）について、ソースの配布、ソースに含まれるドキュメントの情報整理（変更点などが公開されている）、パッチの整理と配布を行っている。また、バージョンごとの新機能や変更点がまとめてられているページなど、役に立つ情報がいっぱいだ。カーネル関連のサイトを集めたリンクも充実しているので一度訪れてみてほしい。

カーネル開発の状況をいちはやく知りたい人には、「Kernel Traffic」がお勧めだ。このサイトでは、カーネル開発のメーリングリストでのやり取りを1週間ごとにまとめて報告してくれる。各スレッドの内容を抜粋・要約して掲載しているので、おおまかなやり取りの様子がわかるようになっている。元のメールの内容にアクセスすることも可能だ（メーリングリストのアーカイブは前出のKERNELNOTES.ORGのものを利用している）。



Linux HeadQuarters
http://www.linuxhq.com/

Kernel Traffic
http://kt.linuxcare.com/kernel-traffice/

ルなのは2.7.2.3であるようだ。キチンとしたいヒトはこのバージョンを使ってネ)。

結局、modutilsのみをアップグレードすることにした。このツールは、カーネルソースと同じFTPまたはHTTPサイトで配布されている（URLは**表1**を参照)。RPMファイルも用意されているのでインストールは簡単だ。なお、バージョン2.3.13ではインストール後に「/etc/conf.modulesをmodules.confにリネームしましょう」というメッセージが表示される。忘れずに実行しておこう。

カーネルのインストール手順

READMEファイルには、カーネルをインストールする手順も記載されている。その記述に従うと、手順自体はいたって簡単で、以下のとおりにコマンドを実行すればよいだけだ。

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

「make xconfig」のところは、「make config」または「make menuconfig」としてもよい。configとmenuconfigはテキストベースなので、X Window Systemのない環境で有効だ。xconfigは、その名のとおりXベースのGUI操作でカーネル設定を行える。以降ここでは、xconfigの画面を使って説明を進めるが、インターフェイスは違っても選択項目はconfigでもmenuconfigでも共通している。

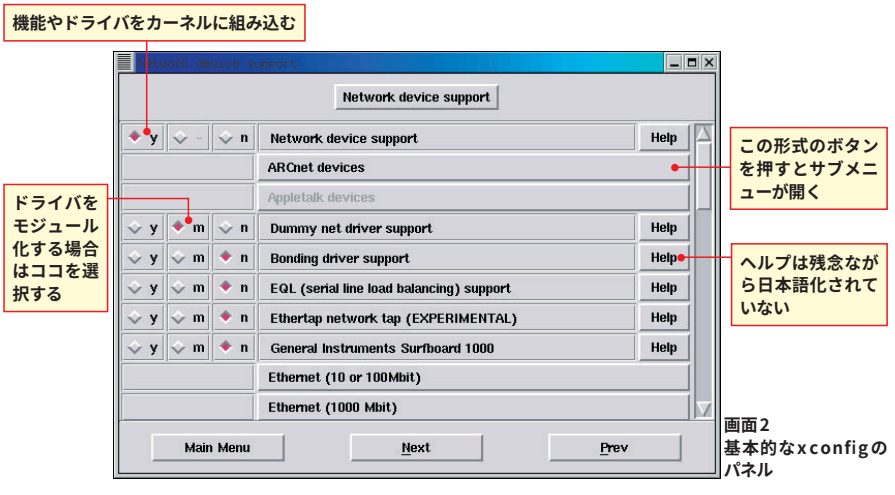
また、「make bzImage」の行は「bzdisk」または「install」を指定する方法もある。これについては、のちほど詳述することにしよう。

設定オプション

インストール手順の中でユーザーが判断して設定を行う必要があるのは、カーネルコンフィグレーション（make xconfig）の部分だ。ここでは、カーネルに組み込む機能（やドライバ)、ローダブルモジュールとしてシステムにインストールするドライバを適切に選択しなければならない。そのため、各オプションに関する知識や、ハードウェア構成の詳細をきちんと把握しておくことが必要になる。マザーボードのチップセット、ネットワークカードやビデオカードなどの構成を事前にチェックしておこう。

xconfigのインターフェイスは、基本的に**画面2**に示すようになっている。オプションの名称と、そのオプションを設定する［y］、［m］、［n］の各ボタン、それに［HELP］ボタンが並んだ直感的でわかりやすいインターフェイスだ。あたりまえだが一応書いておくと、［y］ボタンと［n］ボタンでオプションのオン／オフを選択する。［m］を選択した場合、そのドライバはローダブルモジュールとしてインストールされる。

項目数が非常に多いため、すべてを説明することはできない。ここでは、新しく追加された機能を中心に解説することにしよう。また、ハードウェア構成に依存する項目は、お手持ちの構成に合わせて設定してほしい。では、メニューの順に沿って各設定項目について見ていこう。



画面2 基本的なxconfigのパネル

■メインメニュー

まずはメインメニュー（**画面3**)。カーネルの機能には直接関係ないが、メニュー項目の分類が2.2系（**画面4**）とは異なっている。EthernetやAppletalkなど2.2系では、それぞれ単独のメニュー項目だったものが［Network device support］にまとめられている。逆に、［ATA/IDE/MFM/RLL Support］は［Block device support］から分かれて単独のメニュー項目となった。メインメニュー全体は、少しコンパクトになってすっきりした印象だ。

■Code maturity level options

アルファバージョンの機能やドライバを設定可能にするかどうかのオプション。「n」とすると、アルファバージ

ョンのものは選択不可（xconfigでは淡色表示）となる。先進的な機能を試してみたい場合には「y」に設定しよう。

■Loadable module support

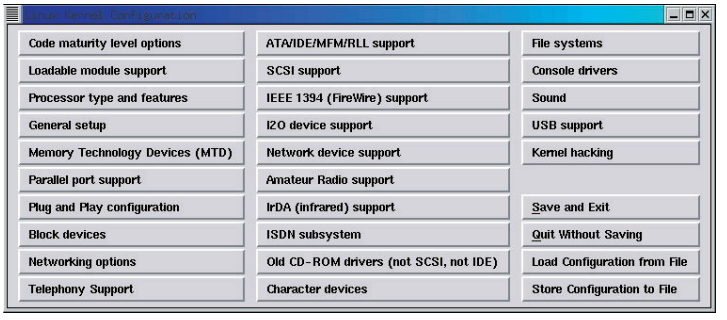
カーネルローダブルモジュールのサポートを設定する。すべての必要なドライバをカーネルに組み込んで、かつ将来にわたってハードウェアを追加することもないといった特別なケースを除いて、ほとんどの場合は3項目とも「y」にしておこう。「n」にする積極的な理由はあまりない。今回も、もちろんこのサポートを組み込んだ。

■Processor type and features

CPUの種類と機能を設定する。AMDのK6シリーズ／Athlon、Winchipなどのオプションが追加されている。

メモリサイズの上限も、ここで設定する。メニュー項目は［High Memory Support］だ。システムに搭載しているメモリが1Gバイトより少ない場合は「off」、1〜4Gバイトでは「4GB」、それ以上は「64GB」に設定する。パーソナルユースで1Gバイト以上のメモリを積むことはあまりないと思うので、どちらかといえば、サーバ用途や将来の拡張に備えた設定項目だろう。

このメニューパネルでぜひオンにしておきたいのがMTRR（Memory Type Range Register）機能のサポート。この機能を有効にすることで、PCIやAGPのビデオカードを搭載したシステムのグラフィック処理性能の大幅な向上が望めるのである。IntelのP6系CPU、AMDのK6シリーズ／Athlonなどが、この機能に対応している。



画面3
カーネル2.4のxconfigメインメニュー

■General setup

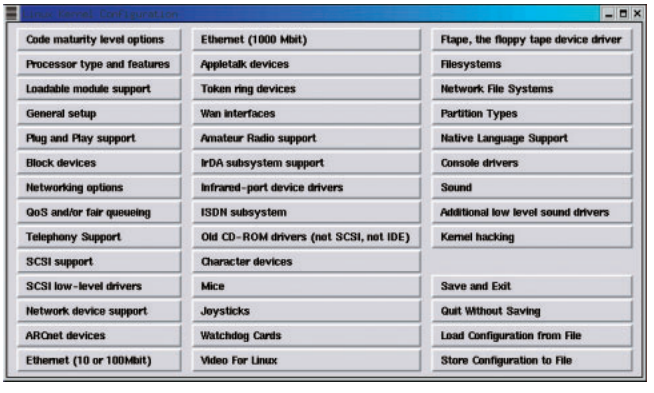
2.2.14にあったいくつかのPCI関連項目がなくなり、PCMCIA/PC-CardBusとACPI（Advanced Configuration and Power Interface）のサポートが追加された。パワーマネージメント機構としては引き続きAPM（Advanced Power Management）もサポートされている。これらの項目は、ハードウェア構成に合わせて設定しよう。

メニューパネルの中ほどにある［Kernel support for〜］で始まる3つの項目は、デフォルトの「y」のままにしておこう（**画面5**）。これらは、カーネルがサポートする実行ファイル形式に関する設定だ。ELF形式が標準だが、一部の古いコマンド、Java・Pythonなどのインタプリタ系の言語で書かれたプログラムの実行に必要となる。

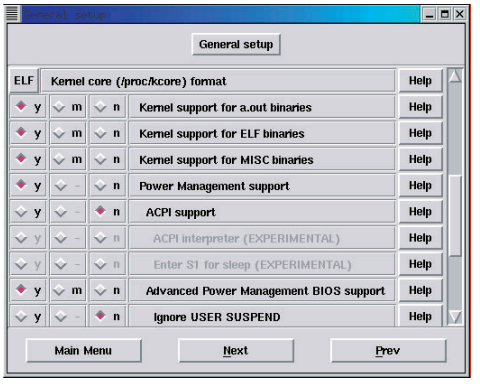
実はコンパイル後のテストで判明したのだが、もうひとつ［BSD Process Accounting］も「y」としておくほうがよいかもしれない。「n」にしてコンパイルしたカーネルでは、システム終了時に「Accounting 〜」というエラーが発生したのだ。ディストリビューションによって差異があるだろうが、xconfigのヘルプでも「Yにするのはグッドアイデアだね」と言っているので、そうしておこう。

■MTD、パラレルポート、PnP

これらのメニューにある項目はハードウェア構成に依存する。自分のシステムや将来の拡張などを考慮して設定しよう。ちなみに、MTD（Memory Technology Devices）はフラッシュメモリなどのメモリデバイスのI/O制御サポートの設定だ。また、PnPはデフォルトのまま、カーネルに組み込んでおくほうがよいだろう。



画面4
カーネル2.2.14のxconfigメインメニュー



画面5　［General setup］パネル
実行ファイル形式のオプションは3つとも「y」にしておくほうがよいだろう。また、ACPIのサポートは一部「EXPERIMENTAL」となっている。現行ではAPMのほうが無難かもしれない。

■Block devices

これも使用するハードウェアに依存するメニューだが、カーネル2.4の新機能として注目のLVM（Logical Volume Manager）が含まれている。詳しくは新機能紹介のコーナーを参考にしてほしいが、LVMは複数の異なるハードディスク上にあるパーティションを1つの「論理ボリューム」として操作するためのメカニズムを提供する。かなり使えそうな機能なので、将来の拡張に備えて、カーネルに組み込むかモジュール化しておくとよいだろう。

ソフトウェアRAIDの設定も、（必要に応じて）このメニューパネルで行う。

■Networking options

ネットワーク関連の設定項目での最も大きな変更点は、パケットフィルタリングやプロキシ、IPマスカレードなどのファイアウォール関連の機能が［Network packet filtering］として一本化された点だ。

ネットワーク関連の設定はマシンの用途によって異なるので、各項目を個別に説明することは避けるが、基本的にクライアントとしてLAN環境やインターネットに接続するだけなら、デフォルトの設定でOKだ。サーバ用途の場合は必要な機能を十分に検討してから、設定を行うようにしたい。

■Character devices

コンソール、シリアルポート、キーボード、マウスなどのデバイスを設定する。また、今回テストに使用したIntel810（i810）チップセットに関する設定項目も含まれている。

i810に統合されているグラフィック機能をX Window System（XFree86）で利用するには、［/dev/agpgart（AGP Support）］と［Intel i810/i815 support］を「y」または「M」としてセットアップする必要がある。項目の名称からもわかるように、i810の後継チップセットであるi815もすでにサポートされているようだ。

XFree86 4.0で導入されたDRI（Direct Rendering Interface）のカーネル側のサポートについても、このメニューパネルに項目が用意されている（**画面6**）。3dfx Voodoo3、3dlabs GMX 2000、ATI Rage128、Matrox G200/G400といったビデオカードに加え、i810のビデオ機能もサポートされている（せっかくなので有効にしておいた）。

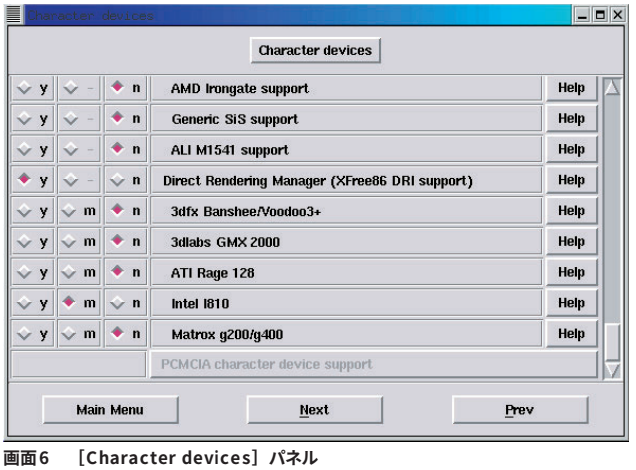
■File systems

標準のファイルシステムは、引き続きext2で変更はない。このメニューパネルでの注意点は、デフォルトで「n」になっている［ISO 9660 CDROM file system］を忘れずに「y」にしておくこと。たいていのLinux用CD-ROMは、ISO 9660フォーマットなので（本誌の付録CD-ROMもそうだ）、ほとんどのケースで必要になるはず。FATのサポートもデフォルトでは有効にならないので、必要に応じて設定しよう。Windows 9x用のFAT32ならVFATサポートを有効にすればいい。

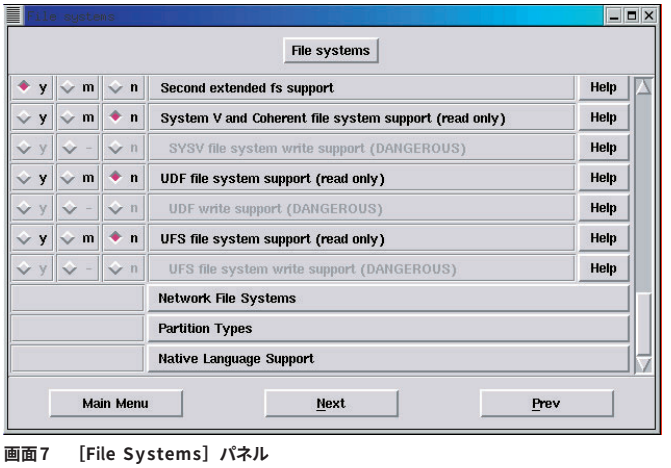
もう1つ注意してほしいのが、メニューパネルの最下部にある［Network File System］と［Native Language Support］以下の各設定項目だ。NFS、SMB、NCPなどを使用する場合は忘れないようにしたい。NFSはオプションながらバージョン3がサポートされるようになった。また［Native Language Support］では、ファイル名やディレクトリ名などファイルシステムで使われる言語として、シフトJISとEUC-JPが新たにサポートされている（**画面7**）。

■その他の項目

ここまでに紹介しなかったメインメニューの項目は、ハードウェア依存の



画面6　［Character devices］パネル
カーネル2.4では、XFree86 4.0の新しい機能「DRI（Direct Rendering Interface）」のサポートが追加された。



画面7　［File Systems］パネル
最下段にある［Network File Systems］と［Native Language Support］も忘れずにチェック。

ものだ。今回はテスト環境に合わせて、［sound］の［Intel ICH audio suport］と［USB］の［USB Human Interface Device（HID）support］、［Keyborad support］、［Mouse support］を有効にした（**画面8**、**画面9**）。それぞれの環境に合わせて機能を選択するようにしよう。

コンパイルとブートローダ設定

コンフィグレーションがひと通り終わったら、メインメニューの［Save and Exit］ボタンをクリック。xconfigが終了し、「make dep」することを求めるメッセージが表示される。

「make dep」が完了したら、次はいよいよコンパイルだ。コマンドは「make bzImage」。このオプションでは、コンパイルのみが行われる。生成されるカーネルイメージのパスは「/usr/src/linux/arch/i386/boot/bzImage」となる。続いて、「make modules」と「make modules_install」を実行すればコンパイルは完了だ。

先に述べたように、bzImageとは別に2つのカーネルコンパイルオプションがある。bzImageの代わりに「make bzdisk」とすると、フロッピーディスクにカーネルイメージがインストールされる（実行の際にはフォーマット済みのフロッピーをセットするのを忘れずに）。この場合にも、先ほどのパスにカーネルイメージが置かれる。

もう1つの方法が「make install」とする方法だ。このオプションでは、コンパイル後に/bootディレクトリに新しいカーネルイメージがvmlinuz-2.4.0-test6としてコピーされ、シンボリックリンク/boot/vmlinuzが更新される。本来はこれで、すぐに新しいカーネルで起動できる状態になるはずなのだが、/etc/lilo.confで以下のように実体ファイルが直接指定されている（LASER5の場合）ため変更が必要となる。

```
image=/boot/vmlinuz-2.2.14-LL2
```

この行を「image=/boot/vnlinuz」に変更し、liloコマンドを実行すれば、新しいカーネルイメージから起動するようになる。

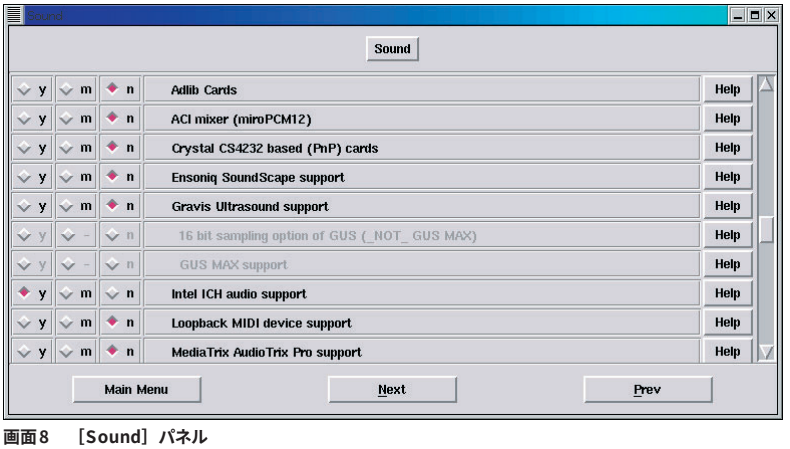
今回はテストバージョンのカーネルなので、bzImageオプションでコンパイルし、liloを手作業で設定した。新しいカーネルで正しく起動できない可能性もあるため、元のブートオプションを残しておきたかったからだ。liloの設定方法は以下のとおり。まず、コンパイルされたカーネルイメージを/bootにコピーする。

```
# cd /usr/src/linux/arch/i386/boot
# cp bzImage /boot/bzImage
```

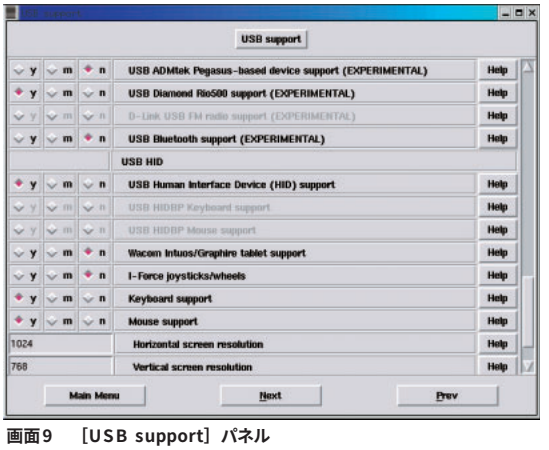
次に/etc/lilo.confを**リスト1**のように修正。こうしておけば、起動時のboot:プロンプトでカーネルを選択できる。最後に「lilo -v」を実行してliloをインストールする。liloのインストールは結構忘れがちなので気をつけてほしい。

▼リスト1　lilo.conf

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

image=/boot/vmlinuz-2.2.14-
12LL1
        label=linux-old          ← 追加
        read-only
        root=/dev/hda2

image=/boot/bzImage
        label=linux              ← 追加
        read-only
        root=/dev/hda2
```



画面8　［Sound］パネル
サウンドに限らず、ハードウェア構成に合わせたオプションの設定が重要だ。事前に環境を再確認しておこう。



画面9　［USB support］パネル
USBのサポートもカーネル2.4の新しい魅力だ。今後も利用可能なデバイスが増えていくはず。期待しよう。

カーネル2.4の新機能を試す

ながながとカーネル構築の手順を説明してきたが、実際にはオプションを選んでコマンドをいくつか実行するだけで、それほど大変な作業ではない。みなさんもぜひ挑戦してほしい。ここからは新機能の検証を進めていくことにしよう。

ホントに2.4？

まずは、システムの再起動。ベータバージョンということで少し心配したのだが、今回のテストでは、やや拍子ぬけするくらい無事に起動することができた。カーネル2.4で動作していることを確かめるには、「uname -r」とする。「2.4.0-test6」と表示されればOKだ。

dmesgコマンドで起動時のメッセージを確認してもいい。ブートが開始された最初の行にカーネルのバージョンが表示されているはずだ。メッセージを追っていくと、デーモンやデバイスドライバの起動時にエラーが発生していないか確認できる。/var/log/boot.logにも起動時のメッセージが記録されているので確かめてみよう。

Xは動作する？

アップグレード前の環境がきちんと動作するか検証するために、XFree86を起動してみた。あわせて、新機能であるi810のサポートも検証できるはずだ。XFree86のバージョンは、LASER5 6.2の標準の3.3.6。もちろんアップグレード前は正常に動作していた。modprobeコマンドでagpgartをロード。Xを起動する。ダメだ。「GARTのI/Oエラーだぁ」というフェイタルなメッセージがぁ！

このあと、agpgartのバージョンをあれこれ変えてみたり（Intelのサポートページにあるバージョンや2.2系カーネルのもの）、Intelが提供しているi810対応のXF86-SVGAサーバなどを試したり、XF86-SVGAサーバに適当にパッチをあててみたり、といろいろチャレンジしたが全部ダメ。結局、XFree86を4.0にすることで解決した。

解決したとはいっても、元の環境に戻すにはGNOMEの再インストールが必要なのである。「やっぱり安易にカーネルのアップグレードはできないなぁ、原稿には『できるだけテスト用の環境を別途用意したほうがいいだろう』って書いておこう」などと思いつつ、ふと思い立って440BXマザー＋Matrox G200という構成のマシン（たまたまLASER5が入っていた）で試みたところ、やけにあっさりとXが起動した。このあたりは、環境によりけりというところなのだろう（か？）。

ややなさけないが、結論としては、i810ではXFree86のバージョンに注意。Linux定番のビデオカードなら、たぶん大丈夫でしょう。という感じだ。i810に関して付け加えておくと、サウンド機能（AC97）のほうは問題なく動作している。確認はxmixerの音楽CD再生で行った。

USBデバイスを試す

続いて、Xの起動で機嫌のよかった440BXマシンでUSBデバイスの接続をテストした。このマシンのUSBホストコントローラはUHCIだ。テストには、LogitechのiTouch Keyboard（**写真1**）とサンワサプライのUSBモバイルマウス（**写真2**）、矢野電器のUSB接続フロッピードライブUFD-04（**写真3**）を使用した。

キーボードに関しては特に問題なし。接続して再起動するだけで、認識され正しく動作した。ただし注意点がひとつ。boot:プロンプトの段階ではLinux側で認識されていないため、カーネルのラベルやパラメータを入力できない。この問題はBIOS設定でUSBポートをPS2エミュレーションにしておけば解決できる（BIOSがキーボード処理を行ってくれるのだ）。

写真1　Logitech iTouch Keyboard
USB製品にはなぜだか、洗練されたデザインのものが多い。

写真2　サンワサプライのUSBマウス
本来はモバイル用のマウスだ。コンパクトで操作性も良い。

マウスはどうだろうか。Xを起動してみたところ、マウスカーソルが動かない。カーネルソースの付属ドキュメントにUSBに関連するものがあったことを思い出し、確認してみることにした。USBマウスの設定が記載されているドキュメントは、カーネルソースツリーのDocumentation/usb/input.txtであった。以下、このドキュメントの記述にしたがって設定を説明しよう。まずは、USBマウス用のデバイスファイルを作成する。

```
# mkdir /dev/input
# mknod /dev/input/mice c 13 63
```

Xで使用するにはXF86Configの変更も必要だ。Pointerセクションにある「Device」の設定を、作成したデバイスファイルを指定するように変更する。

```
Section "Pointer"
    protocol "PS2"
    Device   "/dev/input/mice"
```

これでX上でマウスが使用できるはず。実際のテストでも、問題なく動作した。

ここまでは順調だったが、最後のUSBフロッピードライブで問題が発生。結論から言うと、デバイスの認識とマウントまでは問題ないが、リード／ライトができないのだ。ext2フォーマットのフロッピーでは、lsコマンドもcpコマンドも利かない。さらに不思議だったのが、FAT32フロッピーはリードが可能であること。

写真3　矢野電器のUSB接続FDD
パッケージにはDOS/V・Mac用と記載されていた。Linuxは？

なんだか判らないので、カーネルコンフィグ（xconfig）でUSBサポートにある［USB verbose debug messages］を「y」にして再コンパイルすることにした。これで、USBデバイスに対する操作に関してカーネルメッセージが表示されるようになる。デバイスが正しく認識・動作しているか確認できるので、接続テストに有効だ。問題を切り分けわれるよう、［UHCI support］と［USB Mass Storage support］もモジュール指定にした。

フロッピーをセットし、「modprobe usb-uhci」を実行する。2つのUSBポートを検出したことを示すメッセージが出されるだけで問題はないようだ。「modprobe usb-storage」とすると、ベンダー、モデルなどを示すメッセージが表示される（**リスト1**）。ただ、ext2フォーマットのフロッピーでは最後の1行が「パーティションが見つかりません」というふうに出力された。FAT32の場合は、パーティションが認識できている感じだ。

マウント／アンマウント、モジュールのロード／アンロードを繰り返すうちに、次々にエラーメッセージが表示され、カーネルパニックでシステムが停止することもあった。あらためてXconfigの画面を眺めていると、［File systems］の下のほうに［Partition type］という項目がある。このメニューにあるMS-DOS形式のサポートを有効にしたところ、FAT32フロッピーではリード／ライトできるようになった。

さらにフロッピー関連の項目はないか探してみたが、これといったものは見つけられなかった。USBサポートプロジェクトのWebサイト（http://linux-usb.sourceforge.net/）も覗いてみたが、これといった情報は得られず。締切りまでに原因が究明できなかったのが非常に残念だ。ベータバージョンということで、どうかお赦しいただきたい。

Devfsの導入に挑戦！

カーネル2.4の重要な新機能のひとつであるDevfs（Device File System）も試してみた。前述のとおり、Devfsはブロックデバイスとキャラクタデバイスの管理・制御方式を一新する。その影響はシステム全体に及ぶ。導入には細心の注意が必要だ。カーネルソースに付属のドキュメント（Documentation/filesystems/devfs/README）をよく読んで導入のガイドラインとしてほしい。

Devfsのサポートは、カーネルコン

▼リスト1　usb-storageロード時のメッセージ（FAT32フロッピーの場合）

```
scsi0 : SCSI emulation for USB Mass Storage devices
Aug 27 06:03:03 lmdvd kernel: scsi : 1 host.
   Vendor: TEAC      Model: FD-05PUB          Rev: 1026
   Type:   Direct-Access                      ANSI SCSI revision: 02
Detected scsi removable disk sda at scsi0, channel 0, id 0, lun 0
                                  ⋮
SCSI device sda: hdwr sector= 512 bytes. Sectors= 2880 [1 MB] [0.0
GB]
sda: Write Protect is off
sda:<7>usb-storage: queuecommand() called
sda1 sda2 sda3 sda4
USB Mass Storage support registered.
```

フィグの［File systems］の［/dev file system support］を［y］にすることで有効になる。関連オプションの［Automatically mount at boot］は念のためオフにしておく。

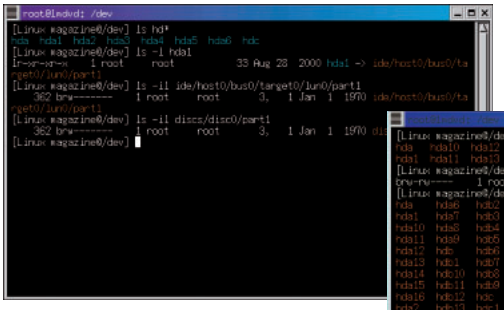
コンパイルの完了後、Devfsの制御用デーモンdevfsdをRichard Gooch氏のWebサイト（http://www.atnf.csiro.au/~rgooch/linux/）からダウンロード。コンパイルは簡単で、「make」、「make install」とするだけだ。READMEによると、devfsdはブートの早い段階で起動しておく必要があるようだ。/etc/rc.d/rc.sysinitの先頭付近に以下のコマンドラインを追加しよう。

```
/sbin/devfsd /dev
```

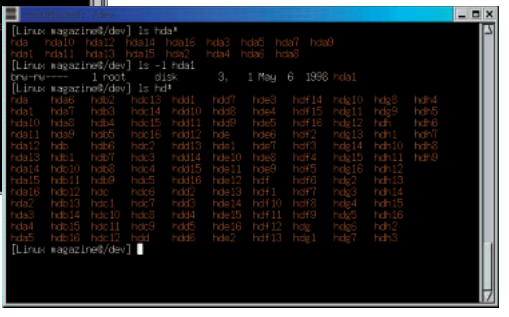
システムで使用しているソフトウェアのバージョンにも注意が必要。Cライブラリは、glibc 2.1.3以降が望ましい。また、XFree86は4.0がベターであるようだ（3.3.6使用時の設定についてはREADMEを参照）。このほかデバイスファイル関連のいくつかの設定を変更する必要がある。たとえば、Devfsでは/dev/ptsは管理しないため、/etc/fstabにあるこのデバイスファイルの設定をコメントアウトしたほうがよい。その他、設定の詳細はREADMEを参照してほしい。

準備が整ったところで、Devfsを起動してみよう。boot:プロンプトで「devfs=mount」というパラメータを与えればいい。今回のテストでは、特にエラーもなくブートが完了した。devfsdが従来の方式との互換性を維持してくれるので、起動に問題が発生することは少ないようだ。

Devfs動作後の/devを**画面1**に、従来の/devを**画面2**にそれぞれ示してある。



画面1　Devfsを導入した後の/Dev
少しわかりづらいかもしれないが、hda1などの従来のネーミングのデバイスファイルはシンボリックリンクになっている。



画面2　従来の/devデバイスファイルシステムには存在しない、hda7以降のパーティションやhdb、hddといったデバイスファイルがあらかじめ作成されている。

画面からわかるように、実際には存在していないhdbなどのデバイスはなくなっている。ネーミング規則が変更されているのに「hdaX」が残っているのは互換性維持のため。これらは実際には/dev/discs/discなどへのリンクで、devfsdが起動時に自動生成する。

限られたテスト時間の中では、システムに対して行った操作も限定されるため、「導入しても大丈夫」とは言い切れない。カーネル自体がベータテスト段階でもあり、やはり慎重を期したいところだ。ただし、少なくともテスト中に、Devfsが原因と思われる問題が起こることはなかったことをご報告しておく。

Column

「2.4は見たいけど導入するのはちとダルイ」そんなあなたにCalderaテクノロジープレビュー

既存のLinuxシステムへカーネル2.4を導入する手順は本文で紹介したとおりだが、「カーネル再構築に失敗してLinuxが動かなくなったらどうすればいいの？」と不安に思う読者もいることだろう。

■カーネル2.4を採用

そこで、手軽にカーネル2.4を使ってみたい読者にお勧めなのが、付録CD-ROMにも収録した「Caldera Linuxテクノロジープレビュー」（以下Caldera LTP）だ。

Caldera LTPは、Caldera Open Linuxでおなじみの Caldera Systems Inc.が、カーネルをはじめ、XFree86やglibcなど、システムの基幹部分に現時点での最新バージョンを盛り込んで、主に開発者へ提供することを目的としたLinuxディストリビューションだ。ベータ版的な性格が強く、Caldera社のサイトからのダウンロードという形でのみ配布されている。

■Caldera LTPが提供する最新の環境

付録CD-ROMに収録したCaldera LTPは、Linuxカーネルにバージョン2.4.0-test3を採用している。Open Linuxでもおなじみの GUIインストーラLizerdで手軽にインストールが行え、カーネル2.4が使える環境をすぐにも整えられる。

Caldera LTPではカーネル2.4のほかにも、XFree86 4.0.1、glibc 2.1.91（glibc2.2ベータに相当）、KDE 2.0 preview versionなどが採用されている。どれも現在Linuxユーザーの注目を集めているものばかりだ。これらすべてを別々に入手して、きちんと動作させるには大変な手間がかかるだろう（それが楽しかったりもするのだが）。

あくまで開発者向けとはいえ、最新の環境をディストリビューションの形で手に入れることができるのはうれしいかぎりだ。Caldera LTPは「最新版のおっかけ」ユーザー待望のディストリビューションといえるだろう。

Caldera LTPのWebサイト
http://www.calderasystems.com/preview/

2台から始める Linuxクラスタ

PCの性能向上は非常にめざましく、現在のPCでは速度に対する不満はほとんどないだろう。しかし、シミュレーションなどの科学技術計算の分野や、Webサーバ、データベースシステムなどでは、より高い処理能力が要求される。そのため、複数のコンピュータを使用して分散して計算させることで、高速なシステムを構築するクラスタリング技術が実用化されている。
Linuxでクラスタシステムを実現するソフトウェア（ディストリビューション）も増えてきている。そこで、今月から3回に分けて一般ユーザーでもできるクラスタリングを紹介する。まず今回は、クラスタシステムの構成としくみを解説する。

文：川村 智／日立エンジニアリング（株）コミュニケーションシステム部
Text : Satoshi Kawamura / Hitachi Engineering Co
Photo : Takashi Shinoha (Dee)

ある夫婦の会話その1

家に帰るとごはんがなかった

夫：ただいまぁ～
妻：お帰りなさい？
夫：おっPCでなにやっているの？
妻：内緒！ねえ、もっとこのパソコン速く動かないの？
夫：どうしたの？　このあいだシミュレーションゲームソフト作りたいって言うからPentium III 800MHzに変更したばかりじゃないか？
妻：だってCGのレンダリングをやらせると簡単なモデリングでも数分、地形描画や光源処理させると数時間かかるのよ～
夫：ゲームソフトのツールは簡単にゲームが作れるアスキーのシミュレーションツクールを使っているはずだったけど、そんなに処理が重いの？
妻：違うの。リアルな絵を書くのが苦手なので、ゲーム用の地形図を3Dグラフィックスソフトで作ったら延々と処理が続いて終わらないのよ。山や木々の描画にすっごく時間がかかるの。
夫：知らないあいだにすごいソフトを使いこなしているんだね。
妻：エライでしょ！
夫：見なおしたよ（うぅ勝手にこんなCGソフトを買っていたのか……）。
妻：将来Pentium 10GHzが発売されたら、衝動買いしちゃうかも。
夫：ブツブツ（すでにCGソフトを衝動買いしているじゃないか！）
妻：だってこのCGソフト遅くて、描画するのを眺めていたら、夕飯のおかずを買ってくるのを忘れてしまった！
夫：ひぇ～。今から近くのスーパーでなんか買ってこいよ！
妻：実はお米もないのよ。私1人では持てないから、あんたも付き合ってよ。
夫：ブツブツ（確かに君はモテるタイプではないネ！）
妻：なにか言った？
夫：いやなにも言っていないよ！
（注：当然この会話はすべてフィクションです）

■より速く計算するために

そもそも人間がコンピュータを発明したのは計算能力を向上させることにあった。

現代のコンピュータ（ノイマン型）の原型ができた当時は弾道計算などの軍事目的によく使われていた。

応用技術が進んだ現在ではさまざまな分野でコンピュータが使用されているのはいうまでもない。コンピュータがまったく利用されていない産業を見つけるほうが困難である。

最近では農業にもコンピュータが大いに利用されているが、たとえコンピュータを利用していない農家があったとしても、彼らの重要な情報源である天気予報も、背後にスーパーコンピュータが膨大な情報を処理して気温や雨量の予測をしている。農作物を豊作にするためにもコンピュータは陰で支援しているのだ。

このように現代社会は間接的または直接的にコンピュータに依存せざるを得ない構造になっている。

特にハードウェアの進歩は著しいものがあり、身近なパソコンでさえ1GHzクロックCPU搭載のマシンが買えるようになった。筆者が初めて購入したPCに搭載されていたCPUがZ80だった時代と、比較すると隔世の間がある。

ソフトウェアの進歩も著しいものがあり、左の夫婦の会話にもあったように複雑な操作をしなくても、3Dグラフィックスが描画できるソフトが各社から発売されるようになった。

しかしこの恵まれた時代でも、より高速に処理をしたいというニーズは減るどころか増える一方である。

この身近な例として3Dレンダリング処理がある。三次元的に物体を配置し、光の屈折率や減衰率を計算して写真のようなリアルな描画を求めると最速クラスのPCでも延々と時間がかかるのが実態だ。

もし10GHzプロセッサなんていうのが秋葉原のショーケースに飾られていたら、クラッと一瞬記憶喪失になって、気がつくと領収書と共にCPUを握り締めている輩が続出するだろう（それを真の衝動買いともいうのだ）。

■10GHzの世界

10GHzとは1秒間に10の10乗、すなわち100億という途方もない周波数だ。1クロックで1命令の処理ができれば、1秒間に100億回の演算ができるのだ。

この1クロックに光がどれだけ進むことができるか計算してみよう。光の速度は約30万km／秒で、これを100億で割ると、

300,000,000,000mm（30万km）÷10,000,000,000（10GHz）＝30mm

しか進まないのだ。

電気回路の電子は光の約0.6倍程度しか進まないので、1クロックでたった18mmしか進まないのだ。これが高速プロセッサの開発を困難にしている最大の理由である（と同時に、ものすごい発熱やノイズ対策も必要になる）。

もはやこれ以上のCPU性能は期待できないのだろうか？

ある夫婦の会話その2
スーパーに行こうとして

夫：ということで現在の技術では1GHz以上の高クロックのCPUを作るのがだんだん困難になってきているのさ。
妻：でも複数のCPUを使えばもっと速く処理できると思うけど。
夫：そのとおり！スーパーコンピュータも実は多くのCPUから構成されているのさ。
妻：わたしのPCもスーパーコンピュータみたいにならないの？
夫：そのためにはクラスタという技術を使う必要があるのだけど……。
妻：本当！もっと速くする方法があるの？　クラスタというのはペルチェクーラと水冷システムを組み込んでオーバークロックして、そんでもって放熱対策としてPCの背面にラジエータを取り付けて自動車のエンジンみたいな外観にするってこと？
夫：違うよ！どこでそんな知識を得たの？　とてもふつうの主婦の話とは思えないぞ！
（今までの会話もふつうではないぞ！）
妻：だって、あなたの買っているPC雑誌にオーバークロック特集とか書いてあって、てっきりその話かと……。
夫：ブツブツ（あぁ！おれが仕事をしているあいだにPC雑誌を読んでいたとは。あの○○マガジンは隔離せねば……。これはある種の院内感染状態と同じだ！）
妻：ブツブツ言っていないで、ちゃんと教えてよ！10年も一緒に暮らすた仲でしょ。
夫：プッ！それをいうなら「暮らした仲」だろ？**（彼女に座布団1枚！）**

まぁとにかくクラスタシステムを簡単に説明すると、複数の計算機をひとかたまりにして利用者からはひとつの計算機のように利用する技術なのだ**（もちろん設計側は複数の計算機を意識するヨ！）**。

■いろいろあるクラスタの形態

人間が複数のコンピュータを使って、クラスタシステムを構築する目的として、大きく（1）信頼性重視と（2）性能重視の2つがある。本音は両方ほしいところだ。

「いや、オレは複数の計算機を使ってジョークを飛ばせるほどの人工知能を実現しようと研究中なんだ！」という人もいるかもしれないが、まあこれも膨大な音声データを処理して認識するための性能を追求する一形態ともいえよう。

1.信頼性向上クラスタ

リライアビリティクラスタといったほうが筆者にはすっきりするのだが、英語に馴染みのない方々から舌が噛みそうだといわれて日本語で表現した。

要はダウンしては困るために複数のコンピュータを用意してひとつのサーバのように見せるクラスタである。

信頼性向上クラスタにはいろいろな形態が発表されている。大別してさらに2つに分類できる。

1.1フェイルオーバークラスタ

2台のコンピュータでディスクをSCSIケーブルやファイバーチャネルにより共有する形態である。共有ディスクにはたいていRAIDシステムが使用される。最もシンプルなクラスタである（図1-1）。

常に主系となっているコンピュータが処理を行い、待機系のコンピュータは主系のコンピュータを監視している。主系がダウンしたと判断したときは、待機系が主系としてサービスを肩代わりする（図1-2）。動作上の見地からはこの形態をホットスタンバイ形式と呼んでいる。

共有ディスク（RAIDシステム）がダウンしたり、ディスクとホスト間のケーブルが切断するとシステムが全滅する弱点がある。高級RAIDモデルにはケーブル断でも全滅を免れるため、SCSIコントローラやファイバーチャネルが二重化されているものがある。

ハードディスクに障害が起こった場合、自動的に予備のドライブに切り替わり、ホットスワップによって不具合のあるドライブを交換できるようにしておく。

フェイルオーバークラスタは、信頼性重視で1台の計算機の能力しか使わないため、コストパフォーマンスがかなり悪く、単一計算機システムの半分以下となる。2台の計算機のほかに高級なRAIDシステムやクラスタソフトがさらに追加となるためだ。さらにマシン設置場所にUPSが設置されていないとUPSの導入も必須となる。重要なシステムでは電源系統二重化＋サーバ専用UPS構成が望まれる。いくらサーバを二重化しても電源断でシステム全滅となるからだ。

1.2スケーラビリティクラスタ

クライアントからのサービスを複数のコンピュータで処理する。目的はトランザクション処理能力の向上である。

多くのクライアントからの処理要求があり、しかも運用後に増加すること

妻：でもなんだか難しそうね。

夫：そうでもないよ。クラスタの類義語にグループという言葉があるだろう？　意味的には同じだよ。

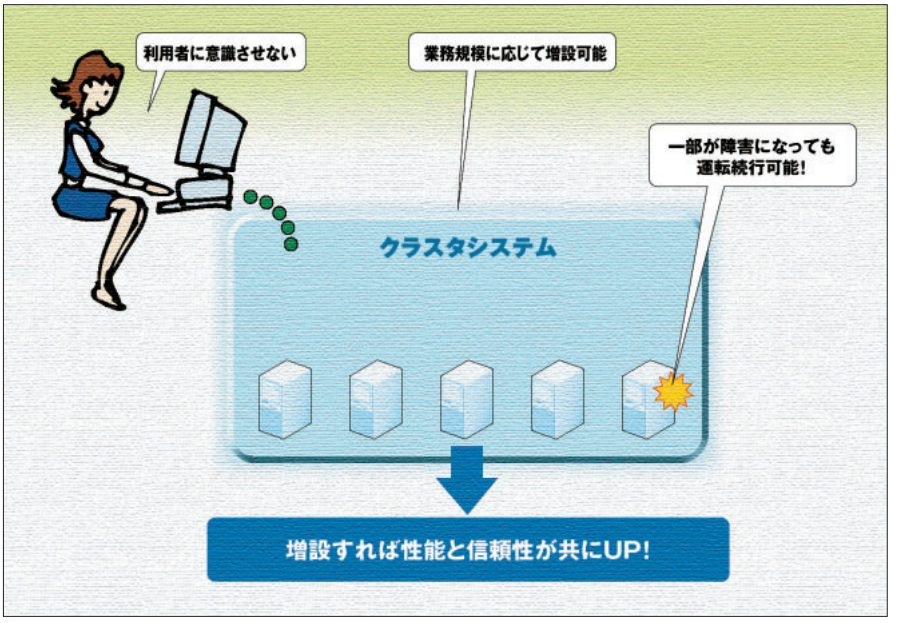
妻：ではどうしてグループではなくてクラスタというの？

夫：英語（欧米）では「人」が中心になって共通の目的を持った集団を「グループ」というのさ。

日本でも○○企業グループとか、暴走族グループなどの表現を使うけど、人が活動の中心にいるだろ？

妻：うんうん。

夫：ところが計算機のような「モノ」が活動の中心となってひとつの集合体



がわかっている場合には最適なクラスタである（図2）。

先ほどのフェイルオーバークラスタでは1台分のコンピュータしか能力を発揮できないが、このクラスタタイプならコンピュータを増設すればするだけ多くのクライアントからの要求に対応できる。日本でLinuxの商用クラスタとしてはTurboLinuxのCluster Serverという製品が最初に出荷され、その後、他のディストリビューターからも次々と出荷されている。

このクラスタの場合、メッセージを分配しトラフィックを制御する機能を持ったノードと、メッセージを処理する機能を持ったサービスノードに分けることが多い。メッセージを分配するノードがダウンするとシステムダウンとなるので、二重化するのが一般的だ。もちろん同一マシンに2つの機能をセットすれば、最低2台構成から可能である（図3）。

クラスタを構成しているコンピュータ（以後ノードと呼ぶ）がダウンすると自動的に障害を検知し、異常になっ

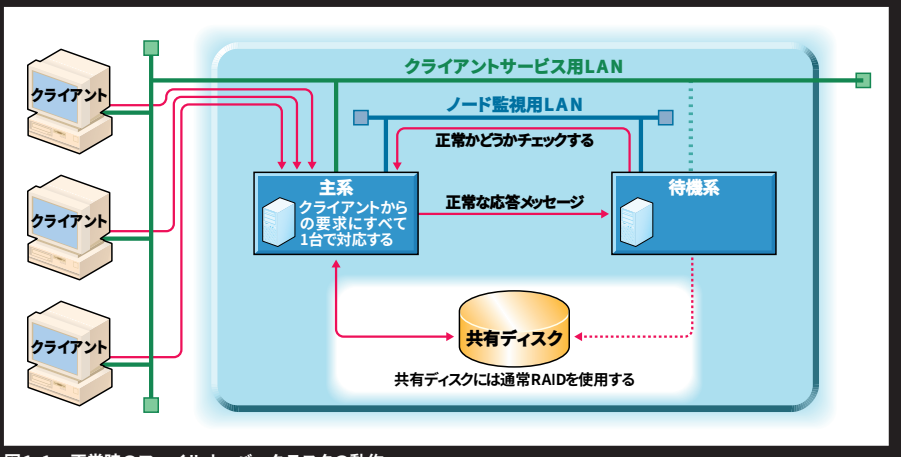


図1-1　正常時のフェイルオーバークラスタの動作
主系のコンピュータがすべての処理を行う。待機系は主系が正常に動作しているか監視している。

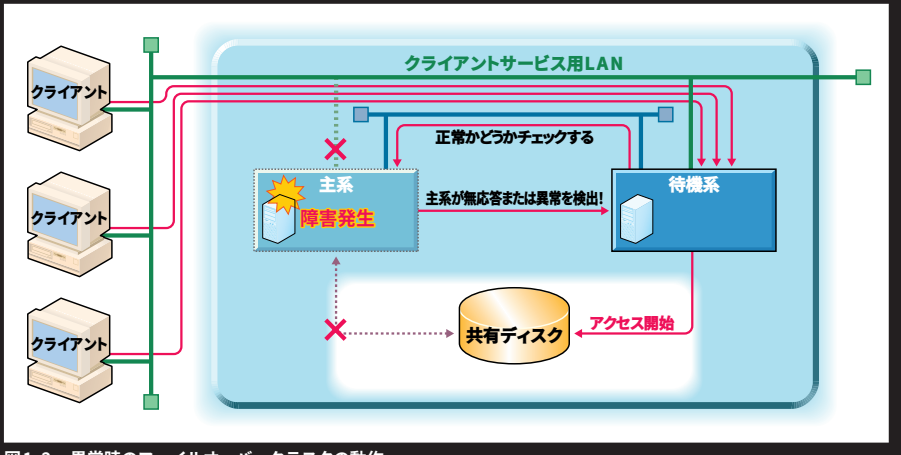


図1-2　異常時のフェイルオーバークラスタの動作
主系に障害が発生した場合には、待機系が代わりにクライアントへサービスを提供する。

として機能する場合、クラスタといって使い分けているのさ。もちろん例外はあるけどね。

「クラスタ」の身近な使用例を挙げてみると、PCが異常終了して再立ち上げすると、『ハードディスクのXXクラスタの○○ブロックが破損しています』という意味のメッセージが出るだろ？

妻：うん。危ないのですぐ修復するけど。

夫：これはハードディスクの中もデータブロックを何個か集めて、同じようにクラスタと呼んでいる一例だよ。これもひとつのアクセス単位として管理されているからなんだ。

妻：ふうん。とにかく複数PCをクラスタ化すればスーパーコンピュータのようになるのね！

でも人間が作るグループにもいろんな形態があるけど複数コンピュータのクラスタにもいろいろな種類があるの？

夫：そのとおり！いい質問だね。でもこれを話し始めると話が長くなるので、とりあえずスーパーコンピュータの話の前にスーパーマーケットの前に早く行こうよ！

妻：うん。

（腹をすかした哀れな旦那にも座布団を！）

ある夫婦の会話その3

スーパーマーケットで並列処理を体験！

夫：さあて、今晩は何食べようか？

妻：おなかすいているんでしょ！すぐできる簡単なものがいいわ。

夫：……（お湯で温めるだけとか）。

妻：あっニンジンとジャガイモが特価だわ！今日はカレーで決まりね。

夫：……（よかったレトルトでなくて）。

（やがてすべての材料が揃いレジの列に加わる）

夫：今日はやけに人が多いね。

妻：だって特売日ですもの。お客さんが多いし、さらにみんなふだんよりた

たノードにはクライアントからの要求が送信されないような機構（自動縮退）を備える（図4）。

さらにオンライン中にサービスノードを増設して、クラスタ定義情報を更新すれば、システムを停止せずにトランザクション処理能力をアップできる。

逆に自動縮退機能を利用して、ひとつだけサービスノードを止めて保守点検や障害部品交換などの作業ができる利点がある。ノードが回復したときには自動的にサービス開始となる。

スケーラビリティクラスタの応用形としてN対1バックアップ方式クラスタがある（図5）。1台のノードだけホットスタンバイ状態にしておき（図中の点線）、オンライン中のノード異常を検出したときに切り替える方式である。

障害や保守点検などのノードダウン時でも、本来のトランザクション処理能力の低下が許されない場合に適用さ

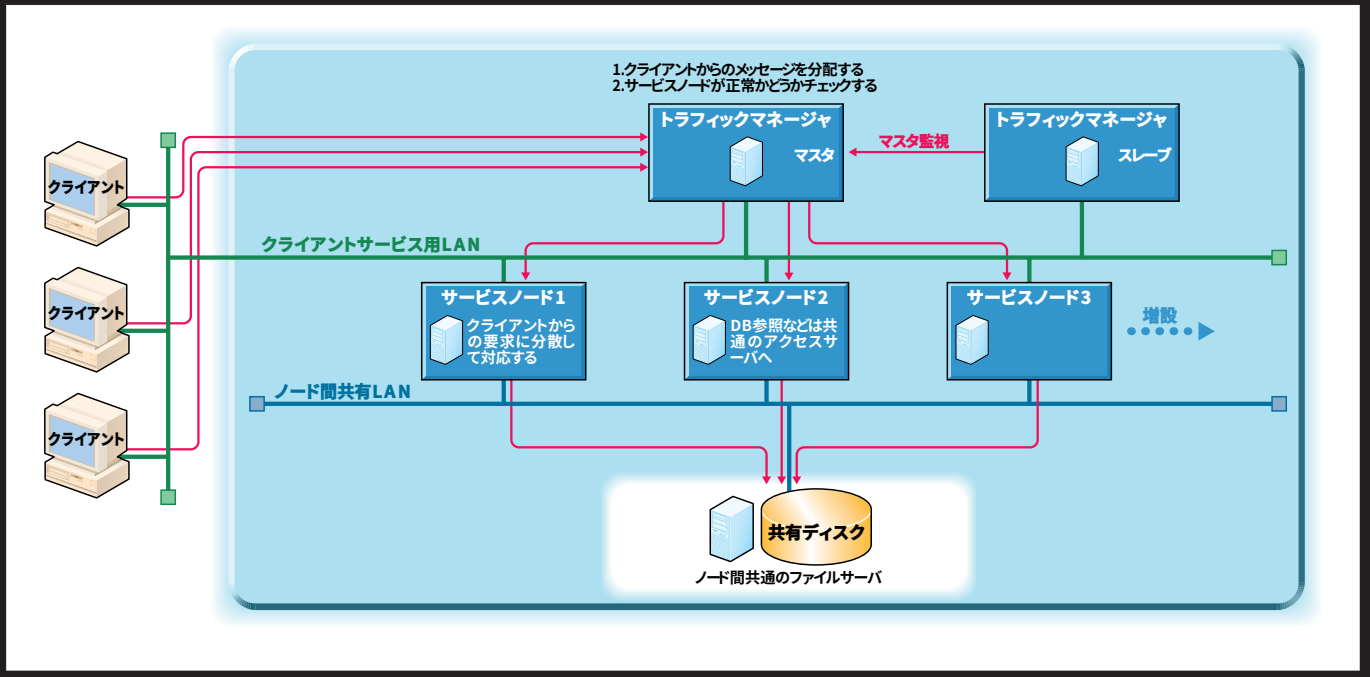


図2　スケーラビリティクラスタの構成例
複数用意したサービスノードに分散して処理させるように、トラフィックマネージャが制御する。

くさん買いこんでいるからレジの人がたいへんなのよ。

夫：こういう場合、クラスタ化処理すると行列を短くすることができるよ。

妻：各レジがノードに相当するのね。

夫：そう。ふつうに考えると君ならどうする？

妻：レジの数を増やすわ。

夫：うん。多くのクライアント（お客）に対して複数のノード（レジ）で処理するスケーラビリティクラスタを適用するのが一般的だね。

まるでこの混雑の状況は、人気のあるWebサイトに集中アクセスしている状況とそっくりだね。でもこのスーパーのレジはすべてフル稼働状態だよ。ほかに案はない？

妻：あとレジの処理スピードを上げるために、ひとつのレジを複数の店員で処理するやり方があるわ。

夫：ノードをこれ以上増やせない場合、ノード中のCPUをSMP化したりI/O周りを強化するのも有効な対策だね。

妻：品物のバーコードに赤外線を当てる作業（入力）と、その品物を別の袋に詰める作業（出力）を別々の店員が行えば処理スピードが向上するわ。

夫：入力と出力が平行に動作するだけでもかなりスピードアップだね。

妻：そしてさらに店員の作業スピードをアップさせる方法があるわ！

夫：目の前の店員さんは結構手際がいいと思うけど。どうするの？

妻：店員を改造してサイボーグ009にするのよ！

夫：そうか！「加速装置！」と叫んで奥歯にあるスイッチを押すと超高速に動作できる機能を使うわけだな？

妻：そしてもう1人をエイトマンで対応するの！

夫：おぉ！弾よりも速く品物を袋に入れることができるわけだ！

妻：そうすれば長い行列もあっという間に解消よ！

夫：素晴らしい〜最高の解決策だ！

れる。サービスノードに予備ノードが追加された形態である。

信頼性向上クラスタの最も効果的な適用分野はインターネットサーバやデータベースサーバである。1トランザクションあたりの実行時間は短縮できないが多くのトランザクションを同時に処理するのに適している。

2.高速クラスタ（科学技術計算向けクラスタ）

前述のスケーラビリティクラスタではトランザクションの実行時間は短縮できない。トランザクションの待ち時間を短縮して、多くのエンドユーザーへの均一なサービスを提供するのが目的である。

それに対して短時間で答えを出したいという要求には、この高速クラスタ（科学技術計算向けクラスタ）が適している。Linuxでは1993年NASAで生まれたBeowulf型クラスタが有名だ。日本で生まれたScore型クラスタも世界的に有名である。

参考として日本国内のLinuxディス

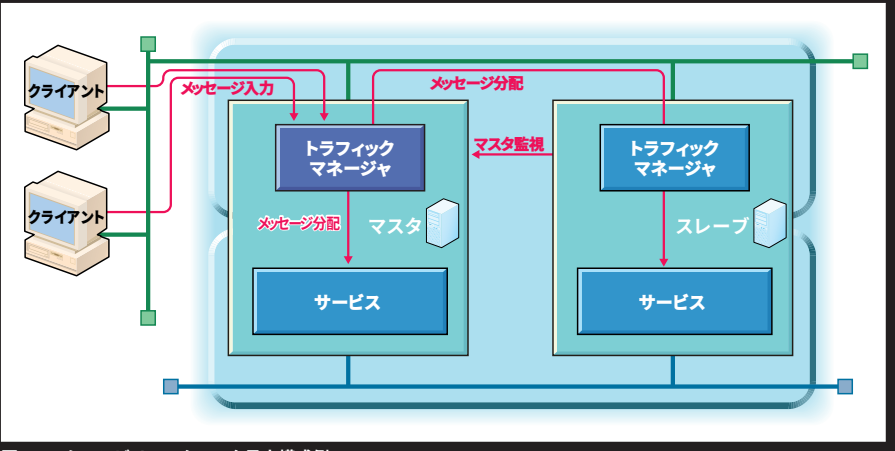


図3　スケーラビリティクラスタ最小構成例
トラフィックマネージャとサービスノードを1台のマシンで行えば、最低2台からクラスタを構成できる。

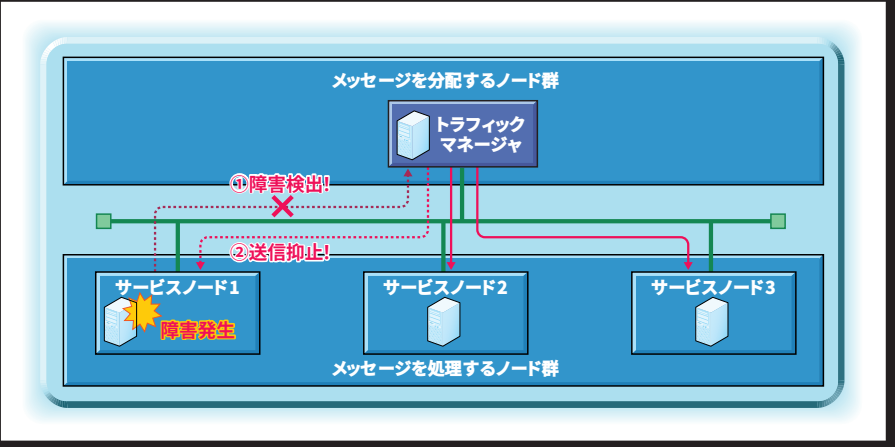


図4　縮退時のクラスタの動作
サービスノードに障害が発生した場合、トラフィックマネージャがそのノードを使用しないようにする。

う〜ん一度でいいからサイボーグ店員の作業姿を見てみたいな。

でもたぶん、卵や野菜がグチャグチャになっていると思うよ。さらに現状の技術では実現できそうもない高度な技術だね。不可能だとはいわないけど。

それに今の話は1960年代前後に生まれた人しか理解できないぞ。若い読者に対してもっとわかりやすい話はないの？

（ところで彼女はいったい何歳？）

妻：あなたって優しいのね。「バカ！」と叫んで、頭ごなしに否定すると思ったわ。

夫：ブツブツ（ほっ！実は内心そう思っていたよ。話をあわせてよかった！）

妻：もうほかに思いつかないわ

夫：そうだね。こんなに混み合っている環境ではどうしょうもないね。

妻：すいている状態だと何かできるの？

夫：うん。サイボーグの店員を使わずに高速にレジの処理スピードを上げる方法があるのさ。

妻：ええーっ！教えて！

夫：すいている場合、レジ（ノード）を増やしても商品の金額を集計する時間（トランザクションの処理時間）はどのレジ（ノード）でやっても大差ない。買い物カゴがレジを通過する時間は不変だ。

妻：うん。

夫：ヒントをあげるよ。この場合、買

トリビューターが手がけるクラスタとしてはレーザーファイブの「頼光：RAIKOU」、TurboLinuxの「En Fuzion」などが発表されている。

2.1 Beowulf

Beowulfは並列計算に使える複数コンピュータのアーキテクチャの名前だ。ハードウェアや制御ソフトの名前ではないことに注意してほしい。

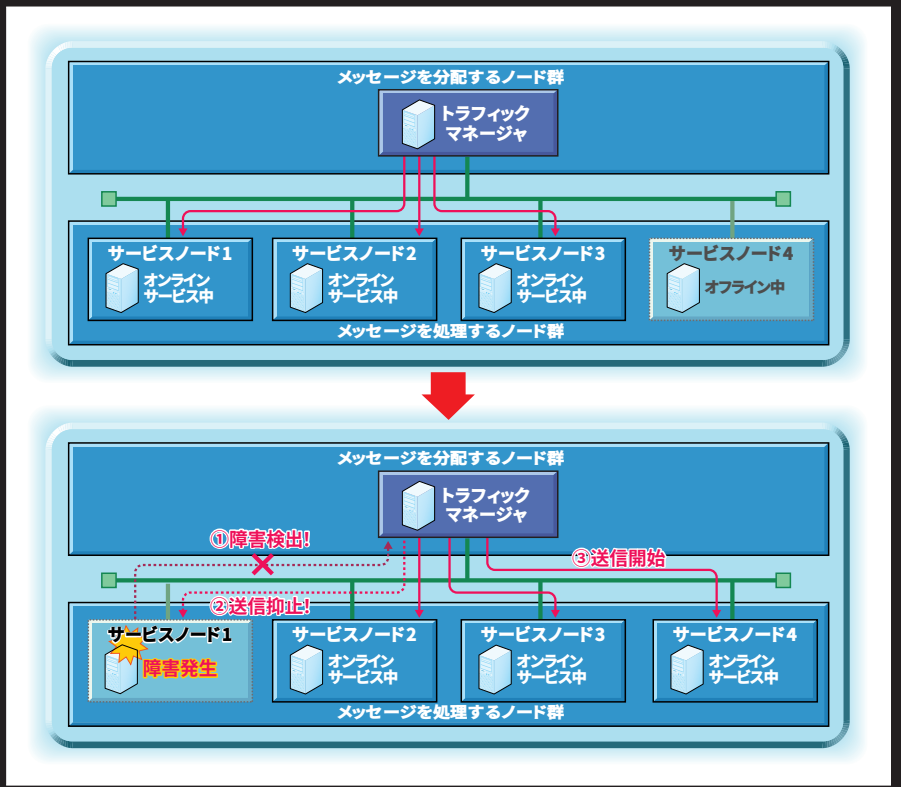


図5　N対1バックアップ方式クラスタ
ノードの障害発生時に性能が低下しないように、代替用のノードを用意しておく。

Beowulfは特別なソフトウェアパッケージの名前でもなく、新規ネットワークトポロジでも、特殊カーネル名でもない。複数のLinuxコンピュータをクラスタ化して高速な並列仮想コンピュータを形成する技術である。

詳細なアーキテクチャの定義は、おそらくBeowulf型スーパーコンピュータの数だけあることになるだろう。よくメーリングリストでもBeowulfは話題になるが、言葉の定義だけの不毛な議論になることだけは避けて欲しいものだ。

簡単にBeowulf型クラスタの考え方を簡単に説明したのが図6だ。

プログラム（計算式）のロジックを眺めると、平行に処理できる部分がある。これらを2台以上の複数のノードに分散して同時に計算すれば処理時間が短縮できる。

ジョブ1のように、1から100までを加算する場合、1から50までと、51から100までに分けて計算し、その結果を合計する。ジョブ2も同様だ。

い物カゴの品物が多いとレジの処理時間が長くなるという関係がある。

妻：少しだけ品物をカゴに入れればレジの店員は速く処理できるのはわかるのだけど……。

夫：おお！イイ線いっているぞ！

妻：残りの荷物をどうするか……。

わかったわ！買い物カゴを2つ用意するのよ。

夫：それで？

妻：品物を半分ずつにして私たち別々のレジに並んで処理すればいいのよ！そうすればトランザクションとしては2倍のスピードアップになるわ！

夫：大正解！と言いたいけど、ひとつ問題が発生するよ。

妻：あなたが一部お金を使うこと？

夫：うっ！（それも重要な問題だが）そうじゃなくて、特価品の扱いをなんとかしなくてはいけないのさ。

妻：あっそうか！1本で60円のニンジンが2本で特価100円だったわね。これを別々の買い物カゴに入れたら20円余計に支払うことになるわ。

夫：そのとおり！その解決策は？

妻：やりかたは2つあるわ。特価品はひとつのカゴに入れるというルールを守ってカゴに入れ、全体として半分の数になるようにするのよ。そうすれば今のレジのシステムを変更しないで会計処理を済ませられるはずだわ。

夫：ニンジンをセットにして1つの商品にして分散するのか。いわゆるデータの正規化処理みたいなものだね。そうすれば複数のノードで計算しても最終合計は矛盾がでないね。

Linuxクラスタの例ではTurboLinuxのEnFuzionで実現している平行処理クラスタに近い方式だね。で、もうひとつはどう処理するの？

妻：このやり方は現状のレジのシステムを変更する必要があるの。

夫：まさかレジに物質転送装置を導入するとかの話？

妻：ビシッ（平手打ち）！

Beowulf型では、各ノード間で通信しながらジョブを進めていくことになる。

このノード間の通信には、MPI（Message Passing Interface）などの並列処理用ライブラリや、PVM（Parallel Virtual Machine）など並列化のためのパッケージを使用する。

しかし、高速処理を実現するうえで大切なことは、ツールの機能を覚えることではなく、業務プログラムのどこがボトルネックになっているのか、平行に処理できる部分はないのかと検討することである。

さらにその考えを進めていくと、並列処理に適したアプリケーションをシステム設計段階の初期から大局的立場で検討することが重要だと気づくだろう。並列化ロジックが決まった段階で、それを実現するために一番適したツールは何かというアプローチが大切である。

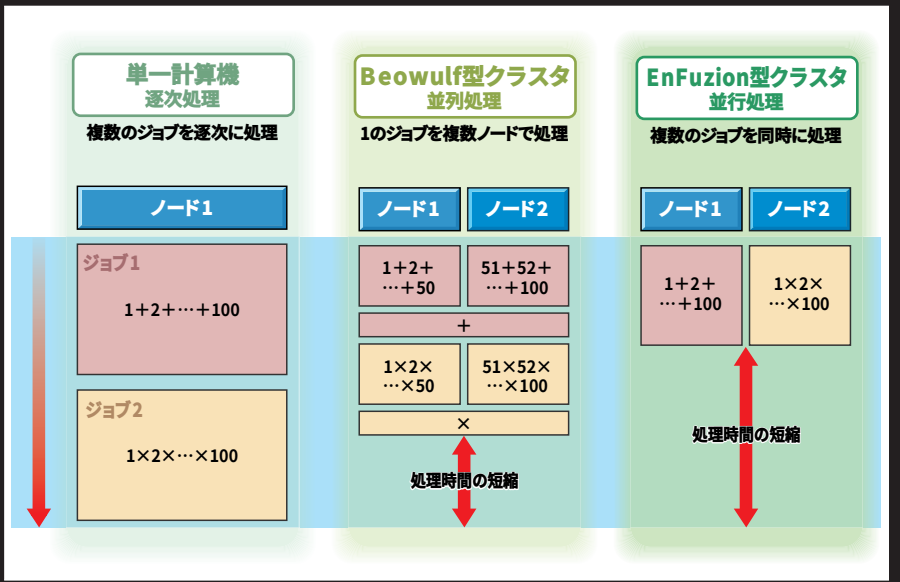


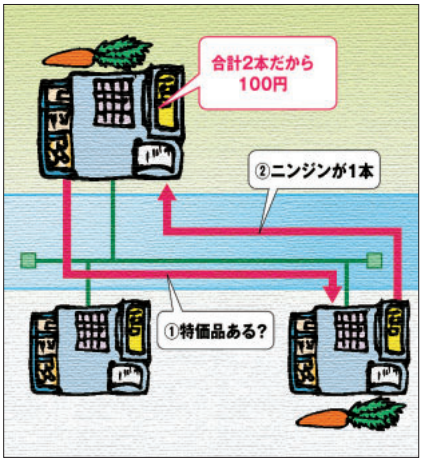
図6　BeowulfとEnFuzionの処理のしかたの違い
多くのデータを扱う計算では、分担して複数のノードで処理をすることにより高速に処理できる。

2.2日本のScore（エスコア）

Beowulfプロジェクトとほぼ同時期に技術研究組合新情報処理開発機構（Real World Computing Partnership、画面1）、通称RWCPという機構が平成4年度から10年計画で通産省からプロジェクトを受託して各種研究を

画面1　技術研究組合新情報処理開発機構
RWCP(Real World Computing Partnership)
http://www.rwcp.or.jp/

今度くだらないギャグを言ったら、その物質転送装置を使って、あなたの脳みそにドリルミサイルを打ち込んで



やるわよ！

夫：いてて！わかったよまじめに聞くよ（あとで波動砲をお見舞いしてやる！）。

妻：お客側（クライアント）では特価品を意識せずに複数の買い物カゴに分散できるの。

夫：確かに品物が増えてくると何が特価品だったか忘れてしまう可能性があるよね。クライアント（客）側でなにも意識しないで、2つ以上の買い物カゴに品物を分けることができればベストだね。

妻：うん。とにかく品物を2つ以上の買い物カゴに分けてお互い分散してレジに並ぶの。

夫：ここまでは同じだね。レジのシステムの改造が済んだという条件ではどのようになるのかな？

妻：次にレジの係員に何番と何番のカウンタで並列に処理してくださいと指示するの。

夫：ほう。

妻：すると指定された番号のレジとレジが互いに通信できるようになるの。つまり各レジ（ノード）間でメッセージのやり取りができるようにするの。

夫：Beowulf型クラスタのようにMPI（Message Passing Interface）によるノード間通信をレジで実現するのだね。

行っている。

Scoreクラスタのシステムソフトウェアは、その技術分野で研究開発されたもののひとつである。

Beowulf型クラスタと比較して、性能と使い勝手を追求したクラスタである。多数のノード構成になった場合、Beowulf型クラスタより効率よく並列化できるのが特徴だ。おもな支援ライブラリとしては、

画面2　並列タンパク質情報解析システム
PAPIAのWebサイトで実際に並列計算を行える。
http://www.rwcp.or.jp/papia/papiaJ.html

- MPI通信ライブラリ
- MPC++マルチスレッドテンプレートライブラリ
- SCASHソフトウェア分散共有メモリ

などがある。

Scoreの最新版は、

```
http://www.rwcp.or.jp/lab/pdslab/dist
```

からダウンロードできる。

Scoreの並列演算のスピードを体験するなら、タンパク質の構造解析を公開しているPAPIA（Parallel Protein Information Analysis system）というWebページにいくとよい。実際にScore型クラスタを動作させることが可能だ（画面2）。

2.3　EnFuzion

高速に処理するためには、先ほどの例のように並列化のためのライブラリをアプリケーションに埋め込んで、処理ロジックを並列用に書き直す作業を行う。当然高度な技術が必要だ。

また並列処理プログラムの出す答えが正しいかどうかの検証も大変な作業である。

特に浮動小数点の演算を多用する科学技術計算などは、並列ロジックが正しいとしても小数点の丸め誤差が処理結果に大きく影響する。

これはなにも特殊な状況ではなく、日常の仕事でも掛け算や割り算が入り混じった計算を電卓や表計算ソフトで実行した場合、論理的には間違っていなくても、計算順序を変えるだけで小数点以下の値が異なってくるのはだれでも経験していると思う。

並列化アプリケーションが今までと異なった値を出したとき、精度が上がって新発見をしたのか、逆に誤差が広がったのか、またはバグなのか、検証用のプログラムが別途必要になるだろう。誤差を考慮しないでやたら並列化して処理を高速化するのも一考の余地があるのだ。

主にロジックを分割するBeowulfやScoreの方式に対して、TurboLinuxから発売されているEnFuzionというクラスタ化ソフトは、他のLinuxのクラ

妻：うん。マスタになっているレジが別のノードに通信して「特価品が買い物カゴに入っていないか？」というメッセージを送信するの。

夫：受信したレジでは「ニンジンが1本あるよ」とか返すわけだな。

妻：そうよ。返答がきたレジに対してはマスタレジから「計算処理済み」というメッセージを送信するの。

夫：メッセージを受け取ったセカンダリのレジでは金額の計算処理から対象外とするためだな。

妻：そう。同時にマスタのレジでは特価品を商品コード別に集計するの。ニンジンの合計本数が2本だから100円という計算をマスタレジで行うの。

夫：Beowulf型クラスタをスーパーのレジのシステムに適用するわけだな。

妻：うん。実際に実現しようとするとかなり面倒だけど、不可能な話ではないわ。これが実現できたら文字通りスーパー[2]マーケット、あるいは超スーパーマーケットになるわ！

夫：あはは！とにかく今回の買い物でクラスタの考え方がすっかり理解できたじゃないか！

妻：あなたのおかげよ。感謝するわ。ということは、我が家でもクラスタを導入すれば、もっと速く計算ができるようになるのね。そうすれば夕飯を買い忘れる心配がなくなるわ。ところで、我が家ではクラスタ用のPCは何台購入できるの？

夫：うっ！（きたか）それは予算次第だよ。台数に応じて速くなるけど、性能に対する人間の欲望には限度がないからね。我が家の家計に依存するのさ（企業の場合も同様だよ）。

妻：それって今月は買えないといっているのと同じだわ！今度のボーナスまで我慢するしかないわね。

夫：あはは！さすが結論が早いね。家計をやりくりしている君は我が家のスーパーコンピュータだよ！

スタ製品とは一味違うユニークなアプローチをしている。

既存のプログラムは変更せずに、そのまま事前に各ノードに分散配置しておくのだ。

プログラム実行時には最初にEnFuzionが入力データを分割する。そして各ノードに分散配置された同一プログラムに対し分割データを送信し、各プログラムが出力したデータをひとつのデータとしてまとめる。さらにその出力データを別のプログラムに分散送信して実行というジョブを繰り返すのだ。

EnFuzionはデータを平行に処理するクラスタであることから、平行処理クラスタともいわれている。もちろん、データが分割されても正常に動作するアプリケーションであることが大前提となる。

もしデータ解析やシミュレーションなどの用途で、データが分割されても正常に動作するアプリケーションがあったら、EnFuzionを検討する価値はあるだろう。利用ユーザーは既存プログラムを各ノードに配置して、データを分割する単位を指定し、出力先を指定するだけで高速処理が実現するのだ。

■次回はいよいよクラスタ始動

今回の特集ではクラスタに関する概念を理解してもらうために、現実には存在しない奇妙な夫婦の会話を通してクラスタの説明をしました。楽しんで読んでいただけたでしょうか？

いま直面している問題がクラスタシステムで解決できるのか、さらに自分が必要とするのは、どのクラスタシステムなのかという指針になれば幸いです。

次回は、TurboLinux Cluster Serverを使って、クラスタの実際の設定と応用例を紹介したいと思います。

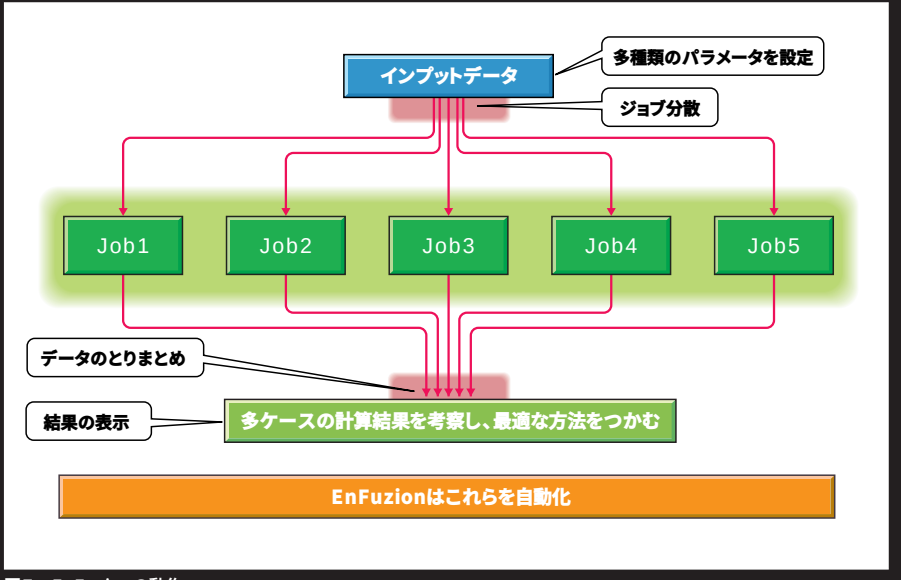


図7　EnFuzionの動作
EnFuzionはデータを分割して複数のノードで実行させ、結果を統合することで高速に計算できる。

手作りサーバでMP3レコーディング

ラジオ自動録音システム(前編)

**動画キャプチャが身近なものになってきた現在であっても、古くからの技術の積み重ねがあり、豊富なツールを簡単に入手できる「音声キャプチャ」は、現在のPCの身の丈にもっとも合った応用分野である。
ここでは、ごく普通の構成のLinuxマシンと、フリーの音声関連ツールを使って、ラジオ番組の自動録音を行うシステムを構築してみよう。**

文:竹田善太郎
Text : Zentaro Takeda

WindowsやMacintoshの世界では、ビデオ編集やTV放送の録画をPCで行うことが流行っている。ビデオキャプチャカードや大容量HDDの普及がそれを後押ししているようだ。しかし、ビデオ編集には専用のソフトウェアが必要で、「まとも」に使えるようにするにはCPUやメモリの増強も含めて10万円以上の出費を覚悟しなければならない。ビデオキャプチャ機能と必要なソフトウェアがバンドルされているPCも発売されるようになったが、価格は高いし、専用ハードウェアと専用ドライバでガチガチに固められた構成になっていて、PCそのものを楽しみたユーザーにとっては「つまらない」マシンともいえるだろう。もちろん、Linuxを動かすプラットフォームとしては不向きである。

現在のPCでは、性能の面においても、ビデオデータを扱うにはまだ十分とはいえない面もある。たとえば、ハードディスクにビデオデータを録画している最中は、マシンをほかの作業で使うのは難しい。高クロックCPUのおかげで、重いアプリケーションでもまったく動かないわけではないが、レスポンスは極度に落ちるし、無理をすると録画中のデータに「コマ落ち」が生じたり、システムのロードが高くなりすぎてハングアップするなど、よいことはあまりないのだ。

では、映像が無理なら「音声」はどうかといえば、こちらはまさに「百花繚乱」といってよい。PCに音声キャプチャ機能を持ったサウンド機能が装備されているのはあたりまえだし、たとえサウンド機能がなくても、2〜3千円も出せば高品質のサウンドカードが入手できる。また、Windows、Linuxを問わず、音声を扱うソフトウェアは、市販のものを買うまでもなく、フリーソフトとして、さまざまな機能を持つものが簡単に入手できるのだ。CPUやメモリ、ハードディスクなどの性能面についても、音声を扱うのであれば数年前のマシン(166MHz程度のMMX Pentium、メモリは64Mバイト、HDDは数Gバイト)で十分である。もちろん、最新の構成のマシンなら、さらに快適に音声データを扱うこと

ができる。

Linuxにおいてつねに心配の種となるのは、ドライバ類の供給だが、商用ドライバの「OSS」やフリードライバの「ALSAプロジェクト」などのおかげで、相当マイナーなサウンドカードであっても、Linuxで次々とサポートされるようになっている。

なぜラジオの録音なのか

PCで音声データを処理するといえば、すぐに思いつくのが「MP3で音楽を……」ということになるだろう。実際、本誌を始めパソコン雑誌に「MP3」関連の記事が載らないことはない。筆者も、手持ちの音楽CDをMP3形式のデータに変換して、仕事中のBGMとして再生したり、シリコンオーディオプレーヤで外出中に楽しんだりしている。また、インターネット経由で（合法、非合法という面ではさまざまなものがあるだろうが）音楽データをダウンロードして楽しんでいるユーザーも多いだろう。

しかし、それだけならLinuxを使う意味はあまりないのではないだろうか。とくに、「バックグラウンドでの自動実行」を得意とするLinuxを、Windows PCでもできる音楽CDのMP3化にしか使わないのは、あまりにももったいない。もう少し生産的なサウンド機能の活用法はないかと考えたのだ。そこで思いついたのが「ラジオの語学番組を録音して、好きなときに再生できるシステム」の構築である。

ラジオ番組の留守録音は、以前はラジカセなどのタイマー機能を使って行うのが普通だった。しかし、予約できる番組の数は1つに限られるし、テープの掛け替えを忘れて録音に失敗することも多い。また、1年分の講座を録音しようとすると相当な本数のテープが必要になり、かなりの場所をとるうえに、目的の日時の録音を検索するのが大変である。

PCのサウンドカードの録音（キャプチャ）機能を使って、ラジオの音声を留守番録音できれば、好きな数の番組を予約録音できる。また、データを圧縮して保存するようにすれば、ハードディスク上で数Gバイトの容量を割くだけで、1年分の講座をすべて記録しておくこともできるはずだ。

ところで、ハードディスクにラジオ番組を留守録音する機能を持った単体の製品は、じつは市販されているのであるが、PCを1台買えるほどの高価なものであるうえ、PCとのインターフェイスが限られているので、たとえばシリコンオーディオプレーヤに転送して聴くというような使い方ができない。Linuxなら、既存のツール類を組み合わせるだけで同様のシステムを作れるうえ、LAN環境でなら別のPCから録音した番組を再生したり、ポータブルプレーヤに転送したりすることも簡単ではないかと考えたのだ。

今回構築する自動録音システムでは、特別なハードウェア

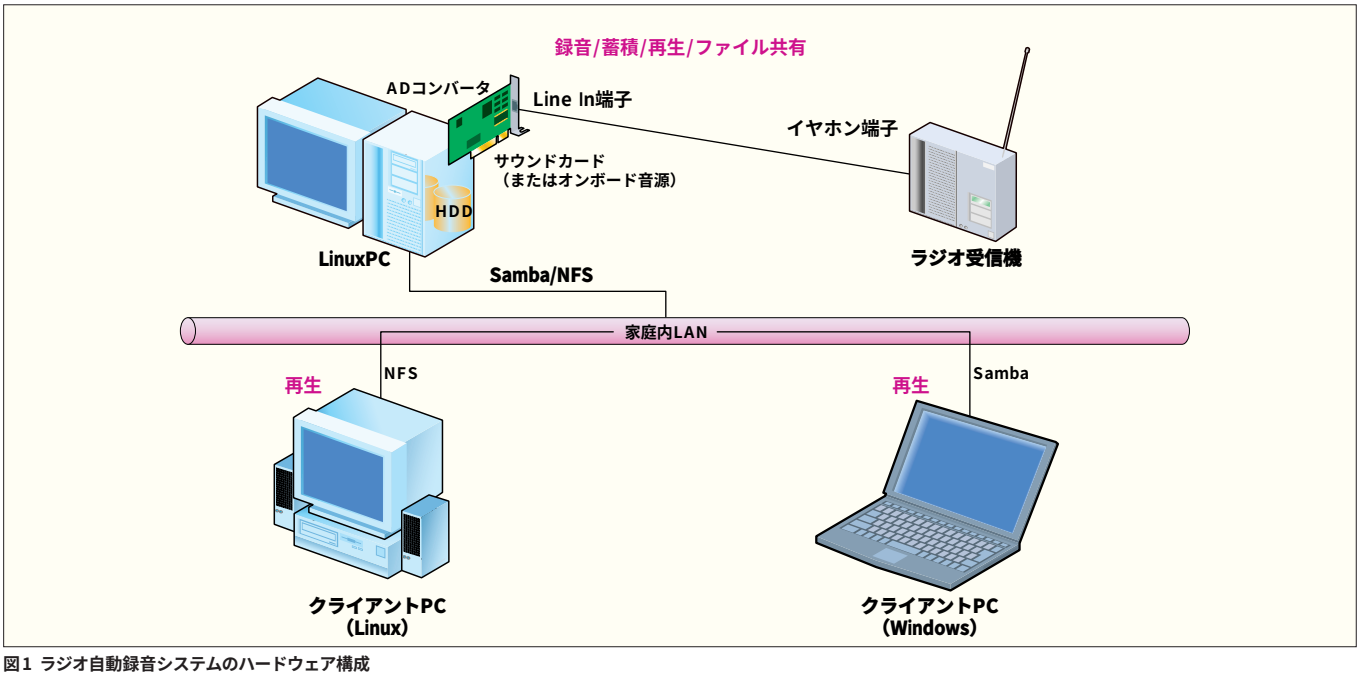


図1 ラジオ自動録音システムのハードウェア構成
サウンドカード、ラジオ受信機以外に、特別な装備は必要ない。サウンドカードは録音（キャプチャ）機能とLinux用のドライバさえあれば、マザーボードに組み込みの音源でも問題ない。ラジオからのアナログ音声は、サウンドカードのライン入力（Line In）に入り、サウンドカード内蔵のADコンバータによってデジタルデータへ変換され、ハードディスクに蓄えられる。

やプログラムは一切使用していない。OS（Linuxのディストリビューション）に付属する機能、あるいはコピーフリーで流通しているプログラムに、若干のPerlスクリプトを追加するだけである。システムの構成もきわめてシンプルなので、読者は自分の用途や好みに合わせて使用するプログラムやドライバを変更したり、機能を追加することができるだろう。

システムの概要

自動録音システムのハードウェアは、音源となる「ラジオ」とPCに内蔵されているサウンドカードが主な要素になる。ラジオの音声出力をサウンドカードの「ライン入力」端子に接続し、サウンドカードの録音機能（音声キャプチャ機能）によって入力信号をPCMデータに変換し、ハードディスク上の音声ファイルにするわけだ（**図1**）。

一方のソフトウェアの構成（**図2**）では、まず重要なのがサウンドカードのドライバである。現在のところ、Linuxのディストリビューションに付属するドライバは、すべてのサウンドカード製品を網羅しているわけではなく、多くの場合、ドライバを別途入手しなければならない。ドライバの入手とインストールの方法については、あとで詳しく説明するが、ドライバがなければ音声キャプチャ機能は使えない。

サウンドカードのドライバをインストールすれば、音声キャプチャ自体は/dev/dspなどのデバイスファイルにアクセスするだけで可能になる。しかし、この状態ではキャプチャするデータの種類（ビットレートやデータの形式）を指定するのが面倒なほか、指定した長さの時間だけデータをキャプチャする操作は難しい。そこで、別途サウンドキャプチャ専用プログラムを入手して使うことになる。今回では、サウンドドライバに付属するキャプチャツールを利用している。

サウンドカードからのキャプチャデータは圧縮されていない生のPCMデータ（WAVファイル）なので、そのままではファイルサイズが大きくなってしまう。そこで、キャプチャしたデータをさらにMP3形式に再圧縮して、ファイルサイズを小さくするとともに、各種のMP3再生用ソフトウェアで扱えるようにしたり、MP3ポータブルプレーヤなどに転送するのに便利なようにする。

さらに、音声キャプチャからMP3形式への圧縮までの一連の作業を1つのコマンドで実行できるようにするために、簡単なスクリプトを作成した。そしてこのスクリプトを、Linuxの「タイマー機能」である「cron」から起動することで、定期的にラジオ番組を録音できるようにするわけだ。スクリプトはPerlで記述したが、シェルスクリプトで記述することもできるはずだ。Perlを使っているのは、ただ単に筆者

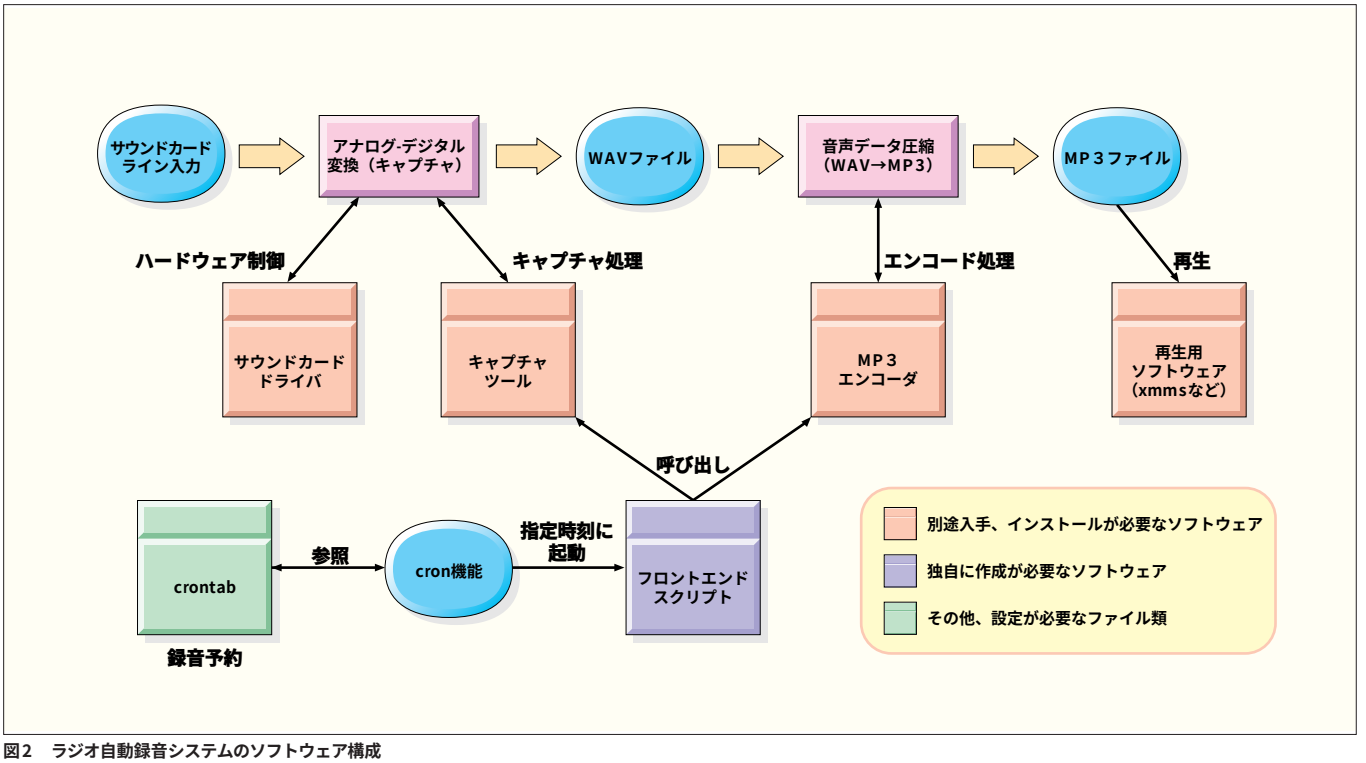


図2　ラジオ自動録音システムのソフトウェア構成
サウンドカードのドライバ、音声キャプチャ用ツール、MP3エンコーダを用意する必要がある。その他のツールやスクリプト類については、Linuxに標準で装備されていたり、自分で作成できるものばかりだ。番組の録音予約にはLinuxの「cron」機能を使うので、「crontab」ファイルを編集することになる。

がPerlに慣れているからにすぎない。

なお、本記事の題材としているシステムで使っているディストリビューションは、TurboLinux Workstation日本語版6.0であるが、RPMによるプログラムのインストール以外で、ディストリビューションに固有の機能は使っていないので、他のディストリビューションでも同じような手順でシステムを構築できるはずだ。

ハードウェアの準備

●マシンのスペック

音声データのキャプチャ処理は、いわゆる「リアルタイム」のデータ処理に分類されるため、ある程度は高速なレスポンスが要求される。しかしながら、録音の最中に重いアプリケーション（X Window Systemなど）を動かしたりしないのであれば、古い486システムでも十分に機能するだろう。MMX Pentium以上のマシンならキャプチャに関して性能上の不安はないし、最近のPentium IIIやCeleronのマシンであれば、キャプチャの最中にXの重いアプリケーションを動かしたり、コンパイルなどの負荷の高い作業を行っても、目立った問題は生じないはずだ。ただし、これについてはサウンドカードの性能やメモリの容量、使用するMP3エンコーダの種類などによって状況が変わる可能性がある。

●サウンドカード

ハードウェアに関してもっとも注意が必要なサウンドカードについては、Linux用のドライバが提供されていて、そのドライバでのサウンドキャプチャが可能でなければならない。一般に、LinuxではSound Blaster系のサウンドカードを使っているユーザーが多いようだが、どのサウンドカードが良いかについては、ここで述べることはできない。たとえばSound Blaster系（「互換」と記載されているものを含む）のカードでも、Linuxでうまく動かない製品があるからだ。

サウンドカードに限らず、ネットワークカードやディスプレイアダプタについてもいえることだが、Linuxで利用しようとする場合、ドライバのサポート状況を確認したうえで、使用するカードを決めなければならない。しかしながら、サウンドカードの場合、ディスプレイアダプタほどはモデルチェンジが激しくなく、パソコンショップでよく見かけるような製品であれば、OSSやALSAのドライバが対応している可能性が高い。

新たにサウンドカードを購入するのであれば、使用するLinuxディストリビューションでサポートが明記されているカード、あるいは、追加でドライバをインストールしてもよいというのであれば、OSSやALSAのサイトを参照して、サポートが明記されているカードを選べばよいだろう。サウンドカードのドライバ類の入手やインストールの詳細については後述する。また、Linux用サウンドドライバの現状については、137ページのコラムを参照してもらいたい。

マザーボードの中には、ボード上に音源チップと入出力端子が搭載されていて、後付けのサウンドカードを必要としないものもある。いわゆる「オンボード音源」であるが、従来はLinuxでのサポート状況が不明のものが多く、敬遠するユーザーが多かったようだ。しかし、とくにALSAドライバではオンボード音源のサポートが積極的に行われており、Intel 8x0系、VIA Apollo Pro 133系の各チップセットのオンボード音源であれば、Linuxでも利用できる可能性が高い。オンボード音源は、別売りのサウンドカードを購入する場合に比べ、コストの面で有利になることもあり、新規にマシンを組み立てるのであれば、検討対象にしてもよいだろう。ここで実例として取り上げている構成（実際に筆者が自宅で自動録音用に使用しているマシン）も、VIA Apollo Pro 133をチップセットに使用したマザーボード「ABIT VA6」のオンボードサウンド機能とALSAのドライバを使用している。

●ラジオ受信機

音源となるラジオ受信機については、「イヤホン端子」さえあれば、どんなものでもよい。ちょっと実験だけしてみたいということであれば、980円で売られているポータブルラジオでも十分だろう。しかし、「1年間にわたって継続的に語学番組を録音したい」というような場合には、AC電源が使えるものでなければならない。100Vの電源を直接使用できるラジオは、最近ではめったに見かけなくなったが、ACアダプタを使用できるポータブルラジオもあるので、そのようなものを選択すればよい。あるいは、使い古した「ラジカセ」（カセットデッキ部分が壊れているが、ラジオだけは聴けるようなもの）を流用してもよいだろう。

高級なラジオやラジカセの中には「ライン出力端子」をもっているものがあり、イヤホン端子よりもこちらのほうが向いているのでは、と考えるかもしれないが、サウンドカードの機種（あるいはドライバのバージョン）によっては、PC側のライン入力のレベル調整ができなかったり、調整できても音が歪んでしまうことがある。イヤホン端子であれば、ラジオ側のボリュームでレベル調整が簡単にできる。

●接続コード

イヤホン端子とライン入力端子を接続する場合、「抵抗入り」の接続コードが必要ではないかと考える読者がいるかもしれないが、結論からいえば「抵抗入り」のコードは不要だ。逆に、抵抗入りのコードを使うと、入力レベルが小さくなりすぎて、ラジオ側のボリュームとPC側の入力レベルの両方を最大にしても、十分な録音レベルを確保できなくなることがある。

接続コードには、ステレオタイプ、モノラルタイプ、ステレオ—モノラル変換タイプなどがあるが、これは録音の目的やラジオの種類によって選べばよい。PC側のライン入力端子はステレオタイプになっているので、ラジオ側の出力もステレオなら、コードもステレオタイプのものを使えばステレオ録音ができる。

なお、PCのライン入力端子にモノラルタイプのケーブルを接続すると、左側のチャンネルにしか信号が入らなくなる

Column

ラジオの選び方

本文中でも触れたように、自動録音用にラジオを用意する場合、イヤホン端子を備えていて、AC電源の使えるものなら基本的になんでもよい。しかし最近では、一般の家電販売店に陳列されているラジオの種類はかなり限定されてしまう。いわゆる「通勤ラジオ」と呼ばれる、電池のみを電源とした超小型のラジオはかなり多数並んでいるものの、AC電源も使えるものは意外と少ないのだ。

ラジオを選ぶにあたっては、設置する場所についても考えておく必要がある。電波を受信する装置であるという性質上、建物の中で受信しようとすると、置く場所や向きによって受信状態が大きく変わってしまう。できるだけ窓側近くの高い位置に置き、電波の入ってくる方向（必ずしも「放送局の送信アンテナの方向」とは限らない）に本体の正面（または背面）を向けるのがよいとされている。

ラジオ受信機を選ぶ場合には、置き場所にうまく収まるかどうかも考慮したほうがよい。幸い、最近のラジオはどれも小型軽量なので、置き場所に困るということはないのだが、「AC電源」にこだわるあまりに、クラシックなスタイルの木箱に納まった大型の製品を買ってしまうと、いざ設置しようとする段階で困ることになる。もちろん、外部アンテナを使うつもりなら、設置場所は関係ないのだが、大型のラジオは内蔵のスピーカーも高級なので、半永久的にイヤホン端子を使うPCとの接続にはもったいない。

都市部などで送信所が比較的近い場合は問題ないのだが、送信所から遠い郊外で受信する場合や、送信所から近くてもコンクリートの建物の中など電波状態が悪い場合には、ラジオ受信機の「感度」も気になるところだ。とくに、BGM代わりに掛け流しにする場合には気にならなくても、語学番組などを「じっくり」聴き取ろうとするとき、ノイズや歪みが多いと、集中力がそがれて「学習効果」も半減してしまう

一般的にいって、ラジオの「感度」は価格に比例するようである。同じような外観・機能の製品を比べた場合、価格の高いほうが音質がよく受信感度も高いと判断して間違いないだろう。もちろん、実際に聞き比べて判断するに越したことはないのだが、時間がなかったり試聴できなかったりする場合には、「価格」を判断材料にするのが間違いない。特に、単機能の小型ラジオは、デザインや付加機能の面で差別化することは難しいので、価格の差は即性能の差だと思っていてよい。

チューニング方式の「デジタル」と「アナログ」については、少なくとも感度や音質の面ではあまり関係はない。チューニングのしやすさは「デジタル方式」のほうが優れているのだが、価格は若干高くなってしまう。個人の好みで選ぶしかないだろう。

本文中では、使わなくなった「ラジカセ」などを流用してもよいと書いたが、一般にラジカセのラジオ部分の性能には過剰な期待はできない。1000円程度で売られているポケットラジオと同程度だろう。とはいえ、読者がふだん使っている場所でクリアに受信できるのなら、自動録音用に使うのになんの問題もない。「リサイクル」の観点からも、古いものを積極的に利用したいものだ。

性能を重視するなら「海外放送対応」の短波ラジオをお勧めする。短波放送を受信しないのにもったいない、と考えるかもしれないが、この種のラジオは、中波放送（いわゆるAM放送）の受信性能も一般のラジオより格段に優れているからだ。普及機種ならば6000円程度で購入できるものもあるし、ACアダプタにも対応しているものがほとんどだ（一部の小型製品を除く）。

参考までに、筆者が自動録音用に使っているラジオは、ソニーの「ICF-SW11」という海外放送対応の短波ラジオである（写真）。アナログチューニング方式のポータブル機で、いわゆる「入門機」に位置付けられる製品だが、筆者が使ってみたかぎりでは、受信性能については上位機種と遜色なく、音質や使い勝手も良好である。もちろん、イヤホン端子と外部AC電源端子を備えており、FMに限るがステレオ放送も受信できる。ACアダプタが別売りである点と、単機能のラジオとして見ると、決して安くはない価格が難点だが、他の短波ラジオに比べても入手しやすく（一般の家電販売店にも置いてあることが多い）、お勧めである。

ICF-SW11設置状況
筆者が自動録音マシン用に使っている、ソニーの「ICF-SW11」。筆者の自室は鉄筋コンクリート造りの建物の北側（放送局とは反対側）にあり、そこそこの受信性能のあるラジオが必要だったのでこの機種を選択した。ラジオ本体は窓枠に置き、そこから3mの接続コードで机の下に置いたPCと接続している。

が、キャプチャやエンコードの段階でモノラルデータに変換すれば、音質上の問題はない。心配なら、モノラル―ステレオ変換タイプのコードを使えばよいだろう。

PCの「ライン入力」端子とラジオの「イヤホン」端子を用意したコードで接続すれば、ハードウェアの準備は完了である。すでにサウンドドライバがインストールされている状態なら、ここでラジオの音量とPC側のライン入力のレベルを調整すれば、ラジオの音声がPCのスピーカーから聞こえてくるはずだ。録音時の細かいレベル調整の方法については後述する。

サウンドドライバの入手とインストール

すでに述べたように、今回のシステムでは外付けのサウンドカードではなく、マザーボード（ABIT VA6）に組み込みのサウンド機能（VIA Apollo Pro 133 VT82C686AおよびAD1881）を使用している。このサウンド機能は、いわゆる「AC '97互換」のオーディオコーデックだが、Linuxで利用するにはこのサウンドチップ専用のドライバが必要になる。TurboLinux Workstation日本語版6.0には、このチップに対応するドライバが含まれていなかったため、別途入手する必要があった。

さいわい、ALSAプロジェクトでVT82C686Aのドライバが公開されているので、それを入手してインストールすることにした。別のサウンドカードを使用する場合も、ドライバの入手、インストールなどの手順はほとんど変わらないはずなので、本記事の手順を参考に作業を進めてもらいたい。ALSAドライバの場合、各ドライバごとのインストール手順は、ドライバのファイルと一緒に配布されているドキュメントファイル「INSTALL」に記載されているので、それを熟読してから作業を進めること。また、ALSAで対応しているサウンドカードの一覧については、ALSAプロジェクトのWebページ（http://www.alsa-project.org/）で参照できる。

ALSAのサウンドドライバの最新版は、ALSAプロジェクトのWebページからダウンロードできる。ダウンロード用のファイルは、ドライバ（alsa-driver-0.5.9b.tar.bz2）、ライブラリ（alsa-lib-0.5.9.tar.bz2）、ユーティリティ（alsa-utils-0.5.9a.tar.bz2）の3種類のパッケージに分かれている（2000年8月現在）。それぞれのファイルにはソースファイル一式とドキュメント類が含まれている。バイナリファイルはないので、自分でコンパイルしなければならないが、コンパイル作業自体に難しい点はない。

▼リスト1　ALSAドライバのINSTALLドキュメントの一部

```
Module snd-card-intel8x0.o
--------------------------

  Module for AC'97 motherboards from Intel.
                    * Intel i810/810E, i820, i830, i840, MX440

  snd_pbk_frame_size  - max playback frame size in kB (4-128kB)
  snd_cap_frame_size  - max capture frame size in kB (4-128kB)
  snd_mic_frame_size  - max microphone frame size in kB (4-128kB)

  Module supports autoprobe and multiple bus-master chips (max 8).

Module snd-card-via686a.o
-------------------------

  Module for AC'97 motherboards based on VIA 82C686A (south) bridge.

  snd_mpu_port        - 0x300,0x310,0x320,0x330, otherwise obtain BIOS setup
  snd_joystick       - 1 = enable, otherwise obtain BIOS setup
  snd_pbk_frame_size  - max playback frame size in kB (4-128kB)
  snd_cap_frame_size  - max capture frame size in kB (4-128kB)

  Module supports autoprobe and multiple bus-master chips (max 8).
```

ALSAドライバのインストール手順ドキュメント（INSTALL）には、インストールの手順のほかに、利用可能なドライバの名前（サウンドカードの機種別）と、固有パラメータの情報が記載されている。パラメータについての情報をユーザーが使うことはまずないが、ドライバ名とサウンドカード（チップセット）との対応、すなわち、自分の環境でどのドライバを使えばよいかについての情報はここで得られる。以下は、インテル製チップセットとVIA製チップセット内蔵のサウンド機能を使うためのドライバ情報である。

●ファイルの展開とコンパイル

コンパイルとインストールは、ドライバ、ライブラリ、ユーティリティの順に行う必要がある。この順番を間違えるとインストールできないので注意すること。まず、ドライバのアーカイブファイルを作業用のディレクトリに展開し、コンパイルする。なお、最近のALSAドライバのパッケージは、tarファイルをbzip2で圧縮された形式になっているので、bzcatコマンドで圧縮を解除してから、tarコマンドでファイルを展開する。

```
% bzcat alsa-driver-0.5.9b.tar.bz2 | tar xvf -
```

すると、カレントディレクトリにalsa-driver-0.5.9というディレクトリが作成され、その中にソースファイル一式が展開される。

このディレクトリには「INSTALL」という名前のドキュメントファイル（英語）が含まれているので、その中の「Quick Install」という項目を参照しながらコンパイル作業を進めるが、特殊なカードを使用していたり、カードのジョイスティックやMIDIインターフェイスを有効にするなどの付加的な機能を使わない限り、通常は、次の手順で問題ないはずだ。

```
$ cd alsa-driver-0.5.9b
$ ./configure
$ su
# make install
# ./snddevices
```

▼リスト2　/etc/modules.confの設定例

```
# ALSA portion
alias char-major-116 snd
options snd snd_major=116 snd_cards_limit=1
alias char-major-14 soundcore

alias sound-card-0 snd-card-via686a ①
alias sound-slot-0 snd-card-0
alias sound-service-0-0 snd-mixer-oss
alias sound-service-0-1 snd-seq-oss
alias sound-service-0-3 snd-pcm-oss
alias sound-service-0-8 snd-seq-oss
alias sound-service-0-12 snd-pcm-oss
```

マザーボード（ABIT VA6）上のVIA Apollo Pro 133のサウンド機能を利用する場合の設定例。「# ASLA portion」以下をサウンド機能のために追加する。通常は、①の部分を自分のサウンドカードの名前に変更すればよいはずだ。

ALSAドライバは必要に応じて動的にロードされる「モジュール」形式のドライバなので、ロードするドライバをLinuxに登録する必要がある。これは、INSTALLドキュメントに従って、/etc/modules.conf（Red Hat 6.2Jなどでは/etc/conf.modules）ファイルを編集することで行うのだが、使っているサウンドカードの機種に合わせて、ロードするドライバ名などを変更しなければならない。対応しているサウンドカード（サウンドチップ）とドライバの対応、詳しい設定方法などは、INSTALLドキュメントファイルの後半に、機種ごとに説明されている（**リスト1**）。ここでは、VIA Apollo Pro 133内蔵のAC'97コーデック（VIA 82C686A）を音源に使用する場合の設定例を示す（**リスト2**）。

/etc/modules.confファイルの編集が終わったら、

```
# modprobe sound-card-0
```

を実行して、ドライバがきちんとロードされるかどうかを確認する。ここで「sound-card-0」となっているのは、/etc/modules.confファイル中で定義したサウンドカードの別名である。正しくロードされれば、何もメッセージが表示されずに、コマンドが正常終了するはずだ。

ドライバのインストールが成功したら、次にライブラリとユーティリティのインストールを行う。ドライバの場合と同様に、アーカイブファイルを展開し、各ディレクトリ中で以下のコマンドを実行する。

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

いずれのパッケージも、展開したディレクトリ中の「INSTALL」というテキストファイルにインストールの方法が英語で解説されている。

●ドライバのテスト

ALSAドライバは、デフォルトの状態ではサウンドカードのすべての入出力の音量が0、かつ「ミュート」された状態になっている。このため、なにもしない状態では、ライン入力に接続したラジオの音は聞こえず、音声の録音や再生もいっさいできない。そこで、Xやコマンドライン用の「ミキサーアプリケーション」で音量を設定する必要がある。

ALSAドライバで利用できるミキサーは、OSS（Open

Sound System）互換のミキサーや、ALSAユーティリィに含まれるミキサーアプリケーションである。通常、Linuxのディストリビューションの多くにはOSS互換のX用ミキサーアプリケーション（xmixer）が付属するので、これを利用してもよいのだが、今回の「ラジオ録音システム」は無人運転が前提になるので、コマンドラインからの操作が可能なミ

Column

PCの雑音電波

PCの内部には、MHz単位の周波数で動作する電子回路が多数集まっており、そこから放射される「雑音電波」はテレビやラジオの受信障害を引き起こす可能性がある。実際、PCのそばでラジオを聞こうとして、雑音の多さに困惑した経験のある読者も多いだろう。このため、「PCでラジオの録音なんて、雑音がひどくて実用にならないのでは」と考えるのは当然だ。しかし、最近のPCは家庭内で使用することを想定して作られているので、以前に比べてラジオなどへの影響はずいぶんと少なくなっているはずだ。マザーボードやグラフィックカードといったパーツ単位でも、不要な雑音電波の放出はPC自身の誤動作を引き起こす原因ともなるので、厳密にコントロールされているはずである。

ところが、筆者が自動録音システム専用にマシンを組み立ててみたところ、ものすごい雑音がラジオに入るようになって、放送の内容を聴き取るのにもひと苦労という状態になってしまった。それまで自動録音の実験に別のPCを使っていたときには何の問題もなかったので、原因が新しく組み立てたマシンにあることは明白だった。

さっそく、原因の究明と対策にとりかかった。雑音の状態をいろいろと調べてみると、ハードディスクへのアクセスが行われているときに、雑音が大きくなるような気がしたので、IDEのフラットケーブルの引き回しを変えてみたり、ケーブル自体を取り替えてみたりしたのだが、効果はない。IDEのケーブルから雑音電波が放出される、などという話は聞いたこともなかったので、これは原因ではないようだ。

次に、ディスプレイケーブルを外した状態で電源を入れてみたところ、雑音が減ったことがあったので、ディスプレイケーブルかグラフィックアダプタが原因ではないかと考えて、ケーブルやグラフィックアダプタを別のものに取り替えてみた。しかし、症状に変化はない。

マザーボード裏側の「バックパネル」をつけないで、開けたままにしていたのが原因かとも思って、バックパネルを取り付けてみたものの、やはり効果はない。さらに市販の「ノイズフィルタコア」を、電源ケーブル、信号ケーブル、PC内部の電源ケーブルなどありとあらゆる「電線」に取り付けてみても、わずかにノイズが少なくなったかな、という程度の効果しかない。

そこで、問題はケースにあると考えて、PCのケースを別のものに交換してみたところ、ノイズはウソのように収まってしまった。雑音がひどかったときに使っていたケースは、国内の某社が販売しているものだったのだが、中身は正体のよくわからない台湾製の製品だった。しかし、ケース自体のつくりはしっかりしていたし、どこが悪かったのかよくわからない。すると、残ったのは「電源」だけということになる。

■怪しい電源に注意

筆者の専門は電子工学ではなく、PCの組み立て工作ができる程度の知識しかもっていないのだが、PCで多く使われている「スイッチング電源」というものが雑音源になるという話は知っていた。しかし、これについても最近の製品では対策がとられていて、少なくともラジオなどの放送電波に影響するようなことはないと信じていたのだ。しかし、問題のケースから電源部分を外して、試験的に通電してみたところ、見事（？）ラジオに雑音が入るようになったのだ。おそらく、電子工学の知識を持った人なら、こんな回り道はしないで真っ先に電源を疑うのだろうが、一般人としては、電源などというPCの中でもっともアナログ的な部分が悪いとは、なかなか想像できなかったのだ。

問題の電源装置をしげしげと眺めてみると、「ATX 2.01」に準拠しているということはわかるものの、製造国やメーカー所在地などが書かれていない「怪しい」ものであった。ブランド名と思しき文字列からWebで検索してみたところ、どうやらチェコにある某メーカーの製品らしい。ラベル部分をさらによく見てみると、「CB」、「CE」など、規格の呼称らしきものは標示されているのだが、その規格のどの部分に適合しているかなどの情報は一切ない。また、PC製品では必須ともいえる「FCC」や「VCCI」の規格標示はまったくなかった。

FCCにパスしていないからといって、即「悪い製品」と決めつけるわけにはいかないのだが、ひどい電波障害を起こすような製品が市中にあるのは良いことではない。この電源装置についてさらに調べてみたところ、一部のPCマニアの間では「音の静かな電源」ということで好評を博しているらしい。たしかにファンの音はきわめて静かで、筆者も最初は感心していたのだが、電波の世界からみると、これは「騒々しい電源」ということになる。雑音電波は自分だけでなく、近隣住民にも迷惑をかけることになりかねないので、注意が必要だ。しかし、電波は目に見えないし耳にも聞こえないので、結局、PCのケースや電源を買うときには、「FCC」や「VCCI」などの妨害電波防止規格にパスしているかどうかを基準に選ぶしかないようだ。

問題の電源
雑音がひどかったケースから外した電源。あえてメーカー名は伏せるが、ショップなどで見ると、安売りのケースに組み込まれて現在でも販売されているようだ。ファンの音は静かなのだが、雑音電波のレベルは尋常でないので、できれば使ってほしくない。

キサーアプリケーションを使おう。

ALSAユーティリティに付属するコマンドライン版ミキサーは「amixer」というコマンドで、コマンドラインから入出力の音量調整やキャプチャ対象となるポートの選択ができる。GUI版のミキサーに比べて操作は複雑になるが、よりきめ細かな設定ができるので、使い慣れるとこちらのほうが便利である。

とりあえず、ラジオのイヤホン端子とPCの「ライン入力」端子に接続し、ラジオのボリュームを適当な大きさ（8割程度のボリューム）に調整しておく。そして、コマンドラインから次のように入力して、ライン入力とスピーカー出力をそれぞれ50%程度の大きさに調整してみる。「Line」と「Master」の設定を済ませた時点で、スピーカーからラジオの音声が聞こえてくるはずである。

```
$ amixer set Line 50% capture unmute
$ amixer set Master 50% unmute
$ amixer set Gain 50% unmute
$ amixer set PCM 50% unmute
```

ここで、「set ～」は設定するポートの指定、「capture」は音声キャプチャを行うときにこのポートからの入力をキャプチャすることを指定、「unmute」はミュート状態を解除することを指定している（デフォルトではすべてミュート状態になっているので、ボリュームを0%以外の値に設定しても音声信号は入出力できない）。なお、サウンドカードの機種やドライバのバージョンによっては、「Gain」のボリューム設定ができない場合がある。このような場合は、「100% unmute」を指定する。

次に、ALSAユーティリティに付属する録音ツール（arecord）と再生ツール（aplay）を使って、音声の録音と再生が問題なくできるかどうかを試してみる。まず録音だが、arecordコマンドを使って10秒間だけラジオの音声をWAV形式で録音してみる。

▼リスト3　/etc/rc.d/rc.localに追加する行

```
/sbin/modprobe snd-card-via686a
/usr/bin/amixer -c 0 set Line 100 capture unmute
/usr/bin/amixer -c 0 set Master 50 unmute
/usr/bin/amixer -c 0 set "Input Gain" 50 unmute
```

サウンドカードドライバのロードと、ミキサーの音量設定を行う。音量設定については「仮」の値なので、本文中で説明する音量調整が終わったら、そこで決定した設定値に書き換えること。また、網掛けしてある部分は、各自が使用しているサウンドカードの機種に応じて、適切な名前に書き換える。

```
$ arecord -t 10 -m -w foo.wav
```

次に、この音声ファイルをaplayコマンドで再生してみる。

```
$ aplay foo.wav
```

PCのスピーカーから、録音したラジオの音声が出力されれば、ドライバやユーティリティ類のインストールや設定がうまくいっていることがわかる。もし音声が出力されなかったり、ノイズしか聞こえなかったりした場合は、インストールするドライバを間違えているか、/etc/modules.confの記述を間違えている可能性があるので、ALSAドライバのINSTALLドキュメントなどを熟読して、設定をやり直してみること。初歩的な間違いでは、ミキサーでの音量設定やキャプチャ先ポートの指定を忘れている場合が多いので、前述したように、amixerコマンドを使って「Master」（スピーカー出力レベル）、「Line」（ライン入力レベル）、「Gain」（録音レベル）、「PCM」（WAV出力レベル）のボリューム値とunmuteの設定、および「Line」のcapture設定を行ってみること。

●/etc/rc.d/rc.localとcrontabの設定（サウンドドライバの初期化）

以上でサウンドカード関連のドライバや基本ユーティリティのインストールは終わった。しかし、この状態では、マシンをリブートするたびにサウンドドライバの初期化やミキサーによる音量設定を行わなければならない。また、長時間運転し続ける場合には、他のアプリケーションに影響を与える可能性を減らすために、定期的に未使用のサウンドドライバを初期化することが、ALSAのドキュメント中では推奨されている（必ずしも必要というわけではない）。

このため、マシンの起動時にドライバの初期化と音量設定を行うためのコマンドを/etc/rc.d/rc.localファイルに記述し、定期的なドライバのアンロードを行うコマンドをrootアカウントのcrontabファイルに追加することにする。

まず、/etc/rc.d/rc.localファイルには**リスト3**のような行を追加する。ここでのボリュームの設定値は仮のものなので、最終的なレベルを決めたら、あらためてその値に書き直す。

crontabの設定は、suコマンドでrootユーザーになった状態で、

```
# crontab -e
```

を実行する。すると、vi（あるいはEDITOR環境変数で指定したエディタ）が自動的に起動し、その時点でのcrontabの設定内容が読み込まれるので、一番下の行に、次のような行を追加する。

```
10 0 * * * /sbin/modprobe -rs snd-card-0; /sbin/rmmod -as
```

この行は、「毎日午前0時10分に、snd-card-0ドライバのスタックをアンロードし、すべての未使用のドライバのアンロードを行う」という意味になる。

音量の調整

WAVファイルへの録音が可能になったところで、録音レベルの調整をしておこう。もともと、PCでのデジタル録音は、専用のオーディオ機器によるデジタル録音に比べて、音質もそれほど良くはなく、厳密な音量調整が必要になるわけではない。しかし、極端に録音レベルが高かったり低かったりすると、音が歪んだり、ノイズだらけで聞こえづらくなってしまう。

録音のレベル調整には、「レベルメーター」のようなものがあると便利なのだが、残念ながらLinuxでリアルタイムに録音レベルを表示できるようなソフトは見つからなかった。このため、レベル調整は、前述のarecordとaplayコマンドを使って、実際にラジオの音声の録音、再生を繰り返しながら、amixerコマンドで少しずつ調整するしかない。

適切な調整値は、使用しているラジオの機種、サウンドカードの種類、そして個人の好みに左右されるのではっきりした値を決めることは難しい（プロが作っている市販の音楽CDでも、音声レベルはタイトルによってまちまちである）。しかし、目標なしにレベル調整を試みても時間がかかると思うので、大まかな手順と目標値をあげておくことにしよう。

まず、ラジオ側のボリュームは、音量が最大になる位置を「10」とすると「8」くらいの位置になるように合わせておく。そして、PC側のスピーカー出力を最大（amixer set Master 100）、ライン入力（録音レベルではない）を80%（amixer set Line 80）にしておく。PCのスピーカーからは、かなり大きな音量で音声が流れ出すと思うが、もしここで音が歪んで聞こえるようなら、ライン入力の値を上下させてみて、歪みが気にならなくなる範囲で最大の値を調べる。ここで、ラジオ側のボリュームを調整してもよいのだが、一般に、ラジオ側のボリュームを小さくしすぎると、ノイズが多くなる傾向があるので、ラジオ側の音量は大きめにして、可能な限りPCのLine入力の値で調整する。それでも調整しきれないような場合に、ラジオ側の音量を少しずつ下げるようにすればよいだろう。

ライン入力の値が決まったら、そのときのだいたいの音量や音色を覚えておいて、いったん「amixer set Line mute」でライン入力からの音声出力をカットする。次に、「amixer set Line capture」で、録音入力をライン入力に設定し、「amixer set "Input Gain" XX」（XXの部分には、先ほど調べた適切なライン入力のレベル値を入れる）で録音レベルの設定をする。

次に、arecordコマンドとaplayコマンドで、録音と再生のテストをしてみる。

```
$ arecord -t 10 -m -w testrecord.wav
$ aplay testrecord.wav
```

aplayコマンドを実行したときに再生される音声を、先ほどライン入力を直接スピーカから聞いた場合の音声と比べてみる。再生音が大きかったり歪んでいるように感じられる場合には、Input Gainの値を下げ、小さすぎるような場合にはInput Gainの値を上げて、もう一度試してみる。なお、Input Gainで設定できる値は、サウンドカードの機種によっても異なるが、多くても32段階程度の設定しかできないので、1%単位での細かな値の調整は無意味である（1%単位で値を指定しても、amixerが自動的にもっとも近い段階の値に設定してしまう）。したがって、10%単位で値を上下させて、「妥協できる」値を見つけるのにとどめたほうがよい。また、サウンドカードによってはGainの調整が一切できないものもあるので、そのような場合は「amixer set "Input

画面1　XMMSの「スペクトラムアナライザ」
XMMSの情報ディスプレイ部分でマウスを右クリックし、[Visualization Mode] − [Analyzer] を選択すると、再生中の音声の周波数帯域ごとのレベルが表示される「スペクトラムアナライザ」モードになる。レベル表示のバーが常時振り切れていて、レベルの変化がわからないような状態だとレベルオーバー、スペクトル分布がだいたい判別できるような状態が適正レベル、というように判断することができる。ただし、このアナライザはあくまで「参考」程度にしかならないので、最終的には再生音を聞いて、自分の好みに合わせた設定を行うしかない。

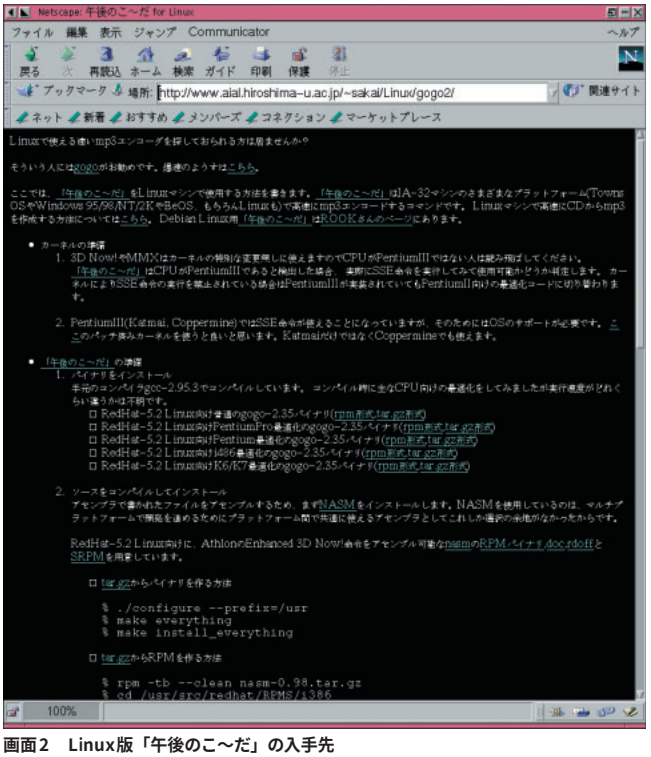
Gain" 100% unmute」に設定して、ラジオ側のボリュームで音量を調整するしかない。

X環境で作業できるのであれば、arecordで録音したデータを「スペクトラムアナライザ」機能をもったオーディオプレーヤ（xmmsなど）で再生してみて、録音レベルを目で確認してみてもよい（**画面1**）。ただし、オーディオプレーヤのスペクトラムアナライザは、厳密な音量レベルを表示してくれるわけではないので、あくまで目安にとどめておくこと。くどいようだが、1%単位での微妙なチューニングを目指しても無駄である。細かい音質の差を気にしすぎると精神衛生上よくないので、自分の聴覚を信じて「これなら耳あたりがいい」と思われる設定値を探すことを目標にすること。

arecordとaplayによる録音再生と音量調整を何回か繰り返せば、最適と思われる設定値が見つかるだろう。見つかった値は、先ほど設定した/etc/rc.d/rc.local内のamixerのコマンドラインに反映させておく。これで、Linuxを起動すれば、自動的に最適な音量レベルに設定されるようになる。

MP3エンコーダの準備

さて、以上でサウンドカードからの録音はできるようになったが、この状態ではWAV形式での録音しかできず、録音データのファイルは膨大なサイズになってしまう。そこで、音声データを圧縮してディスク容量を節約し、長時間あるいは長期間の録音に対応できるようにする。



画面2　Linux版「午後のこ〜だ」の入手先
(http://www.aial.hiroshima-u.ac.jp/~sakai/Linux/gogo2/)

音声データの圧縮技術はさまざまなものがあるが、現時点では、圧縮効率の高さ、エンコーダの入手性、プレーヤアプリケーションの対応状況などの点から、MP3（MPEG Audio Layer-3）以外の選択肢は考えられない。ところが、MP3に関しては、最近になって、開発元であるFraunhofer研究所とTHOMSON Multimedia社が、すべてのエンコーダに対して高額なライセンス料を要求する方針を発表しているため、いわゆる「フリー」のエンコーダは利用できなくなる可能性が高い。このため、将来的にはMP3に代えて、GPLでのライセンスが可能なOgg Vorbisなどのエンコード形式に切り替える必要があるだろう。ただし、本記事の執筆時点ではVorbisエンコーダはベータ版の状態であるため、記事中ではフリーのMP3エンコーダである「午後のこ〜だ」（PEN@海猫、へるみ両氏作のフリーソフトウェア）を利用することにした。

●午後のこ〜だの入手とインストール

「Linux版午後のこ〜だ」（以下「gogo」と略す）は酒居敬一氏のWebサイト（http://www.aial.hiroshimau.ac.jp/~sakai/Linux/gogo2/、**画面2**）から、ソースコードとバイナリファイル（Red Hat 5.2用のRPMファイルとtarボール）が入手できる。また、Debian用のパッケージについては、ROOK氏のサイトhttp://ya.sakura.ne.jp/~rook/linux/gogo.html（**画面3**）より入手できる（なお、「午後のこ〜だ」全体の一時配布元は、へるみ氏のWebページhttp://homepage1.nifty.com/herumi/である）。

gogoのインストールは、ソースファイルのコンパイルから始めることもできるのだが、特殊なアセンブラ（NASM）を別途インストールする必要がある。今回は、使用しているディストリビューションがTurboLinuxということもあって、バイナリファイルのRPMをインストールすることにした。

バイナリのRPMファイルは、使用するCPUに応じて最適化されたいくつかの種類があるのだが、通常は「普通の」という形容詞がつけられているバイナリをダウンロードすればよいだろう。Pentium IIIやCoppermine CeleronのSSE命令については、Linuxのカーネルさえ対応していれば、gogoのどのバイナリを使っても自動的対応してくれるようだ。

RPM形式のファイルを入手したら、インストールは簡単である。RPMファイルを置いたディレクトリで、

```
# rpm -i gogo-2.35-1.i386.rpm
```

を実行するだけだ。これで、/usr/binディレクトリにgogoというコマンドファイルがインストールされる。

●録音のテスト（その2）

gogoエンコーダの詳しい使い方の説明は省略するが、コマンドラインオプションでエンコード形式（ビットレート、サンプリングレートなど）を指定できる。この時点で、arecordコマンドでキャプチャしたWAV形式ファイルを、gogoエンコーダで問題なくエンコードできることを確認しておこう。

```
% arecord -t 10 -m -w testrecord.wav
% gogo -m -b 64 testrecord.wav
```

この例では、64kbps、モノラル形式でエンコードしてみた。エンコードの結果作成されるファイル（testrecord.mp3）を、xmmsやmpeg123などのオーディオプレーヤソフトで再生してみて、問題なく変換されていることを確認する。

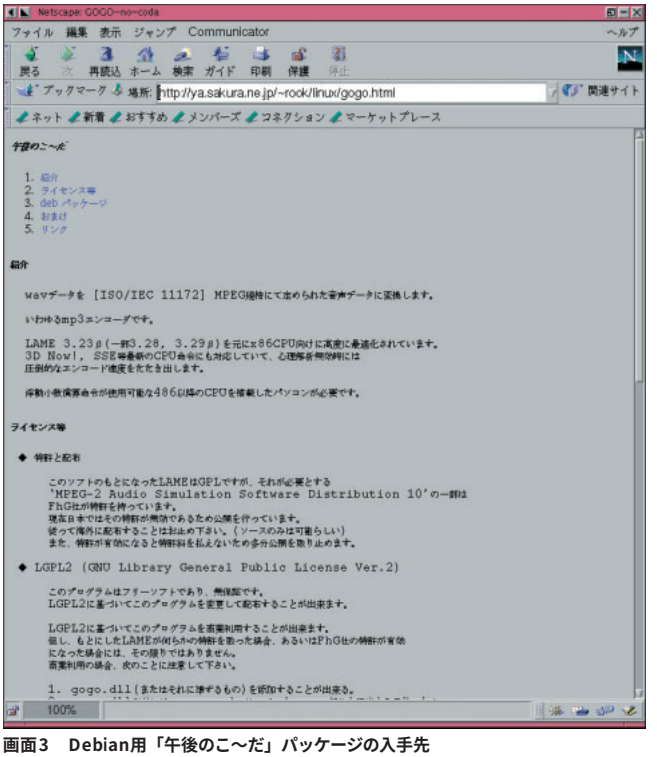
どの変換形式を採用するかについては、個人の好みや使用する目的によって異なる。とくに、シリコンオーディオプレーヤに転送して再生するような場合、MP3ファイルの形式によってはうまく転送できなかったり、一見転送できたようでも正しく再生できないこともある。たとえば、筆者はRCA製の「Lyra」というシリコンオーディオプレーヤ（日本では未発売）を使っているのだが、64kbps未満のmp3形式には対応していなかったり（転送時に強制的に64kbps形式に再エンコードされるので、転送に相当な時間がかかる）、Windows 2000上のWindows Media Player 7を経由して転送を行うと、見かけ上は転送できても、128kbps、44kHzサンプリング以外の形式だと正しく再生できないなどの問題がある。このため、Linuxマシンで64kbpsモノラル形式のMP3にエンコードして、プレーヤへの転送はWindows 98のマシンで行う、というような使い方をしているのだ。

シリコンオーディオプレーヤの多くは、このように、対応しているMP3のエンコード形式がかなり制限されているようなので、各自の環境で相当回数の実験が必要になるだろう。あるいは、これからシリコンオーディオプレーヤを購入しようと考えている人は、このような制限がどんな製品にもありうる点を知っておいたほうがよいだろう。余談だが、筆者所有のプレーヤ「Lyra」を発売しているRCAは、MP3技術のライセンス元であるTHOMSON Multimedia社配下のグループ企業なのだが、肝心のMP3ファイルの取り扱いに難点のある製品を出しているのはいただけない。

フロントエンドスクリプトの作成

さて、ライン入力から音声をキャプチャして、そのデータをMP3形式に変換できるようになったが、このままでは毎日決まった時刻に録音して、録音データを蓄積するような自動運転には不十分である。まず、crontabからキャプチャ動作を開始させるには、1つのコマンドでキャプチャとMP3への変換を一度に起動できたほうが面倒がない。また、キャプチャした音声のファイル名も、録音開始日時からファイル名を生成するようにすれば、録音後のファイルの整理が楽にできるようになる。

これらの要求を1つのコマンドで実行できるように、「radiorec」という名前の簡単なPerlスクリプトを作成した（**リスト4**）。radiorecスクリプトは、/usr/local/binディレクトリにコピーし、「chmod +x /usr/local/bin/radiorec」で実行属性をつけて、ユーザー権限で実行できるようにしておく。crontabからは、このスクリプトファイルをコマンドとして実行するようにした。最初にも述べたが、Perlを使って



画面3　Debian用「午後のこ〜だ」パッケージの入手先
(http://ya.sakura.ne.jp/~rook/linux/gogo.html)

▼リスト4　radiorecスクリプト

```
#!/usr/bin/perl

# radiorec -- record wav file from line input,
#             and convert it to 64K and 32K Mono MP3 file
# usage: radiorec -l <minutes> -f <file-prefix>
#      where: <minutes> specifies recording duration in minutes,
#           <file-prefix> specifies prefix for filename of result mp3 data
#           (path name is accepted, too). Date and time of the recording will
#           be added to the filename automatically.
#
# Copyright (c) 2000 by Zentaro Takeda
# "Personal use only" at this time

require "getopt.pl";

$encoder = "/usr/bin/gogo";
$wavrec = "/usr/bin/arecord";
$bash = "/bin/bash";

&Getopt('fl'); # f for filename, l for record length

if ($opt_f =~ /[^a-zA-Z0-9\/\-\_]/ ) {
    $recfile = "radio";
} else {
    $recfile = $opt_f;
}

if ($opt_l <= 0) {
    die "ERROR: Please specify length of recording in minutes.\n";
} else {
    $minutes = $opt_l;
}

$seconds = $minutes * 60;

($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst) = localtime();

$year = $year + 1900; # y2k
$mon = $mon + 1;

$timestring = substr("0000$year", -4) . substr("00$mon", -2) .
substr("00$mday" , -2) . substr("00$hour", -2) . substr("00$min", -2);

$recfile = $recfile . $timestring;

$commandline = "$wavrec -t $seconds -q -m -w $recfile.wav \&\>/dev/null;" .
"$encoder -b 32 -silent $recfile.wav $recfile" .
"_32K.mp3 2\>\>/tmp/radiorec.log;" .
"$encoder -silent -b 64 -delete $recfile.wav $recfile" .
"_64K.mp3 2\>\>/tmp/radiorec.log";

exec $commandline;
```

①

実行にはPerl 4.0以上が必要だが、ほとんどのLinuxディストリビューションには標準で付属している。スクリプトの先頭で、エンコーダ、キャプチャツール、シェルのフルパス名を指定しているので、場合によっては各自の環境に応じて書き換える必要がある（①）。

いるのに特別な理由はない。ただ単に筆者がPerlに慣れているだけのことである。同様のスクリプトはbashなどのシェルスクリプトでも記述できるはずだし、Perlで記述するにしても、もっとスマートに記述できるはずだ。このスクリプトについては、読者各自が自分の用途や好みに応じて、好きなように作成すればよい。

radiorecスクリプトの機能を簡単に説明する。-lオプションで録音時間（すなわち、録音しようとする番組の長さ）を分単位で指定し、-fオプションでファイル名の先頭につける「番組パス名」を指定する。ここには、録音ファイルを保存するディレクトリと、番組を識別するための文字列を指定できる。番組の部分はあまり長くしてもファイル一覧を表示するときにうっとうしくなるので、英文字2文字で識別できるようにしたほうがよいだろう。

```
$ radiorec -l 20 -f /home/zen-t/Language/English/ec
```

上のコマンドラインを実行すると、/home/zen-t/Language/Englishディレクトリ上でただちに録音が開始される。20分後に録音が終了すると、すぐにMP3エンコーダが呼び出されて、32kbpsと64kbpsの2つのMP3形式ファイルが作成される。エンコードの終了後、キャプチャしたWAV形式のファイルは削除される。2つの形式のMP3ファイルを作成している理由については後述する。作成されるファイルの名前は、次のような形式になる。

<番組名><年><月><日><時><分>_<ビットレート>.mp3

たとえば、2000年9月10日15時20分に「ec」という番組名で録音を開始すると、

```
ec200009101520_32K.mp3
ec200009101520_64K.mp3
```

の2つのファイルが作成される。日時データが数字で長々と並んでいて、人間の目にはちょっと分かりづらいファイル名になるのだが、今後、このファイルをさまざまなスクリプトで処理したり、WebページからCGI経由でアクセスできるようにする場合に好都合なので、このようなファイル名をつけている。

スクリプト内部の処理も複雑なものではない。コマンドが起動された時点の時刻をPerlのlocaltime()ライブラリ関数で

取得して、年月日、時分秒のデータに分解し、それをもとにファイル名を生成し、最後にキャプチャからmp3エンコーダ起動までの「コマンドライン文字列」を組み立てて、一気にシェルに渡しているだけである。余談だが、スクリプト中ではY2Kがらみの処理も行っている。Perlのlocaltime()関数では、西暦2000年が「100年」になってしまうからだ。

ここで、32kbpsと64kbpsの2つのバージョンのファイルを作っているのは、家庭内LANに外出先からダイヤルアップ接続して、いつでもどこからでも録音した番組を聞けるようにする際に、モデム経由での転送に適した「軽い」データも保存しておきたかったからだ。必要なければ、作成するMP3ファイルは1つだけにしてもよい。

録音の予約

コマンド1つで録音からMP3への変換までできるようになったので、あとは録音の予約をすればよいだけだ。予約にはLinuxのcron機能を利用する。これは、ユーザーが指定した時刻に指定したコマンドを実行する仕掛けである。Linuxのcronでは、月、日、曜日、時、分、秒などの時間単位を駆使して、起動のスケジュールを細かく指定できる。たとえば、「4月から9月まで、毎週月曜日から土曜日の午前6時45分」というような予約が可能なのだ。ラジオの語学番組などの録音予約を行うのに、もってこいである。また、cron機能は、予約の数は事実上無制限である。前述の「番組パス名」を工夫して、録音データを保存するディレクトリを分けたり番組を識別できる名前を付けたりして、複数の番組を予約録音することも簡単にできる。

cronの設定方法については、130～131ページでも少し触れたが、ここではrootユーザーではなく、一般ユーザーの権限で「crontab -e」コマンドを実行して予約のエントリを編集する。cronの予約では、1件1行になるように、次の順に予約データを指定する。

<分> <時> <日> <月> <曜日> <起動するコマンド行>

日、時、分の各データは数値で直接記述し、曜日については日曜日を0、月曜日を1のように、0～6の数値で指定する（6が土曜日になる）。また、月の値も1～12ではなく、1月は0、2月は1、……といった具合に、1月から12月を0～11の値で表す。

「,」（カンマ）で区切って複数の値を並べると、それぞれの値に合致したときにコマンドが起動される、また、「－」（ハイフン）をはさんで2つの数値を並べると、その範囲内に合致したときに起動される。たとえば、「月曜日から土曜日の毎日」という指定をしたければ、<曜日>のフィールドに「1－6」と記述すればよい。フィールドに値を指定しない場合には、「＊」（アスタリスク）を記述する。先ほどの例である、「4月から9月まで、毎週月曜日から土曜日の午前6時45分に、radiorecコマンドを起動する」というような場合、crontabのエントリ行は次のように記述する。

```
45 6 * 3-8 1-6 radiorec -l 15 -f /home/zen-t/Language/English/ec
```

同様にして、合計6つの番組を予約すると、crontabの状態は**リスト5**のようになる。

番組を予約する際には、予約時間が重ならないように注意すること。cronコマンドは指定された時刻に正確にコマンドを起動することしかしてくれない。たとえば、ある番組をradiorecで録音している最中に、別のcrontabエントリから別のradiorecを起動してしまうと、最初の録音が中断されたり、次の録音ができなかったりする（図3）。Linuxのサウンド機能では、同時に複数のキャプチャ動作はできないからだ。前の番組のキャプチャが終了して、エンコード処理に移っている段階で次の番組を録音することは、原理的には問題はないが、マシンの性能が低いとキャプチャデータに雑音が入ったり、音が途切れたりする可能性もある。時間が近接している番組を予約録音する場合には、事前に実験を行ってから予約のスケジュールを決めたほうがよいだろう。ntpなどを使ってマシンの時計を合わせておくことも必要だ。

▼リスト5　合計5つの語学番組を予約した場合のcrontabの状態

```
45 6 * * 1-6    /home/zen-t/mp3/radiorec -l 15 -f /home/zen-t/Language/English/ec
25 15 * * 1-6   /home/zen-t/mp3/radiorec -l 20 -f /home/zen-t/Language/English2/ed
40 22 * * 1-6   /home/zen-t/mp3/radiorec -l 20 -f /home/zen-t/Language/Business/be
20 13 * * 1-6   /home/zen-t/mp3/radiorec -l 20 -f /home/zen-t/Language/French/fr
0 8 * * 1-6     /home/zen-t/mp3/radiorec -l 20 -f /home/zen-t/Language/Spanish/sp
```

NHK第2放送の「英会話入門」、「英会話」、「やさしいビジネス英語」、「フランス語講座」、「スペイン語講座」の5つの番組を予約してみる。それぞれの番組は、1日に2回ほど放送されるので、予約スケジュールが隣接しないように組み合わせている。

制限事項と今後の課題

使用するサウンドカードの機種にもよるのだが、録音をしている最中には、同じマシン上での音声ファイルの再生は行えない場合が多い。逆に、録音開始時刻に音声の再生を行っていると、録音に失敗することがある。サウンドカードのハードウェアの制約から、音声の録音とデジタルデータの再生（エンコードとデコード）は同時に行えないからだ。

高級なサウンドカードの中には、エンコード部とデコード部が分かれているものもあるようだが、そのようなカードがLinuxでも使えるかどうか、そして、録音と再生が本当に同時にできるかについて、筆者は調べていない。

ALSAドライバでは、1台のマシンに複数のサウンドカードをインストールして、同時に使えるような設定もできる。そのような構成にすれば、再生と録音を1台のマシンで同時に行うことも可能かもしれないが、カード同士の相性の問題もあるだろうし、ドライバのインストールや設定がかなり難しくなると思われるので、ここでは取り上げない。これらの問題については、今後の課題ということにしたい。

いずれにせよ、カセットデッキやビデオデッキで放送を録音・録画するときも、録画中は再生できないのだから、それと同じようなものと考えれば納得がいくだろう。それでも、LANを使って複数のマシンを動かしているような環境なら、SambaやNFSなどのファイル共有の仕組みを使って、録音中に別のマシンから録音データを読み出して再生することは問題なくできる。

録音したデータをCD-Rなどに保存したり、録音済みの番組を、家族が自分の好きな時間に再生できるようにしたいのであれば、LANによるファイル共有が便利である。このような場合、クライアントマシンはWindowsマシンが使われることが多いだろう。WindowsマシンとLinuxの間のファイル共有は、Sambaを使えば簡単かつシームレスに行える。Sambaの設定や運用については、別の記事などを参考にしていただきたい。EthernetのLANならば（10Mbpsか100Mbpsかを問わず）、LinuxマシンのMP3ファイルをWindowsマシン上のサウンドプレーヤで直接再生してもまったく問題ない（大量のファイルをLAN経由でコピーするなど、極端にLANが混雑している場合には音が途切れることもあるが）。

しかしながら、エクスプローラなどでディレクトリを開き、複雑なファイル名から番組の種類と日時を判断しなければならないので、PCの扱いになれていないユーザーにはつらいかもしれない。そこで、後編では、この自動録音システムを発展させて、CGIを使ったWebページから、録音済みの番組に簡単にアクセスできるようなシステムを作ってみる。

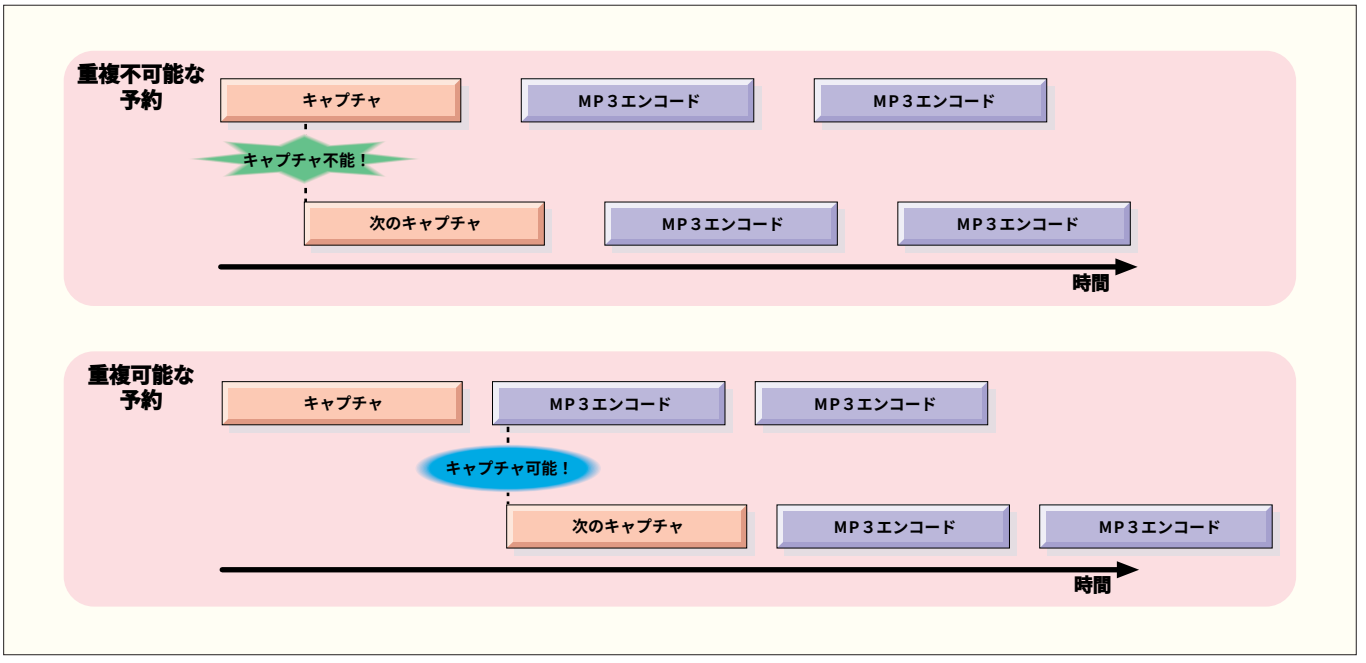


図3　番組予約が重複する場合
1つの放送局の番組を予約するのであれば、予約時間が重なることはありえないのだが、番組の長さ（radiorecスクリプトの-lオプションの数値）を間違えたり、連続する番組を予約したりすると、前の番組のキャプチャ終了前に次のキャプチャを開始されてしまう場合がある。キャプチャが終了してエンコード段階に入っている場合は、次のキャプチャを開始しても問題ない（ただし、マシンの性能が低い場合には支障が出る可能性もある）。

Column

Linuxのサウンドドライバ

現在、主流となっているLinux用サウンドドライバには、大きく分けて3つの種類がある。Linuxカーネルといっしょに配布されているもの、商用ドライバである「OSS」、ボランティアによる開発が行われていて、フリーで流通している「ALSA」の3種類だ。

・カーネル標準のサウンドドライバ

カーネルといっしょに配布されているサウンドドライバは、カーネルの作成（再コンパイル）時に、ドライバを選択してカーネル自体に組み込むようにするか、あるいは必要時に動的にロードされる「ローダブルモジュール」としてコンパイルして、設定ファイルの記述に従ってロードするかを選べるようになっている。

カーネル標準のドライバは設定が面倒という欠点があるが、現在、多くのディストリビューションでは、サウンドドライバをローダブルモジュールとして動的に組み込むようにしており、ディストリビューション固有のツール（Red Hatのsndconfigなど）を使って設定できるようになっている。

実は、現在の「カーネル標準サウンドドライバ」は、「OSS/Free」という別名で呼ばれることもあるとおり、4Front Technologies社でOSSドライバの開発に携わっているHannu Savolainen氏がその大部分を開発した。現在でも、OSSから「無償」のドライバの寄付が行われており、対応サウンドカードは徐々に増えている。

・OSS

OSS（Open Sound System）は、米国の4Front Technologies社が開発し配付しているサウンドドライバキットで、対応するサウンドカードの種類によって、無償で使用できるものと、使用料金が必要になるものがある。一時期は、カーネルが標準でサポートしているカードの種類が少なかったこともあって、商用ディストリビューションの中にはOSSドライバをバンドルしていたものがあったが、最近では、カーネル標準のドライバが充実してきたこともあって、OSSのバンドルを取りやめたディストリビューションもある。

OSSドライバは、カーネル標準のドライバと互換性があり、カーネルのドライバを使用するように設計されたアプリケーションは、OSSのドライバでも動作することになっている（というよりは、前述したようにカーネル標準ドライバとOSSは出自が同じである）。

OSSのドライバは、カーネル自体にサウンドドライバが組み込まれている場合には使用できないが、ローダブルモジュールとしてロードするような設定になっている場合には共存できる。前述のように、現在のディストリビューションの多くは、サウンドドライバをローダブルモジュールとして使用しているので、多くの場合、カーネルの再コンパイルなどの必要なしに、OSSドライバをインストールできるはずだ。

インストール作業は簡単で、配付パッケージ自体がLinuxの実行ファイル（ELF形式のバイナリファイル）になっており、これを直接実行して画面上の指示に従えばよい。

・ALSA

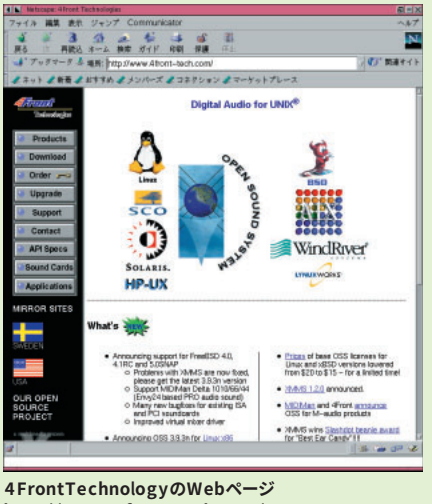
Jaroslav Kysela氏（チェコ在住）がメインプログラマーとして活躍する、ボランティアプロジェクト「ALSA Project」が作成、配付しているフリーのLinux用サウンドドライバキットがALSA（Advanced Linux Sound Archtecture）である。

「商用」であるOSSとは異なり、GPLによる再配布が可能になっている。このため、サウンドカード（あるいはサウンドチップ）のメーカーが無償での情報公開に応じなかったり、情報公開にあたってNDA（非開示契約）を要求するような製品についてはサポートされない。

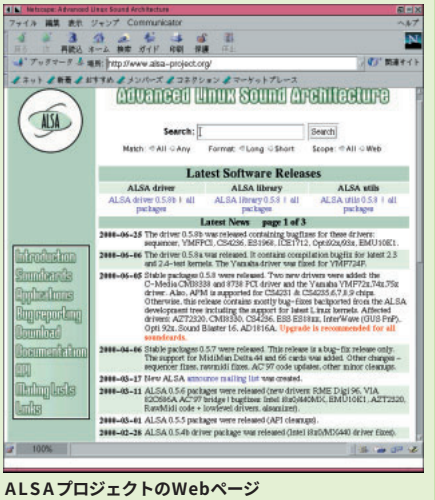
カーネル標準ドライバやOSSとの互換性があり、これらのドライバ用に作成されたアプリケーションのほとんどはALSAドライバでも利用できる。

本文中で解説しているように、インストールについては若干面倒な作業が必要になる。しかし、インストールや設定を簡単に行えるようにする「ALSA Manager」の開発が予定されており、この問題については早晩に解決されるだろう。また、「録音と再生が同時にできない」という点に関しても、全二重対応のサウンドカード（録音用エンコーダ部と再生用デコーダ部が独立しているカード）について、徐々に対応を進めていくとしている。

sndconfig
Red Hatのサウンド設定ツール



4FrontTechnologyのWebページ
http://www.4front-tech.com/

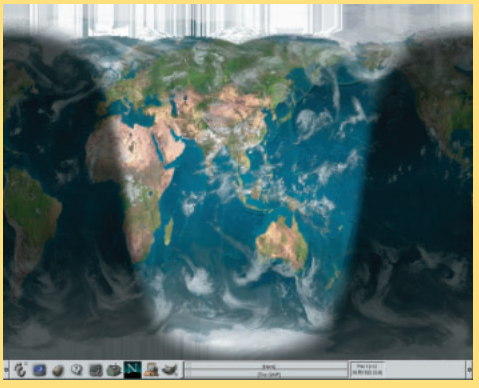


ALSAプロジェクトのWebページ
(http://www.alsa-project.org/)

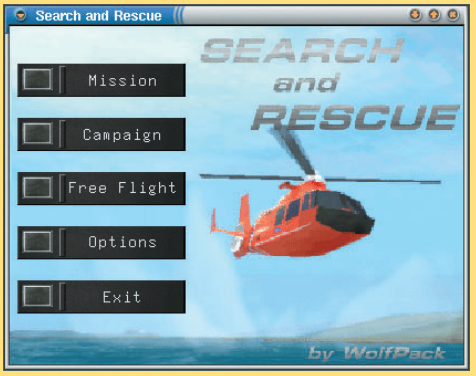
Free Application Showcase

文：出井 一
Text : Hajime Dei

XFireworks　P.132



XPlanet　P.134



WolfPack's Search and Rescue　P.136

※紹介したソフトは、すべて付録CD-ROMに収録されています。

パネルにメモやニュースなどを表示する多機能ソフト

MemoPanel

バージョン：6.9　　ライセンス：GPL

http://kano.technolust.cx/memopanel2.html
http://kano.technolust.cx/HtmlHeadLine.html（HtmlHeadLine.sh）

MemoPanelは、GNOMEパネルにメモを貼りつけるアプレットだ。文字や背景の色を自由に指定して、複数行にわたるメモを貼りつけられる。一定時間ごとに表示を更新する機能を利用してカレンダーや時計代わりに使えるほか、メールチェックや画面キャプチャも可能だ。同じ作者の「HtmlHeadLine.sh」と組み合わせることで、取得したニュースを順次ヘッドラインニュースとしてパネルに表示できる。

●ビルドから起動まで

MemoPanelは、ファイル一式をtar＋gzipしたtarボールのみ配布されている。configureスクリプトは用意されていないが、ビルドとインストールは、「make」→「make install」とするだけでいい。

コマンドラインで「memopanel&」とするか、パネル上で右クリックし、ポップアップメニューの[アプレット]－[ユーティリティ]－[MemoPanel]を選択すると、パネル上に起動した日時のメモが表示される（**画面1**）。

MemoPanelを複数起動して、パネル上に複数のメモを置くことも可能だ。また、設定内容は自動的に保存されるため、次にGNOMEを起動した際には、パネルの同じ位置に同じ内容のメモが表示される。

●メモの内容を設定する

メモの内容を変更するには、メモ上で右クリックし、メニューから[プロパティ]を選択しよう。MemoPanelに関するさまざまな設定を行うプロパティダイアログが開く（**画面2**）。MemoPanelの多機能さを反映して設定項目は多岐にわたっており、それぞれのジャンル別にタブ付きページで分類されている。ここでは、基本となるメモ機能について説明しよう。

通常、メモの内容は[メモ1]に設定する。メモには日本語を利用可能だ。複数行にわたるメモを書くには、改行したい位置に「%n」を書けばいい。たとえば「12:30%nそばいち」と書くと、1行目に「12:30」、2行目に「そばいち」と表示される。

また、2種類のメモを切り替えて表示することもできる。[メモ1]と[メモ2]のそれぞれに内容を設定し、[ディレイ（秒）]に表示間隔を設定すればいい。なお、ディレイが0のまま2つのメモを設定すると、[OK]や[適用]ボタンを押すたびに表示が切り替わるので注意されたい。

このほか、メモに使われる文字や背景の色、フォント（英語と日本語の2種類）などを、それぞれのメモごとに独立して設定できる。メモの内容や重要性などに応じて使い分けるとよいだろう（**画面3**）。

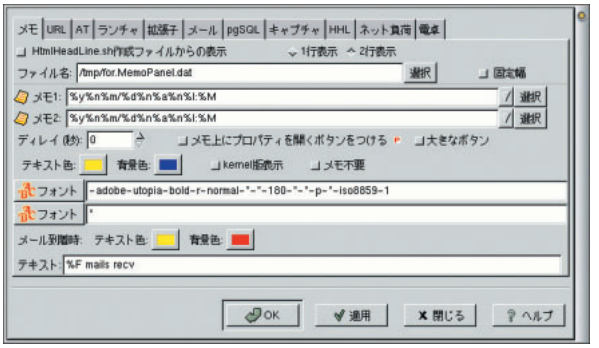
●現在の日付・時刻を表示する

MemoPanelを起動すると、**画面1**のように作成（あるいは更新）日時が表示されるが、プロパティダイアログで

(1) 時刻関連	
%H	24時間制の時間（00-23）
%k	同上（0-23）
%I	12時間制の時間（01-12）
%l	同上（1-12）
%M	分（00-59）
%S	秒（00-61）
%p	ロケール対応のAM／PM表記
%r	12時間制の時刻
%T	24時間制の時刻
%X	ロケール対応の時刻表記
%Z	タイムゾーン
(2) 日付関連	
%y	下2桁の年（00-99）
%Y	4桁表記の年（1970-）
%m	月（01-12）
%b	ロケール対応の月名（省略形）
%B	ロケール対応の月名（完全形）
%d	日（01-31）
%a	ロケール対応の曜日名（省略形）
%A	ロケール対応の曜日名（完全形）
%J	日本語の曜日名（※独自拡張）
%D	日付
%x	ロケール対応の日付表記
%c	ロケール対応の日付・時刻表記
(3) その他	
%n	改行
%%	%自身

リスト1　strftime形式の文字列（一部抜粋）

画面1
GNOMEのパネルに日付と時間を示すメモが表示される。



画面2
プロパティダイアログの[メモ]ページでメモの内容などを設定。

メモの内容を確認すると、実際の日時の代わりに「%y%n%m／%d%n%a%n%l:%M」という文字列が設定されている。

これらは、dateコマンドで用いられるstrftime形式の文字列で、主に日時に関する置換を行うためのものだ。MemoPanelで有用と思われるstrftimeを**リスト1**に抜粋した。

たとえば、メモを書く際にいちいち今何時か調べないでも、「%r 笹北さんから電話…」などと書いておけば、「10:31:48 AM 笹北さんから電話…」のように、メモを作成した時間に自動的に置き換えてくれる。

さらに、一定時間ごとにメモの表示を更新するディレイ機能と組み合わせることで、MemoPanel単独でカレンダーや時計アプレットの代わりに使うことができる。

たとえば、[メモ1]に「%x%n%r」、[メモ2]は空欄、[ディレイ(秒)]を1に設定すると、1秒ごとに表示が更新され、そのたびにメモの内容である日付や時刻が更新される（**画面4**）。メモの内容に使われるstrftimeをカスタマイズすれば、自分好みの表示に変更できるし、[メモ1]に日付、[メモ2]に時刻のstrftimeを設定して、両者を切り替えて表示することも可能だ。

●指定時刻にメールや音楽を

MemoPanelは、重要なメモをユーザーが見逃さないように、atコマンドを利用して指定時刻に自分自身にメールを送信したり、音楽を演奏する機能を備えている。なお、メールの送信にはsendmailを利用するため、sendmailが正しく動作するようあらかじめ設定しておこう。

この機能に関する設定は、プロパティダイアログの[AT]ページで行う（**画面5**）。時刻の設定はatコマンドと同様で、絶対指定だけでなく、「now + 3 minutes」（今から3分後）のような相対的な指定も可能だ。

メールを利用する場合は、宛先・件名を設定後、[Eメールをあなた自身に送信予約する]ボタンを押す。また、音楽を演奏する場合は、プログラム（mpg123など）と演奏するファイルを指定して[音楽を演奏予約する]ボタンを押せばいい。なお、mpg123のコマンドラインには、余計な文字列を出力しない−qオプションを付けておこう。

どちらの場合も、予約ボタンを押した時点でatコマンドのキューにメール送信や音楽再生用のコマンドが登録され、指定時刻になると実行される。後述のメールチェック機能と組み合わせると、メールが届いた時点でメールクライアントを起動できる。

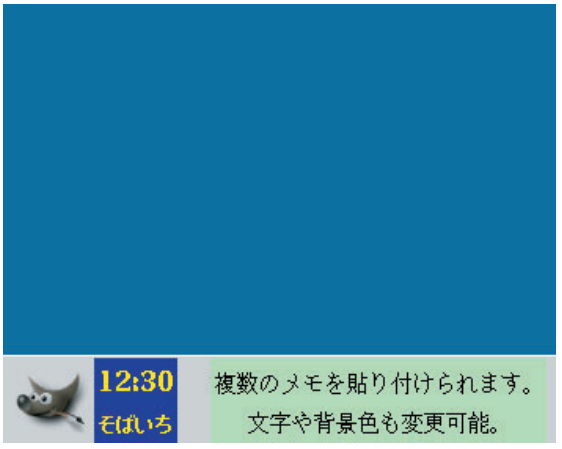
●メールチェッカーとして使う

MemoPanelをメールチェッカーとして利用すると、一定時間ごとに指定されたメールサーバをチェックして、到着したメールの数をメモとして表示する。さらに、自動的にメールクライアントを起動して、それらのメールを受信することも可能だ。

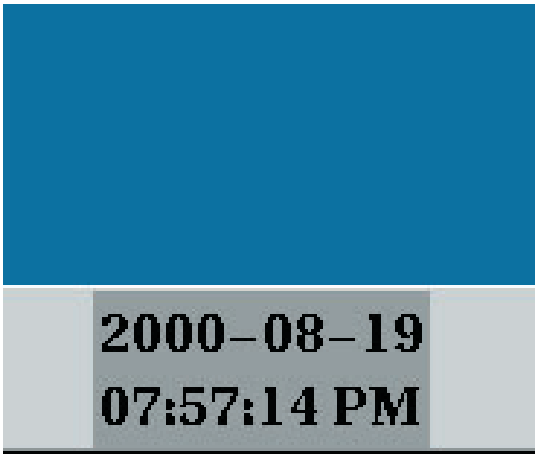
この機能に関する設定は、プロパティダイアログの[メール]ページで行う（**画面6**）。メール用の[サーバー]と[ユーザーID]、[パスワード]を設定し、[メールチェック]をチェックすればいい。チェックの間隔は、このページの[ディレイ(秒)]で設定する。初期値は180秒になっている。

指定した秒数が経過するか、メモを右クリックして[直ぐにメールをチェックする]を選択すると、メールボックスがチェックされ、パネルに「6 mails recv」とメール数が表示される（**画面7**）。しばらくすると、メモの表示は元の内容に戻る。

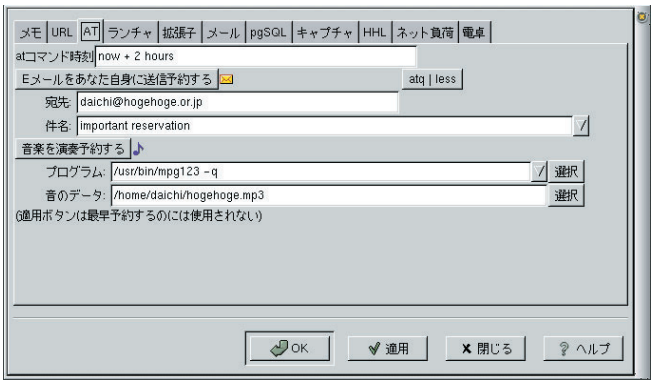
メールクライアントと連動させるには、プロパティダイアログの[新メール着信時起動コマンド]に、メールクライアントを起動するコマンドラインを設定する。メールクライアント側の設定



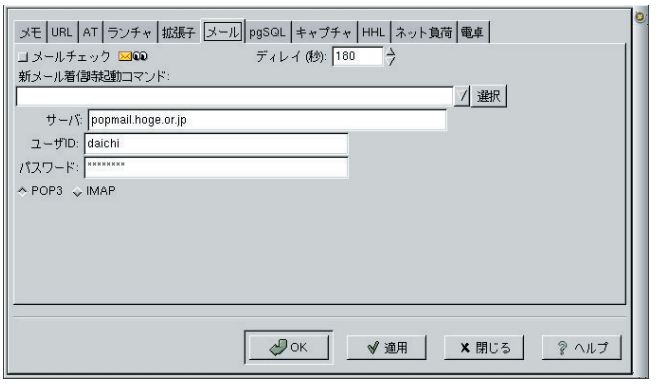
画面3
複数のメモで異なる色やフォントを利用できる。



画面4
strftime文字列とディレイ機能の組み合わせで時計に変身。



画面5
指定時刻に音楽を演奏したり、メールを送信することも可能だ。



画面6
一定間隔でメールの到着を調べるメールチェッカーとしての設定。

により、そのままメールの受信を行うことも可能だろう。

●HtmlHeadLine.shと組み合わせる

MemoPanelでは、ファイルの内容を先頭から順に1、2行ずつ、時間をおいて表示する機能が用意されている。この機能は、同じ作者の「HtmlHeadLine.sh」と組み合わせて、MemoPanelをニュースティッカーとして利用するために用意されたものだ。

HtmlHeadLine.shは、freshmeatなど多数のニュースサイトから情報を取得し、自動的にヘッドラインニュースを作成するシェルスクリプトだ。インストーラは付属しないので、HtmlHeadLine.shとSITES.tableを/usr/local/binに手動でコピーしよう。

なお、実行にはネットワークライブラリの「GNet」（http://www.eecs.umich.edu/~dhelder/misc/gnet/）が別途必要となる。

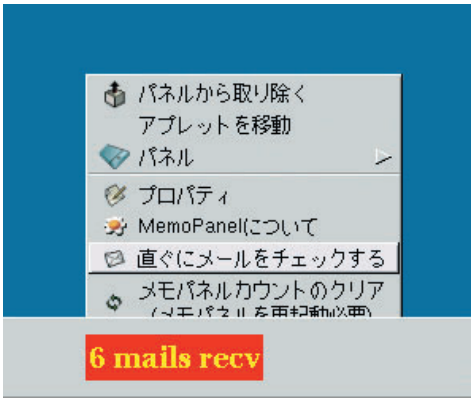
初期設定では、その名の通りHTML形式でヘッドラインを作成し、Netscapeで表示する。MemoPanelから利用できるプレーンなテキストを作成するには、環境変数FILE_ONLYに「YES」を設定すればいい。

なお、情報を取得するニュースサイトは、SITES.tableに列挙されている。初期設定では、155ものニュースサイトのうち、「END」の前に書かれた5サイトが取得対象だ。興味のあるサイトがあったら、その記述を「END」の前に移動しておこう。

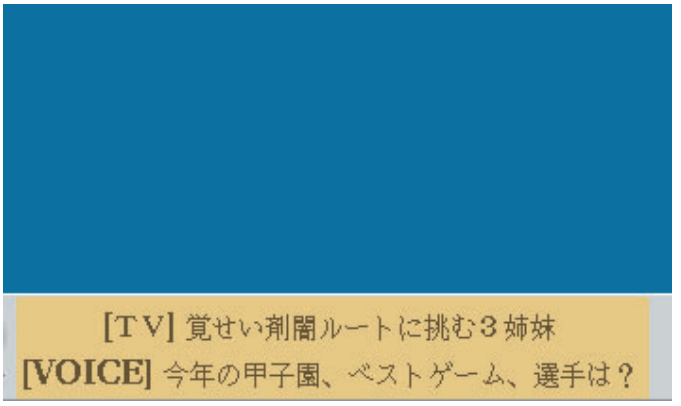
HtmlHeadLine.shの起動には、MemoPanelのランチャ機能を利用する。プロパティダイアログの[ランチャー]ページで、[コマンド]のリストから「export FILE_ONLY=YES;HtmlHeadLine.sh」を選択して、右端の[開始]ボタンを押せばいい。

取得した情報をMemoPanelで表示するには、プロパティダイアログの[メモ]ページで、[HtmlHeadLine.sh作成ファイルからの表示]をチェックし、[ディレイ(秒)]に短めの値（5など）を設定する。これで、ヘッドラインの内容が5秒ごとに2行ずつメモとして表示されるようになる（**画面8**）。

なお、HtmlHeadLine.shを一定時間（たとえば1時間や30分）ごとに繰り返し実行すれば、ヘッドラインの内容を常に最新の状態に保てる。同梱のシェルスクリプトSTART_HHL_30.shをバックグラウンドで実行するのが最も簡単な方法だ。ただし、スクリプトの設定部分で「export FILE_ONLY=YES」とする必要がある。



画面7
到着したメールの数がこのようにメモに表示される。



画面8
HtmlHeadLine.shで取得した情報をヘッドラインとして表示。

ルートウィンドウを華麗な花火で彩る

XFireworks

バージョン：1.3　　ライセンス：GPL

http://www.seki.ee.kagu.sut.ac.jp/~sakai/myfreesoft/

XFireworksは、Xのルートウィンドウ（背景）にさまざまな種類の花火を表示するソフトだ。用意されている花火は多種多様で、思わず長時間見入ってしまう。ユーザーが新たな花火を追加することも可能だ。バックグラウンドで常用するにはCPUへの負荷が気になるところだが、花火の上がる頻度や残像の度合い、グラデーションの色数などを調整できるので、自分のマシンの能力に合わせて楽しめる。

●ビルドとインストール

XFireworksはファイル一式をtar＋gzipしたtarボールのみ配布されている。configureスクリプトは含まれておらず、設定の変更はMakefileなどを直接書き換えることで行う。

もっとも、一般的なLinuxのディストリビューションでは設定を変更する必要はなく、そのまま「make」→「make install」とすればいい。/usr/X11R6以下の各ディレクトリに実行ファイル・マニュアル・設定ファイルがそれぞれインストールされる。

●バックグラウンドに花火を表示

ktermなどのコマンドラインで、

```
$ xfireworks
```

とすると、ルートウィンドウにさまざまな花火が表示される（**画面1**）。

XFireworksの動作設定はコマンドラインオプションで行う（**リスト1**）。たとえば、GNOMEのファイルマネージャgmc（GNOME Midnight Commander）を常用している場合、そのままだとXFireworksの画面表示が遅くなったり、ちらついたりするので、-no-direct-drawオプションを指定して、

```
$ xfireworks -no-direct-draw
```

とするといい。ルートウィンドウに直接描画する代わりに、いったんピックスマップに描画してからコピーするため表示がスムーズになる。

●ハイエンドマシンの場合

処理能力に余裕のある最近の速いマシンならば、動きの細かさを指定する-fineオプションと、残像の度合いを指定する-after-imageオプションを組み合わせて、

```
$ xfireworks -fine 200 -after-image 150
```

とすると、より美しい表示が得られる（**画面2**）。数値の初期値はいずれも100（%）なので、この例では動きが倍細かくなり、残像が5割増しになる。

さらに、ハイカラー・フルカラー表示環境では、グラデーションの色数のベースを指定する-gradationオプションを使って、

```
$ xfireworks -fine 200 -after-image 150 -gradation 64
```

としてみよう。数値の初期値は16で、16×16×16＝4096色が使われるのに対し、この例では64×64×64の約26万色が使われるようになる。

なお、バックグラウンドジョブとして常用する場合は、ウェイトを入れる-waitオプションを追加して、

```
$ xfireworks -fine 200 -after-image 150 -gradation 64 -wait 10000&
```

などとすると負荷が低くなる。数値はマイクロ秒単位で、この例では0.01秒のウエイトが入る。

-h	ヘルプメッセージを表示
-display [名前]	使用するXサーバ名を指定
-bg [色名]	ルートウィンドウの背景色を指定
-f [ファイル名]	設定ファイルを指定
-wait [時間]	ウエイト時間を指定（μ秒単位）
-fine [値]	動きの細かさの度合いを指定（%単位）
-gradation [値]	グラデーション色数のベースを指定（初期値16）
-direct-draw	ルートウィンドウに直接描画（初期値）
-no-direct-draw	ピックスマップに描画してからコピー
-probability [値]	花火の打ち上げ頻度を指定（%単位）
-size [値]	花火の大きさの度合いを指定（%単位）
-air [値]	空気抵抗の度合いを指定（%単位）
-gravity [値]	重力の大きさの度合いを指定（%単位）
-transmission [値]	速度の変わる度合いを指定（%単位）
-after-image [値]	残像の度合いを指定（%単位）
-color-length [値]	色の変わる度合いを指定（%単位）
-next-power [値]	花火の爆発力の度合いを指定（%単位）
-next-number [値]	花火の数の度合いを指定（%単位）

リスト1 オプション一覧

●ローエンドマシンの場合

処理能力に余裕のないローエンドマシンでは、ハイエンドマシンとは逆に、動きの細かさや残像の度合いを下げればいい。たとえば、

```
$ xfireworks -fine 75 -after-image 75
```

とすると、どちらも25%カットになり、それだけ動作が速くなる。

また、256色表示環境で使っているなら、グラデーションの色数も減らしたほうがいい。たとえば、

```
$ xfireworks -gradation 5
```

とすると、5×5×5＝125色だけ使われるようになる。

このほか、花火が上がる頻度自体を下げることも可能だ。-probabilityオプションを使って、

```
$ xfireworks -probability 50
```

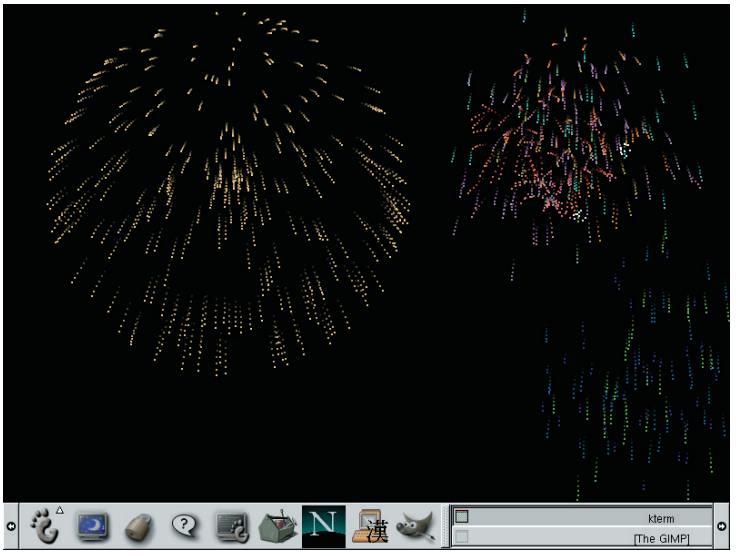
などとすればいい。数値の初期値は100（%）なので、この例では花火が上がる頻度が半分になる。

こうした設定を組み合わせれば、ローエンドマシンでもある程度スムーズな表示を行えるだろう（**画面3**）。

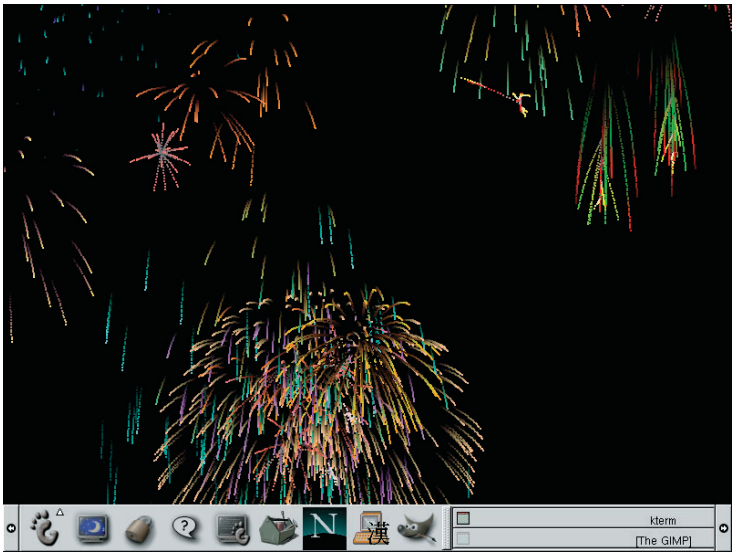
●カスタマイズも可能

XFireworksは、起動時に設定ファイル（xfireworks.conf）を、「カレントディレクトリ」→「/usr/X11R6/etc」の順で探して読み込む。この設定ファイルには、さまざまな花火のパラメータや次の花火への繋がり方がテキスト形式で記述されており、ユーザーが新たな花火を追加することも可能だ。

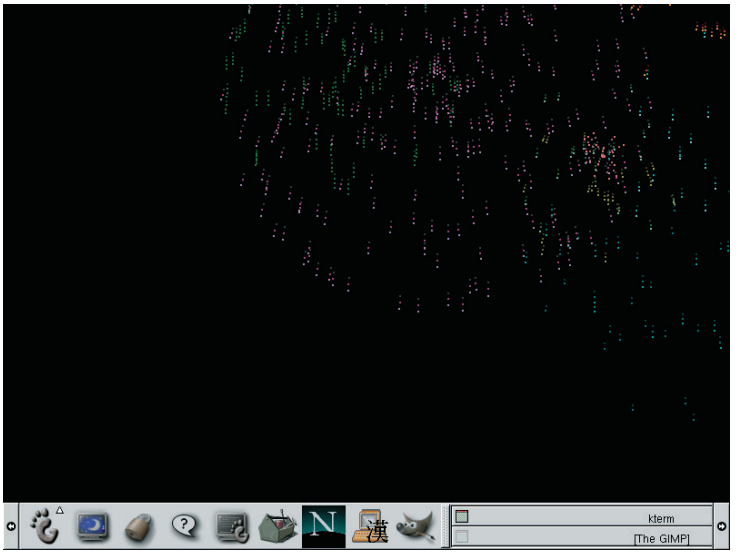
画面1
ルートウィンドウにさまざまな色や形の花火が表示される。



画面2
速いマシンでは細かさや残像の度合いを上げるといい。



画面3
遅いマシンでも設定次第でスムーズに動作するはずだ。



ルートウィンドウにリアルな惑星を表示

XPlanet

バージョン：0.73　　ライセンス：GPL

http://xplanet.sourceforge.net/

XPlanetは、Xのルートウィンドウ（背景）に地球をはじめとする惑星や月の画像を表示するソフトだ。地表全体をカバーするマップ画像を利用して、平面や球面に投影されたリアルな地球などを表示する。一定時間ごとにXPlanetを呼び出して再描画したり、設定をGUIで行う関連ソフトも同梱されている。また、通常のウィンドウに表示してアニメーションさせることも可能だ（Mesaライブラリが必要）。

●ビルドとインストール

XPlanetはファイル一式をtar＋gzipしたtarボールとRPMパッケージの両方で配布されている。Red Hat系ディストリビューションではRPMのバイナリ（またはソース）パッケージを使うといいだろう。この場合、アニメーション機能が標準で組み込まれるため、Mesaが必須となる。

Mesaの使えない環境では、tarボールを利用しよう。「./configure --disable-animation」として、アニメーション機能を組み込まない設定にすればいい。あとは、「make」→「make install」とすれば、ビルドとインストールが行われる。

●各惑星のマップ画像を入手

マップ画像は同梱されていないので、**リスト1**のWebサイトから別途取得する。JPEGやPNG、TIFFなど各種の画像形式に対応している。なお本誌CD-ROMには、「XGLOBE & XPLANET MAPS」提供の地球のマップ画像（earth_400.jpg、earth_800.jpg、earth_1600.jpgの3サイズ）を収録した。

ルートウィンドウと同程度のサイズのマップ画像を使うと生成される画像の質が良くなる。あまり大きな画像を扱うと実行時に大量にメモリを消費するので、GIMPなどで適当なサイズに縮小してから使うとよいだろう。

取得・作成したマップ画像は、/usr/share/xplanet（tarボールを利用した場合は/usr/local/share/xplanet）に置く。また、各惑星のマップ画像のうち、よく利用するものを「惑星名.拡張子」（「earth.jpg」など）というファイル名に変更しておく（シンボリックリンクでも可）と、いちいちコマンドラインでファイル名を指定する手間が省ける。

●オプションで画像などを指定

XPlanetでは、コマンドラインオプションによりさまざまな設定を行う。たとえば、「xplanet -i earth_800.jpg -p or -markers」とすると、しばらくたってからXのルートウィンドウ（背景）に円形の地球の画像が表示される（**画面1**）。

－iオプションはマップ画像のファイル名を指定するもので、省略した場合にはearth.jpgが使われる。

－pオプションは表示図法を指定するオプションで、上の例では「正射投影図法」（or）による円形の表示が行われる。このほか、「矩形図法」（re）・「メルカトル図法」（me）・「モルヴァイデ図法」（mo）・「正距方位図法」（az）による表示も可能だ。－pオプションを省略すると、矩形図法による平面表示となる（**画面2**）。

－markersオプションは各地の地名（マーカ）を表示するもの。なお、XPlanetには、地球・火星・月のマーカデータが付属している。

このほかにも、表示の中心となる緯度／経度を指定する（－lat、－lon）、雲の画像を重ねて表示する（－cloud_i）、地球以外の惑星や月を表示する（－bo）といった設定も可能だ。

なお、XPlanetのWebサイトでは、世界各地の気象衛星（「ひまわり」を含む）のデータから作成された雲の画像が6時間ごとに更新されており、これを取得するスクリプトと組み合わせることで、ほぼリアルタイムに現在の雲の画像を表示できる。

●tkxplanetとxplanetbg

もし、コマンドラインでオプションを指定するのが面倒なら、同様の設定をGUIで行うtkxplanetを利用するとよ

XGLOBE & XPLANET MAPS (http://www.radcyberzine.com/xglobe/)	写実的、生物圏、高度別などさまざまな地球のマップ画像を提供（JPEG形式）。サイズはいずれも400×200、800×400、1600×800ドットの3種類。一部画像については、さらに大きなサイズも用意されている。
The Planetary Maps Hub (http://www.lancs.ac.uk/postgrad/thomasc1/render/maps.htm)	地球のほか、火星、木星、土星、天王星、海王星の惑星のマップ画像を提供（JPEG／PNG形式）。画像サイズは惑星ごとに異なる。低地を海で覆うことで作成した「青い金星」や「青い火星」は必見だ。
Planetary Image Maps (http://www.jht.cjb.net/)	水星や冥王星を含む全惑星や月のマップ画像を提供（JPEG形式）。地球の画像サイズは他のWebサイトのものより大きいので、利用する際には注意しよう。

リスト1　マップ画像のあるWebサイト

いだろう。

tkxplanetのウィンドウには設定用のラジオボタンやスライドバーが並んでいる（**画面3**）。たとえば、日本を中心とした表示にするには、[longtitude]を「135」、[latitude]を「35」付近に設定すればいい。

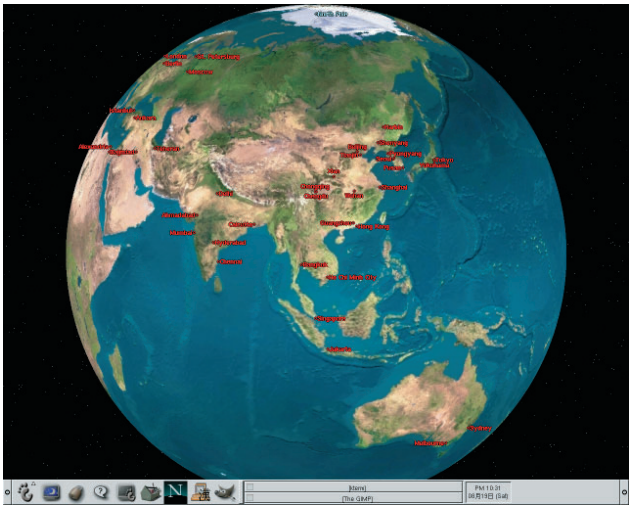
[Run]ボタンを押すと、現在の設定でXPlanetが実行され、しばらくするとルートウィンドウに画像が表示される。なお、端末画面（ktermなど）には、現在の設定に対応するオプションの付いたコマンドラインが表示されるので、これを利用して気に入った設定をシェルスクリプトやエイリアスに登録することも可能だ。

一定時間ごとにルートウィンドウの表示を更新するには、xplanetbgを利用する。表示に関するコマンドラインオプションはXPlanetと同じだ。初期設定では5分ごとにXPlanetを呼び出す。この間隔は－waitオプションで変更できる（値は秒単位）。

●ウィンドウ表示も可能

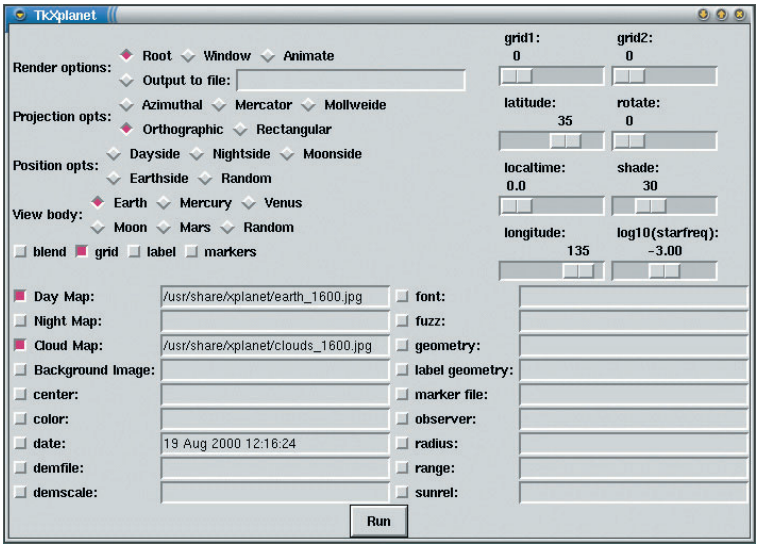
XPlanetを－wや－aオプション付きで起動すると、通常のウィンドウに画像が表示される。特に「アニメーション」を意味する－aオプションを指定した場合は、Mesaライブラリを使った3D表示により、ウィンドウ内の惑星がリアルタイムに自転する（**画面4**）。

アニメーション表示では、カーソルキーで表示の中心を変更したり、Home／Endキーで拡大・縮小したり、＋／－キーで自転速度を変更するなどの操作が可能だ。詳細はhキーで表示されるヘルプを参照されたい。

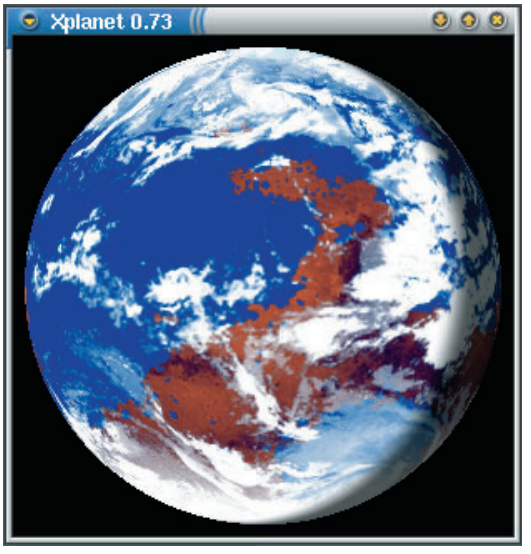


画面1
ルートウィンドウに地球の画像を表示する（正射投影図法）。

画面2
矩形図法による表示の例。雲の画像を重ねることも可能だ。



画面3
オプション設定をGUIで行うtkxplanet。



画面4
「青い火星」を通常のウィンドウにアニメーション表示。

ヘリコプターでレスキュー活動を行う3Dゲーム

WolfPack's Search and Rescue

バージョン：0.3　　ライセンス：GPL

http://wolfpack.twu.net/SearchAndRescue/
http://wolfpack.twu.net/YIFF/ (YIFF)
http://wolfpack.twu.net/libjsw/ (jsw)

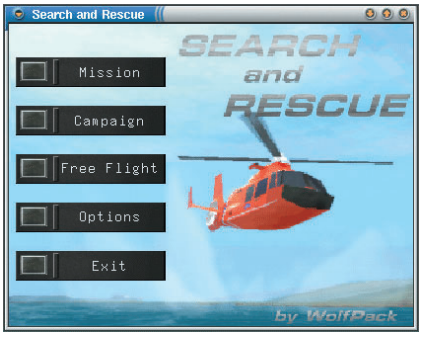
WolfPack's Search and Rescueは、ヘリコプターを操縦して、海に山にのレスキュー活動を行う3Dシミュレーションゲームだ。ヘリの操縦はそれほど難しくないので、アクションゲームのノリでプレイできる。まだ開発中だが、すでに3つのシナリオが用意されているほか、ミッションのないフリーフライトも可能だ。ビルドと実行には3Dライブラリのmesaが必要。3Dアクセラレータ無しでもプレイ可能だ。

●ビルドとインストール

WolfPack's Search and Rescue（以下Search and Rescue）は、ファイル一式をtar＋gzipしたtarボールのみ配布されている。configureスクリプトは付属しないため、sarディレクトリのMakefileを自分の環境に合うように書き換える必要がある。

初期設定では、サウンドドライバYIFFと、ジョイスティックドライバ（jsw）を利用する設定になっている。たとえば、ジョイスティックドライバを利用しない場合は、Makefileの98行目の「-DJS_SUPPORT」と、114行目の「-ljsw \」を削除すればいい。同様に、YIFFを利用しない場合は、98行目の「-DHAVE_Y」と、117行目の「-lY2」を削除する必要がある。また、Mesaなどのインクルードファイルが/usr/X11R6/includeに置かれている場合は、136行目に「-I/usr/X11R6/include」を追加しておこう。

こうした変更を行った後、sarディレクトリで「make」→「make install」とすると、ビルドとインストールが行われる。



画面1
タイトル画面。まずはオプション設定を行おう。

●初期設定とミッション選択

Search and Rescueの実行ファイルは/usr/gamesにインストールされる。/usr/gamesが環境変数PATHに含まれる場合は「SearchAndRescue」、含まれない場合は「/usr/games/SearchAndRescue」として起動しよう。

最初に、メニュー付きのタイトル画面が表示される（**画面1**）。ウィンドウの大きさはユーザーが自由に変更できるが、大きくするとそれだけプレイ画面の更新に時間がかかるようになるので注意されたい。

[Option]以下の画面では、シミュレーションの難易度やジョイスティック操作、グラフィックやサウンドの設定を変更できる。3Dアクセラレータがない場合は、[Graphics]の設定（**画面2**）で、[Object texure]以外の設定をオフにすると処理が軽くなる。

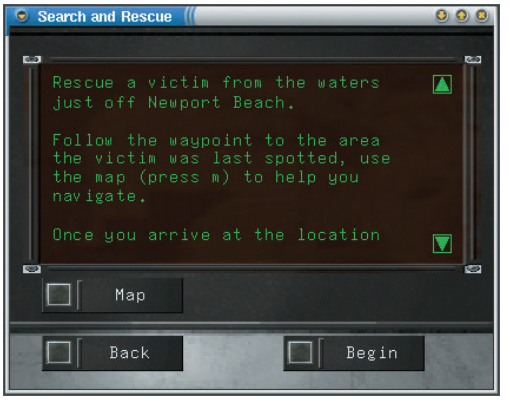
また、サウンド機能を利用する場合は、[Sound]の設定で、[Sound Level]を「All On」に変更しよう。初期設定は「Off」なので、そのままだとヘリのローター音やエンジン音などが出力されない。

●ミッション選択から飛行まで

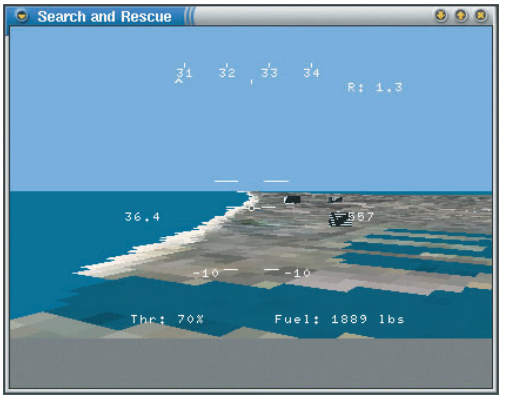
タイトル画面で[Mission]をクリックすると、ミッション選択画面に切り替わる。現時点では、トレーニング用の3種類のミッションのみ用意されている。プレイするミッションを選択し、[Briefing]をクリックして説明を受けよう（**画面3**）。[Begin]をクリックするとミッション開始だ。

ウィンドウには、ヘリのコックピットからの眺めが表示される（**画面4**）。

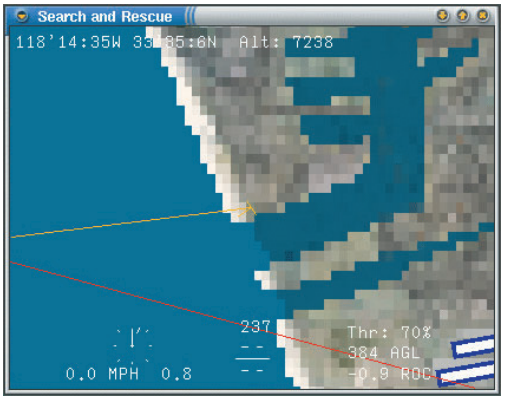
画面2
グラフィックに関する設定はこの画面で行う。



画面3
ミッションの概要を確認。マップの表示も可能だ。



画面4
ヘリのコックピットからの眺めが表示される。視点は切替可能。



画面5
マップ画面で現在位置と救助に向かう方向を確かめる。

ヘリの操作はキーボードで行う（F1キーでヘルプ表示）。海で遭難している人を救助するミッション「Rescue Over Water」を例にとって基本的なキー操作を説明しよう。

まず、PageUpキーでスロットルを開けてエンジン出力（Thr）を上げる。数値が70%以上になるとヘリが浮上しはじめる。なお、ある程度高度（AGL）をとったら、gキーを押してギア（車輪）を格納しよう。

ヘリを前進させるには、↑キーを何回か押してヘリを前傾させればいい。前傾しすぎたら↓キーで戻す。逆にヘリを後ろに傾けてバックさせることも可能だ。左右に向きを変えるには、Ctrl－←／→キーを使う。

なお、一部のウィンドウマネージャ（sawfishなど）は、Ctrl－カーソルキーをページ切り替えなどに利用しているので、このゲームをプレイする前に、あらかじめウィンドウマネージャ側のCtrl－←／→キーの設定を無効にしておく必要がある。

●救助用バスケットで遭難者を救出

mキーを押してマップ表示に切り替えると、ヘリの位置と向きが逆T字で表示される（**画面5**）。また、出発地から目的地に赤い線、ヘリから目的地に黄色い線が伸びているので、赤い線に沿ってヘリを目的地に向かわせよう。なお、マップは＋／－キーで拡大／縮小が可能だ。F2キーでもとのコックピット視点に戻る。

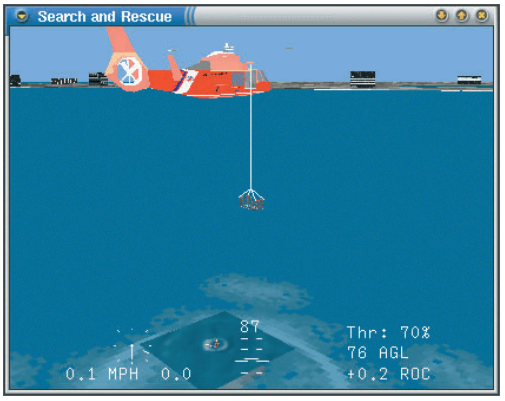
目的地に近づいたら、↑／↓キーを使ってヘリを水平に戻し、PageDownキーでスロットルを閉じて降下する。最終的には、スロットルを70%に保って、高度100フィート以下でホバリングしている状態にしよう。

続いて、F5キーを押して救助用バスケット視点に切り替える。Ctrl－←／→キーで向きを変えつつ、海面を泳ぎながら救助を待つ遭難者を探そう。最初は、－キーで画面をズームアウトすると見つけやすい。

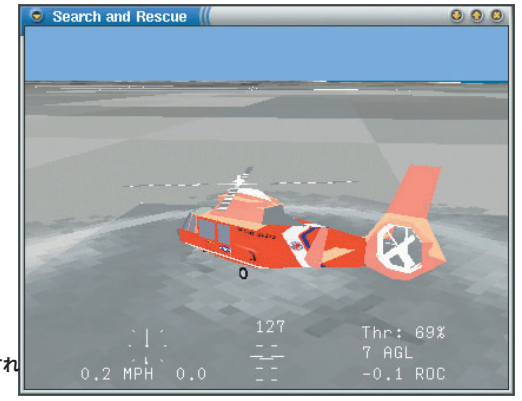
遭難者を確認したら、そばまでゆっくり移動し、Ctrl－“－”（マイナス）キーを押し続けて救助用バスケットを海面まで下ろす（**画面6**）。遭難者のすぐそばにバスケットを移動させるように、＋キーで画面をズームインしつつヘリの位置を微調整しよう。

遭難者がバスケットに乗ったら、Ctrl－＋（プラス）キーでバスケットを巻き上げてヘリに収容すれば救助完了だ。再びスロットルを開いて高度をとり、ヘリポートを目指そう。

ヘリポートへの着陸は、F3キーのスポット視点やF5キーの救助用バスケット視点で行うといい（**画面7**）。着陸前にgキーでギアを出すことを忘れずに。無事にヘリポートに着陸できればミッションクリアだ。



画面6
救助用バスケットを遭難者のすぐそばまで下ろそう。



画面7
救助後にヘリポートに着陸すればミッションクリアだ。

CDのカバーを作成・印刷する

KCDLabel

バージョン：0.3　　ライセンス：GPL

http://www.linuxsupportline.com/~pep/kcdlabel/kcdlabel.html

KCDLabelは、CDケースサイズのカバーやブックレットなどの「ラベル」を作成するソフトだ。画面上で実際のイメージを確認しながら手軽に作成できる。単にテキストを並べるだけでなく、CD-ROMやCD-Rのファイル一覧を取り込んでツリー表示したり、凹凸や円形に変形させたテキストや画像を挿入することも可能だ。作成したラベルはKCDLabelから直接印刷できる。動作にはQt／KDEが必要だ。

●ビルドから起動まで

KCDLabelは、tarボールとRPMパッケージの両方で配布されている。ただし、RPMパッケージの導入にはいくつか問題があるので、tarボールを利用したほうがいい。「./configure」→「make」→「make install」という一般的な手順だ。

起動するとラベルが新規作成され、ケースの裏側に入れる「バックページ」が表示される（**画面1**）。ページの左右に「スライド」（背表紙）が設けられており、ページ番号が0なので他のページと区別できる。

↓キーを押して1ページ目に切り替えると、CDケースの表側に入れる「フロントページ」の1ページ目が表示され、さらに↓キーを押すと2ページ目以降に順次切り替わる。これらはブックレット（歌詞カードなど）に相当するもので、スライドは含まれない。

ページの切り替えは↑／↓キーやPageUp／Downキーで行うほか、[Page]－[Go to page number]で直接指定することも可能だ。

●バックページを作成

ラベルの各ページには、1行・複数行・変形テキスト、JPEG／PNG／XPM形式などの画像、ファイル一覧のツリー表示（ボディ）を組み合わせて配置できる。フロントページの1ページ目とバックページを表・裏の表紙とし、フロントページの2ページ目以降にボディを配置するのが一般的だ。

ここでは、バックページに1行テキストと画像を配置してみよう（**画面2**）。まずは、[String]－[Insert single line]を選択し、ダイアログで文字列（日本語可）や色などを設定する（**画面3**）。画像についても同様に、[Image]－[Insert]で配置できる。

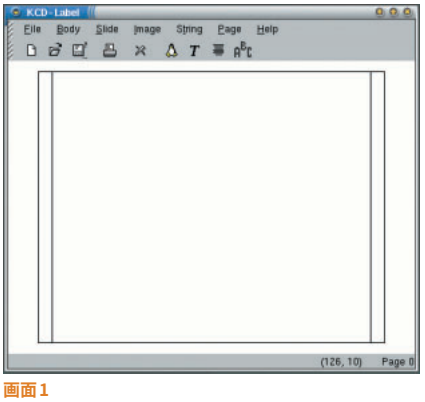
配置後のテキストや画像は、ダブルクリックで選択し、ドラッグで位置を変えられる。また、テキストでは設定内容の変更、画像では拡大・縮小、タイル表示、背景の透明化といった操作が可能だ。

なお、バックページの左右端にあるスライドには、テキストのみ配置できる。設定は、[Slide]以下のメニューで独立して行う。

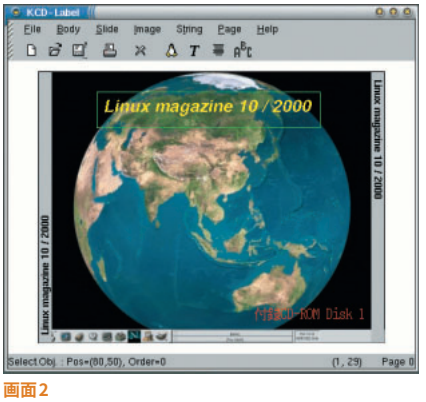
●フロントページを作成

続いて、フロントページの1ページ目に複数行テキストと変形テキストを配置してみよう（**画面4**）。

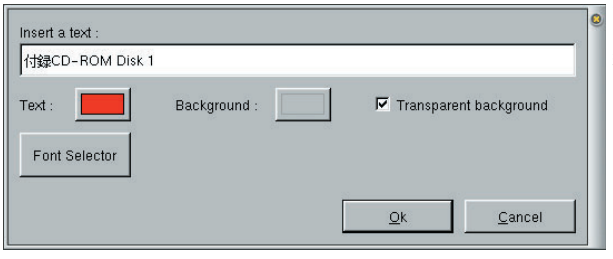
複数行テキストは、[String]－[Insert multi line text]を選択し、ダ



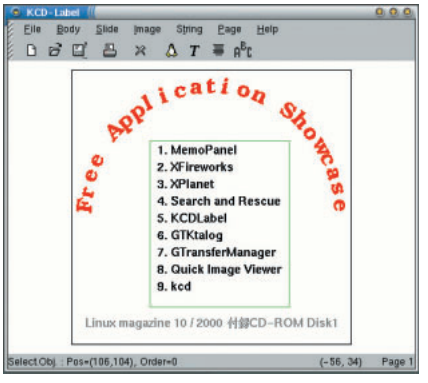
画面1
起動時にはケースの裏側に入れるバックページが表示される。



画面2
バックページに裏表紙となる1行テキストと画像を配置する。



画面3
1行テキストの文字列には日本語を利用可能だ。



画面4
表紙となるフロントページの1ページ目を作成。

イアログで複数行の文字列（日本語可）を設定する。左右寄せやセンタリング、行間スペースの設定も可能だ。

一方、変形テキストは、[String]－[Insert curved single text]を選択し、ダイアログで文字列や変形の種類（凸形・凹形・円形）などを設定する。なお、プログラム内で文字列を1バイトずつ表示しているため、日本語は表示できないので注意されたい。

●ボディを挿入する

フロントページの2ページ目以降には、CD-ROMのファイル一覧をツリー表示した「ボディ」を配置しよう。まずは、対象となるCD-ROMをCD-ROMドライブに入れ、

```
$ mount -t iso9660 /dev/cdrom
/mnt/cdrom
```

などとしてマウントし、アクセス可能にしておく。

続いて、[Body]－[Read content]でファイル一覧情報を取り込む。初期設定のままだとボディは表示されないので、[Body]－[Number of columns]を[1]に変更しよう。これで、フロントページの2ページ目以降に、CD-ROMのファイル一覧がツリー表示される（**画面5**）。上記の設定を[2]以上に変更することで、多段表示も可能だ。

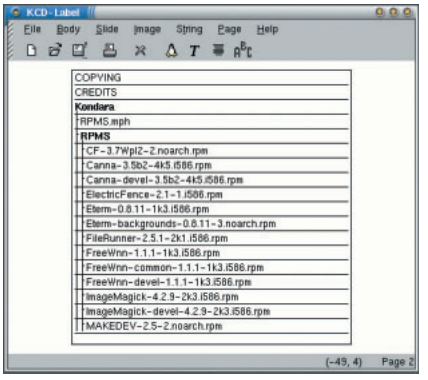
ところで、初期設定では全ファイルがツリー表示されるため、CD-ROMの内容によってはフロントページのページ数が膨大になる。この場合は、[Body]－[Numer of directory levels]を[All]以外の値に設定して、表示するディレクトリの深さを制限しよう。

このほか、水平線やツリー表示の罫線を消したり、右下にページ番号を表示することも可能だ。さらに、テキストや画像をボディに重ね合わせることもできる。

なお、[Body]－[Body layout]で開くダイアログでは、ボディの表示を開始するページを変更できる。フロントページの1ページ目や、バックページに表示する設定も可能だ。

●完成したラベルを印刷

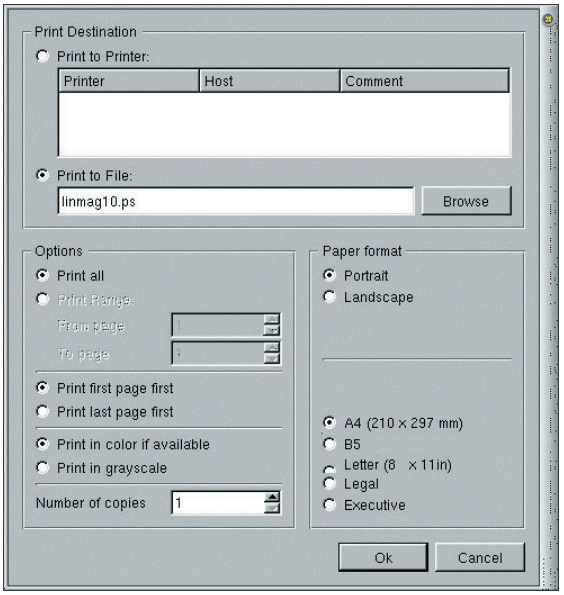
完成したラベルは、[File]－[Save]で独自形式（拡張子.kcd）で保存するほか、KCDLabelから直接プリンタで印刷したり、PostScript形式のファイルとして出力できる。



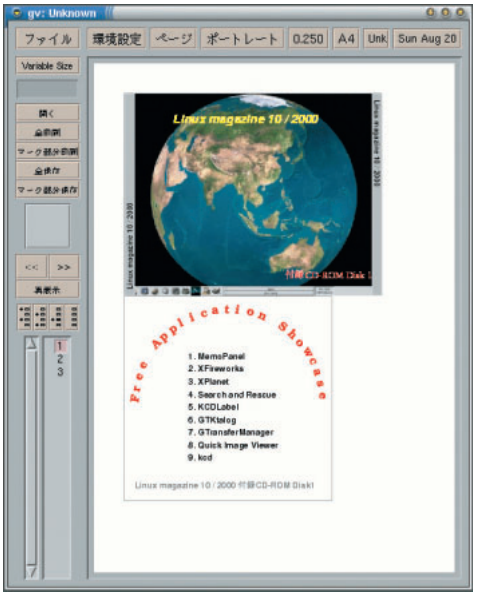
画面5
ファイルツリーを表示する「ボディ」を配置。多段表示も可能だ。

[File]－[Print]を選択すると、印刷ダイアログが開く（**画面6**）。直接プリンタで印刷する場合は、[Print to Printer]をチェックして、その下のリストからプリンタを選択する。

一方、いったんファイルに出力する場合は、[Print to File]をチェックしてファイル名（拡張子は.psか.eps）を設定すればいい。出力したファイルをgvでプレビューすることも可能だ（**画面7**）。なお、印刷の際には、1枚の紙に2ページ分のラベルがまとめて出力される。



画面6
印刷ダイアログで直接印刷かファイルへの出力を選択する。



画面7
PostScript形式の出力ファイルをgvでプレビュー。

CD-ROMのカタログを作成する

GTKtalog

バージョン：0.1.4　　ライセンス：GPL

http://gtktalog.sourceforge.net/

GTKtalogは、CD-ROMのファイルとディレクトリ情報をデータベース化したディスクカタログを作成するソフトだ。複数のCD-ROMのファイル情報を1つのカタログでまとめて扱える。作成後は、実際にCD-ROMをマウントすることなく、高速にファイルを検索可能だ。ファイル名による検索だけでなく、日付やサイズなどの条件を組み合わせた検索にも対応している。動作にはGTK＋が必要だ。

●ビルドから起動まで

GTKtalog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。ビルドとインストールは「./configure」→「make」→「make install」という一般的な手順だ。

「gtkatlog&」として起動すると、左右にペイン（領域）が分割されたウィンドウが開く。左のペインにはCD-ROMごとのディレクトリツリー、右のペインには選択したディレクトリ内のファイル一覧が表示される（**画面1**）。

初めて起動した際には、ツールバーの[新規]ボタンを押してディスクカタログを新規作成しよう。次回からは、保存したカタログファイルを[Open]ボタンで読み込めばいい。

●CD-ROMのファイル情報を追加

ディスクカタログにCD-ROMのファイル情報を追加するには、2つの方法が用意されている。

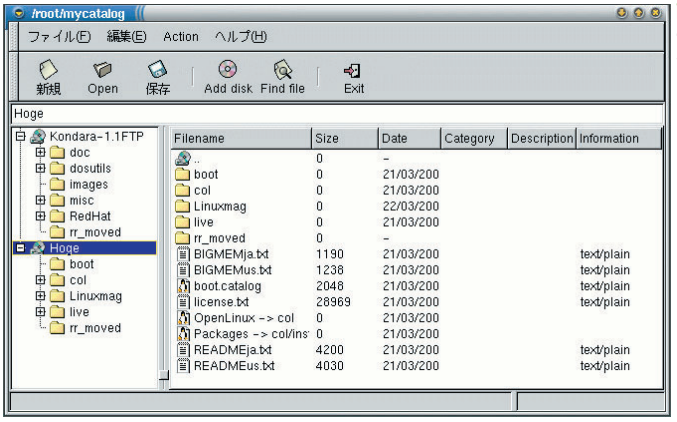
CD-ROMをマウント済みの場合は、ツールバーの[Add Disk]ボタンを押そう。ダイアログ（**画面2**）が開いたら、ラベル（ディスクの名称）とCD-ROMドライブのマウントポイント（「/mnt/cdrom」など）を設定する。[了解]ボタンを押すと、マウントポイント以下のファイルがカタログに追加される。

一方、複数のCD-ROMを効率よくカタログ化したい場合は、[Action]－[Fast Add]を選択する。こちらは、ラベルを入力するだけで、自動的にCD-ROMのマウントを行い、マウントポイント以下のファイルがカタログ化される。また、処理後はCD-ROMドライブが自動的にイジェクトされる。

●カタログからファイルを検索

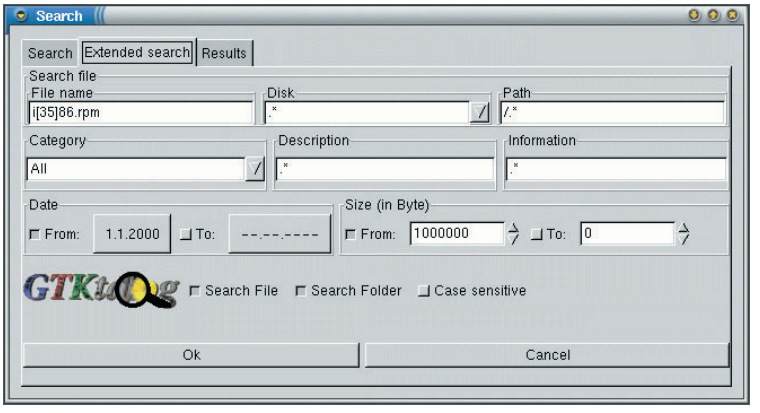
カタログのファイルやディレクトリに対して検索を行うには、ツールバーの[Find]ボタンを押す。検索ダイアログでは、ファイル名のみの検索のほか、日付やファイルサイズ、ディスクのラベルなどの条件を組み合わせた拡張検索も可能だ（**画面3**）。

検索結果はダイアログの[Results]ページに一覧表示され、ファイルをダブルクリックすることで、ウィンドウの表示がそのファイルのあるディレクトリに切り替わる。



画面1
ディレクトリツリーとファイル一覧が表示される。

画面2
ディスクを判別するラベルと、マウントポイントを指定しよう。



画面3
複数の条件を組み合わせた検索が可能だ。検索速度も高速。

複数ファイルを効率よくダウンロード

GTransferManager(GTM)

バージョン：0.4.5　　ライセンス：GPL

http://gtm.sourceforge.net/

GTransferManager（GTM）は、HTTP／FTPを利用してファイルをダウンロードするwgetのフロントエンドだ。Netscapeなどから URLをドラッグ＆ドロップすることでファイルを登録でき、複数のファイルを効率よくダウンロードできる。拡張子に応じて異なるディレクトリに振り分けることも可能だ。GNOMEのパネルに常駐する小型のアプレットも用意されている。動作にはGTK＋／GNOMEとwgetが必要だ。

●ビルドとインストール

GTMは、tarボールのほか、各種のパッケージでも配布されている。Red Hat系ディストリビューションではRPMバイナリ（あるいはソース）パッケージを利用するといいだろう。tarボールでは、「./configure」→「make」→「make install」という手順でビルドとインストールが可能だ。

また、筆者が作成した日本語カタログ（ja.po、ja.gmo）を別途収録した。バイナリパッケージを利用している場合は、バイナリ形式のja.gmoを、「cp ja.gmo/usr/share/locale/ja/LC_MESSAGES/gtm.mo」とリネームコピーする。tarボールの場合は、GTM展開先のpoディレクトリにja.poをコピーし、configureの2544行目、「ALL_LINGUAS="…"」に「ja」を追加した後、ビルドを行えばいい。

●ドラッグ＆ドロップで登録可能

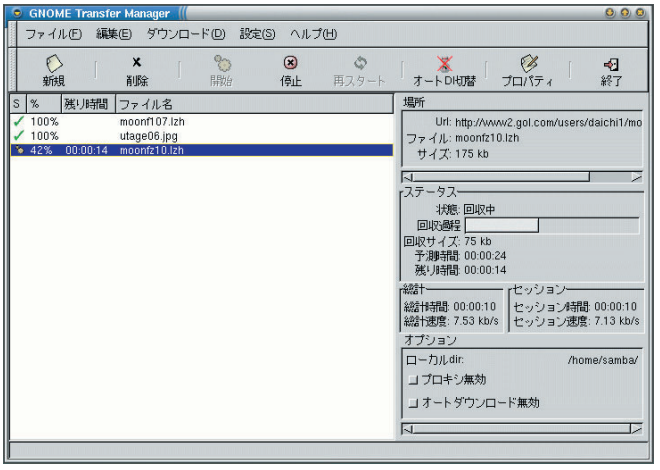
ktermなどから「gtm&」として起動すると、左側にダウンロードするファイルの一覧、右側にダウンロードに関するさまざまな情報が表示されたウィンドウが開く（**画面1**）。

まずは、[設定]－[設定]でダイアログを開いて、デフォルトのダウンロード先ディレクトリや使用するインターフェイス（ppp0／eth0）、同時にダウンロードするファイル数などを設定しておこう。

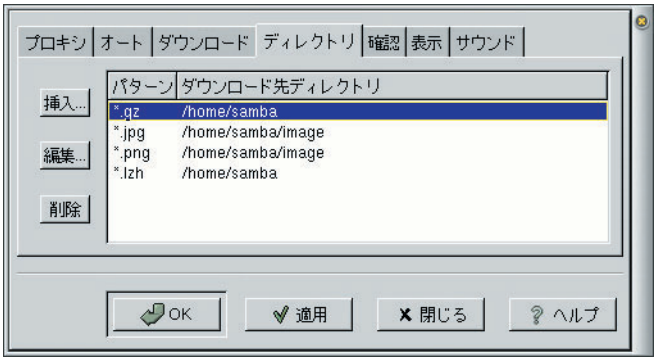
またファイルの拡張子に応じてダウンロード先ディレクトリを切り替える設定も可能だ（**画面2**）。

ファイルの登録は、URLをキーボードから入力することでも行えるが、NetscapeなどからURLのリンクをGTMにドラッグ＆ドロップするほうが簡単だ。ドロップしたファイルは左側のリストに追加され、クリックすると右側のパネルにURLなどの情報が表示される。このほか、GNOMEパネルに常駐してドロップを受け付けるアプレット（gtm_applet）も利用できる（**画面3**）。

初期設定では、ファイルの登録と同時にダウンロードが開始され、ウィンドウ右側のペインに進行状況や残り時間などが表示される。一時停止後ダウンロードを再開（レジューム）したり、ファイルの先頭からダウンロードし直すことも可能だ。



画面1
左にファイル一覧、右にダウンロード情報が表示される。



画面2
拡張子ごとに異なるダウンロード先を設定できる。

画面3
GNOMEのパネルに常駐し、URLのドロップを受け付けるGTMアプレット。

小型で軽量な画像ビューア

Quick Image Viewer(qiv)

バージョン：1.4　　ライセンス：GPL

http://www.klografx.de/software/qiv.shtml

Quick Image Viewer（qiv）は、imlibを利用したシンプルな画像ビューアだ。JPEG／GIF／TIFF／PNGなど主要な画像形式に対応している。サムネイル（縮小イメージ）表示などの機能を持たない代わりに、動作は高速だ。マウスやキー操作により、画像の切り替えや拡大・縮小、フルスクリーン表示、ルートウィンドウ（背景）への表示、スライドショーなどを行える。

●ビルドから起動まで

qiv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。付属のconfigureスクリプトはダミーなので、「make」→「make install」とするだけでビルドとインストールが行われる。

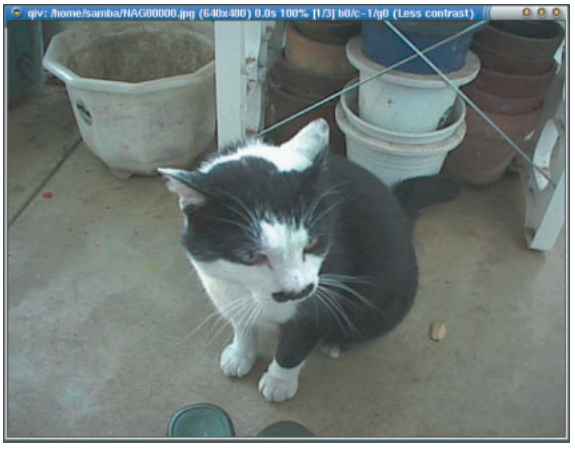
使い方は簡単で、「qiv 画像ファイル名」とすればいい。複数の画像を指定することもできる。ウィンドウには最初の画像が表示され、タイトルバーにファイル名や画像サイズ、何枚目の画像かといった情報が表示される（**画面1**）。

qivにはさまざまなコマンドラインオプションが用意されているが、ほとんどのオプションは実行中のキー操作で代替できるので、ここではディレクトリを再帰的にたどって画像を表示する－uオプションのみ紹介する。

－uオプションに続けて、ファイル名の代わりにディレクトリを指定するだけでいい。たとえば、

```
$ qiv -u ~/image
```

とすると、ホームディレクトリ以下のimageディレクトリや、さらにそのサブディレクトリに含まれる画像が表示されるわけだ。



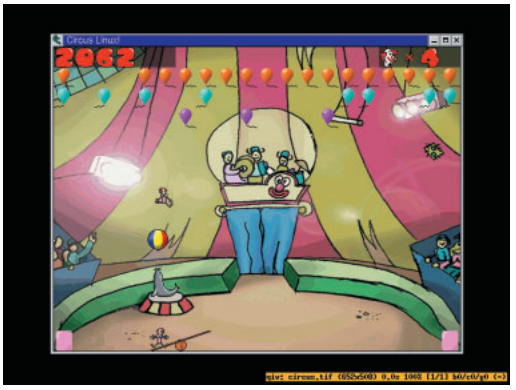
画面1
シンプルなウィンドウ。情報はタイトルバーに表示される。

●マウスとキーで操作する

画像が表示された後は、マウスの左右ボタンやSPACE／Back Spaceキーで次の画像や前の画像に切り替えられる。ウィンドウ表示のほか、fキーでフルスクリーン表示に切り替えることも可能だ（**画面2**）。

フルスクリーン表示の状態では、ファイル名などの情報が右下に表示され、カーソルキーで画像を移動できる。ウィンドウ表示に戻るには再度fキーを押せばいい。

sキーを押すと、一定時間ごとに自動的に次の画像に切り替わるスライドショーが始まる。時間間隔（初期値は3秒）は－dオプションで変更可能だ。このほか、x／y／zキーで画像をルートウィンドウに貼り付けたり（**画面3**）、＋／－キーで画像を拡大・縮小することもできる。その他のキー操作については、？キーで表示されるキー操作一覧を参照されたい。



画面2
fキーを押してフルスクリーン表示に切り替える。

画面3
画像をルートウィンドウ（背景）に貼り付けることも可能だ。

ディレクトリツリーから移動先を選択

kcd

バージョン：5.0.0　　**ライセンス：GPL**

http://kcd.sourceforge.net/

kcdは、ディレクトリツリーからの選択でカレントディレクトリの切り替えを行うソフトだ。コンソールやktermなどの端末画面でも、GUIを使ったファイルマネージャと同じ感覚でカレントディレクトリを変更できる。ディレクトリ情報はファイルに保存されているため、ツリーの表示は高速だ。Emacsライクなインクリメンタルサーチ機能も用意されている。動作にはncursesとzlibライブラリが必要だ。

●ビルドとインストール

kcdは、tarボールのほかRPMパッケージも配布されている。Red Hat系ディストリビューションではRPMパッケージを使うのが簡単だろう。tarボールからのビルドとインストールも、「./configure」→「make」→「make install」という一般的な手順だ。

インストールすると、/etc/profile.dディレクトリにkcd.shとkcd.cshが置かれる。これらは、ログイン時に実行されるシェルスクリプトで、「kcd」というエイリアスを定義する。なお、X上のktermなど、ログインシェル以外でもkcdを使いたい場合は、.bashrcに「source /etc/profile.d/kcd.sh」という行を追加しておこう。

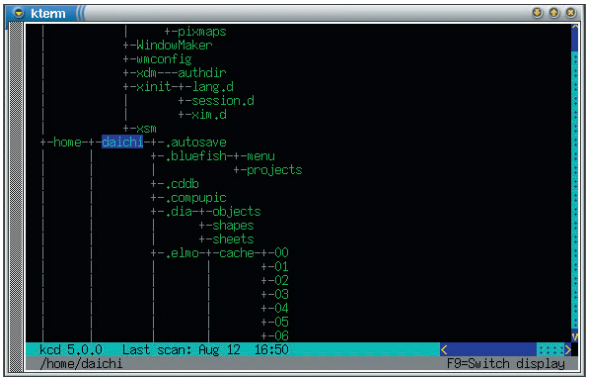
このほか、ディレクトリ情報のスキャンの対象外にしたいディレクトリがあれば、/etc/kcd.conf（あるいは~/.kcd.conf）に「SkipDir="ディレクトリ"」という形式で記述する。CD-ROMやWindowsファイルシステムのマウントポイントを登録しておくといいだろう。

●ツリー表示から移動先を選択

コマンドラインで「kcd」とすると、ディレクトリツリーが画面に表示される（**画面1**）。なお、そのユーザーが初めてkcdを起動した場合、最初にディレクトリ情報のスキャンとファイル（~/.kcd.save.gz）への保存が行われる（**画面2**）。再度スキャンしたい場合は、−rオプションを付けてkcdを実行すればいい。

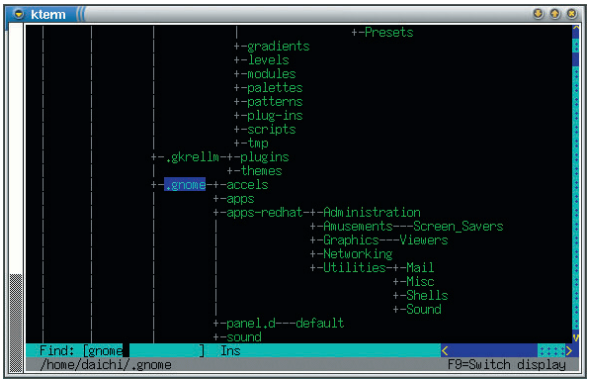
現在注目しているディレクトリは、画面で青く反転表示した「カーソル」で示される。このカーソルをカーソルキーなどで移動し、Enterキーかスペースキーを押すと、カレントディレクトリがカーソル位置のディレクトリに変更される。変更しないまま終了したい場合はCtrl−Cキーを押そう。

また、英数字キーを押すと、自動的に「findモード」に切り替わり、入力した文字列を含むディレクトリが検索される（**画面3**）。Emacs風のインクリメンタルサーチにより、必要最小限のキー入力で目的のディレクトリを見つけられる。検索結果が複数のディレクトリにマッチしている場合は、↑／↓キーで前後の候補に移動できる。findモードから抜けるには、Tabキーを押せばよい。



画面1
ディレクトリツリーが端末画面に表示される。

画面2
初めて実行した際にディレクトリ情報がスキャンされる。



画面3
Emacs風のインクリメンタルサーチでディレクトリを検索中。

隠喩としてのコンピュータ

自由なコンピュータと不自由な音楽

文：豊福 剛
Text : Tsuyoshi Toyofuku

IRCのあるチャンネル（＃初心者）でスペインの人とプライベートトークをした。そのスペイン人は日本語を勉強している学生だったのだが、てっきり留学生かと思ったら、スペインに住んでいる人だった。その彼は、日本のアニメやJ-POP趣味が高じて日本語を勉強し始めたらしく、浜崎あゆみ、Every Little Thing、Janne Da Arc、L'Arc～en～Cielといったあたりが大好きだという。

J-POPのCDはスペインでも売っているらしい。とはいえ、現地では通常の倍以上の価格になるらしく、彼のポケットマネーを圧迫している。というわけで、MP3ファイルで少なからず入手してもいるらしい。プロモーションビデオもMPEGで見ているようだ。どこかのサイトに海賊ファイルのアーカイブがあって、そこからダウンロードしているのか、それとも、IRCで知り合った人から直接ファイル転送してもらっているのか、そのあたりの詳しい事情までは聞かなかったのでわからない。Webサイトなどで不特定多数に向けて海賊ファイルを公開するのは音楽著作権の侵害になるのだろうが、友達から直接ファイルを転送してもらうのは、友達からCDやビデオを借りることと同じように思えた。

CDやビデオカセットは物であるので、一度に一人の友達にしか貸せない。それをカセットにダビングしてあげたとしても、ダビングには元のCDやビデオを再生するのと同じ時間がかかるので、ある時間内にダビングできる数は限られていた。そのような貸し借りをする友達というのは、せいぜい5人前後といったところなのではなかっただろうか。しかしファイルにしてしまえば、そのような制限はなくなる。それでも、ファイルの転送をする相手は5人前後になるのか。5人前後の友達に貸す行為が著作物の個人的使用に属するとしたら、500人の友達に貸す行為が個人的使用に属さない根拠はどこにあるのだろうか。それとも、5人前後の友達に貸す行為すら、非合法ということになってしまうのだろうか。

レンタル業の興隆とともに変ったこと

'80年代の前半、大学生だったとき、三鷹市に住んでいたのだが、三鷹駅の南口にあるパチンコ屋の近くにレコードのレンタル屋がオープンした。たしか中央大学の学生が始めたベンチャーだったように思う。当時はCDが普及する以前の時代で、レンタル屋の利用者のほとんどは、借りてきたレコードをカセットに録音していたはずだ。それで、類似のレンタル屋がいくつも開店するようになり、レコード業界から音楽著作権の侵害だとして訴訟され、社会問題

として新聞などで報じられた記憶がある。また、レコードのレンタル屋と前後して、ビデオのレンタル屋も開店し始めた。ビデオデッキそのものは'70年代の終わり頃には、かなり一般家庭に普及し始めていたのだが、最初の普及段階では、テレビ番組を録画するのに使われていた。'80年代になってから、映画やAVのビデオが商品として売られるようになったが、1本1万円くらいする高価な商品だった。

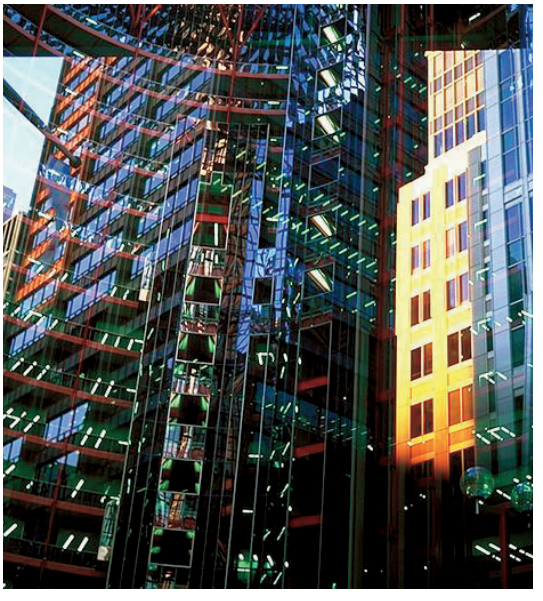
'80年代の後半になると、アナログ盤からCDへの移行が全面化するが、それと並行して、レコードのレンタル屋にあまり新譜が入ってこないようになり、店をたたむところが増えてきた。一方ビデオのレンタル屋のほうは、それ以前のちょっと薄暗い印象のする店ではなく、映画の品揃えが豊富で、店内も明るい雰囲気の店が増えてきた。また、ビデオのレンタル屋でCDも借りられるようになった。

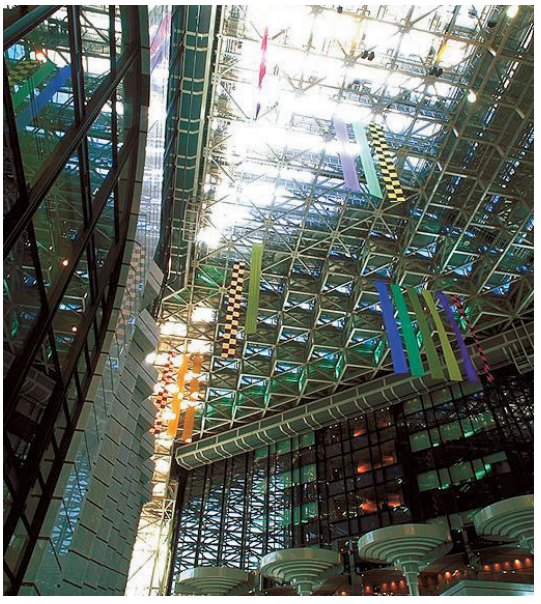
おそらく'80年代の終わりから'90年代の前半にかけて、レンタル業とレコード／ビデオ業の間で、著作物の使用料をめぐるルールが確立されたのだろう。もともと家庭用ビデオデッキが登場したとき、ハリウッドは映画ビジネスの収益を脅かすことを危惧し、ビデオデッキのメーカーに圧力をかけたこともあったらしい。もしハリウッドがビデオレンタルをつぶしていたら、結果的に自らの首を絞めることになっていたはずだ。ただし皮肉なことに、LPがCDになり、CDの値段も安くなったのと並行して、友達とCDの貸し借りをするようなこともなくなってしまったように思う。

プロモーション・ビデオの閉塞

アナログ盤からCDへの移行が進行した'80年代は、MTVがある意味で音楽のスタイルを変えた時代でもある。マイケル・ジャクソンの『スリラー』のプロモーションビデオが登場したあたりから、音楽に映像を重ね合せた表現の文法のようなものが、確立したように思う。それまでのプロモーション・ビデオには、どちらかといえば表現のスタイルを手探りで実験しているような印象があり、どこか素人くささの残る、ほのぼのした遊び心が感じられたのだが、おそらく『スリラー』以降は、最新のCGやビデオ編集技術を駆使するのが当たり前といった感じになり、映像表現があまりに過剰になってしまったように思う。音楽が映像にジャックされたのだ。

とはいえ、ポップ・ミュージックは音楽だけで成立しているわけではない。ブライアン・イーノは次のように書いている。「ポップには常に最低でも以下のようなものが混じり合っていた―メロディ、サウンド、言葉、服装、ファ





ッション、ライフスタイル、年代に対する考え方、権威、人間関係、肉体と性、ダンス、視覚的イメージ、こういったものの価値を評価しなおす作業。＜中略＞そしてもちろん、人々が『音楽』にエキサイトしてレコードを買うとき実際に買っているのは、この大きな混合物全体なのだ。買う側にとってはもちろん、伝統的な『音楽的』要素が少なくとも受容でき、できれば素晴らしいものであるほうがいいわけだが、みんなそれだけを買っているのではない。カルチャー・ライフの哲学、スタイル、フィーリングを買い込んでいるのである」(『A YEAR』パルコ出版)。

しかし、プロモーション・ビデオは基本的には商品ではない。もちろん、プロモーション・ビデオを集めたビデオが商品として発売されることはあるが、そのようなビデオを買う人は、けして多くはない。やはりプロモーション・ビデオは、CDを売るための販促ツールなのであり、広告なのだろう。そうである以上、それは繰り返し何度も放送されることで、見るものを魅惑し、CD購入の契機として作用すべき性質のものでなければならない。プロモーション・ビデオに感じる過剰さは、だから広告表現が過剰であるということなのだ。

音楽再生装置としてのコンピュータ

CDの売上げが音楽ビジネスの根幹を支えている以上、MP3で勝手に楽曲の海賊コピーがやり取りされるのを、業界が黙認するわけはない。ただし、MP3が支持されている背景には、音楽再生装置としてのコンピュータの利用法を、音楽を売る側ではなく、買う側のユーザーが発見したことがある。MP3ファイルでは、3分の曲はおよそ3Mバイトになる。10曲で30Mバイト、100曲で300Mバイトになるわけだが、「10GバイトのIDEディスクが2万円で買える」ハードディスクを「長尺のオーディオテープのようなもの」(四本淑三『MP3でサウンド天国』UNIX MAGAZINE 1999年7月号)と捉える発想は、違和感なく受け入れられる。

もちろんオーディオテープと違って楽曲にはランダムにアクセスできる。74分のMDには3分の曲が24曲録音できるだけだが、ハードディスクに750Mバイトの空き容量があればMP3ファイルで240曲保存できる計算になる。そうなると、CDやMDを入れ替える手間がなくなるだけでない。WinampなどのMP3再生ソフトを使えば、選曲して曲順を決めて再生するといった操作が、格段にやりやすくなるのだ。またイコライザの設定もいくつかプリセットで用意されているし、細かくカスタマイズすることもでき

る。そのうち、クラブのDJがやっているような感じで曲間をつなぐことができるプログラムが開発されるかもしれないし、すでに存在しているのかもしれない。いずれにせよ、CDやMDをただ再生するというだけでなく、ある種の編集を加えたいという欲求が音楽のリスナーにあり、MP3やその再生ソフトはそうした欲求にある程度応えているのだと思う。

リスナーが音楽の文脈を編集するとき

CDやLPをその曲順通りにカセットやMDに録音するだけでなく、いくつものCDやLPから特定の曲だけをピックアップして、カセットやMDにコンピレーションしているリスナーはきっと多いだろう。もちろん、ビートルズの『アビーロード』やピンク・フロイドの『狂気』のように、全体が1つの組曲として編成されているアルバムは少なくない。かつてはコンセプト・アルバムという言い方もあった。1つの曲の中に、イントロ、サビ、クライマックス、エンディングがあるように、アルバム全体にもそれに類似した構造があった。そうした構造は、ある種の物語性を表現するための文法だったといえるだろう。

しかし、アルバムの曲順は唯一絶対的なものではない。ピンク・フロイドの『狂気』はコンサートでもアルバム通りの曲順で演奏されていたとはいえ、ほとんどのコンサートではアルバムとは別の曲順で演奏されるのが普通だし、アーティストがどの曲をどのような曲順で演奏するかを楽しめる。コンサートにはコンサートの文法があるのだろう。そして、アルバムの中ではあまり目立たなかった曲が、コンサートという独自の文脈の中で演奏されるとき、いままで意識しなかった良さを発見することもある。曲は、その前後の曲次第で、微妙に印象が変わるものなのだ。

そうした性質が曲にはある。であれば、リスナーが自分なりのプレイリストでコンピレーションする行為は、曲を素材にしてある物語を紡ぎ出すための編集行為といえるだろう。ここまでは、多少の不便さはあったとはいえカセットやMDでもできたことである。

DJがまだディスク・ジョッキーと呼ばれていた時代は、曲の紹介がかぶさることはあっても、曲そのものは、原型のままかけていた。今日のDJは、単なる皿まわしではなく、原曲に独自のリズムトラックをミックスしたりもする。もちろん楽器の演奏に上手・下手があるように、DJにも上手・下手はある。それでも、高価な機材なしでミックスやサンプリングができるソフトがあったら、DJ的な曲の編集を楽しみたいというリスナーは、かなりいそうな気がする。

Profile

とよふく つよし

1962年生まれ。メディアデザイン研究所技術顧問。訳書に『Javaプログラムクイックリファレンス』『Java分散コンピューティング』（オライリージャパン）『GIMPパーフェクトガイド』（エムディーエヌコーポレーション）などがある。

盗聴装置もオープンソースだ

文：安田幸弘
Text : Yukihiro Yasuda

昨年、国会を通過した通信傍受法、またの名を盗聴法ってやつが、8月15日とうとう施行されることになってしまった。これは警察に電話やファクス、電子メールといった通信を盗聴することを認めてやろうという法律なのだけど、これは相当アブナイ法律だったりする。

「ボクは盗聴されるような悪いコトしてないもんネ」とか「盗聴の令状は、凶悪犯だけに発行されるんでしょ？ ボクはパンピーだから関係ないし」なんて言ってる人が多んだけど、それが違うんだなあ。悪いコトをしててもしてなくても、凶悪犯だろうがパンピーだろうが、警察がその気になればいくらでも盗聴できちゃうようになっている。オマケに最近の警察といえば不祥事続き、とても信頼できるような状態じゃない。仮に盗聴法がちゃんとした法律だったとしても、日本の警察なんかに盗聴の権限を与えたら何をやってくれるかわかったもんじゃない。

そんなわけで「こんな法律、rm -fしちまえ」と言われてるわけだ。

海の向こうじゃ食人鬼……

日本の警察はあれこれ理由をつけてるようだけど、実はこの盗聴法、どうやら警察が言ってるような「凶悪犯罪云々」という以前に、アメリカのFBIから「おい、日本でも盗聴やれよなー」と言われて、ハイハイと作ってしまった法律らしい。当局の役人はこれを「国際的な要請」なんて言ってるけど、まあ、日本の警察がFBIのパシリをやらされてるなんて、言えないでしょうねえ。

で、そのFBIなんだけど、最近、FBIが作り出した「食人鬼」が世の中を騒がせている。食人鬼といっても、ホラー映画の話ではない。Carnivore（カーニボー）、つまり食人鬼という愛称を持つ盗聴装置の話だ。

Carnivoreは、FBIによって開発された最新鋭の（？）電子メール盗聴装置で、電子メールを片っ端からバリバリ食ってしまうというところからこの愛称が付けられたという。ちなみに、命名者はレッキとしたFBIの係官らしい。日本の警察も、盗聴装置に「山婆一号」ぐらいの名前をつけてマスコミに公開すりゃいいのに。

ま、それはともかく、である。

Carnivoreが開発されたのは、ややこしい電子メールの盗聴を一気に片付けてしまおうという理由なのだが、ごぞんじのとおり、狙った電子メールを確実に盗聴するのは、結構難しい。

SMTPという電子メールのプロトコルは、DNSのMXレコードの情報を見ながらあちこちに転送してくれる。そしてやっと目的のホストに到着したと思ったら、こんどは/etc/aliasesだの.forwardだので転送しまくりされまくりの世界で、電子メールを追っかけるのは実にやっかいなのだ。

しかし、その反面、技術的には電子メールの盗聴はそんなに難しくないという面もある。つまり、MXが向いているホストに網を張って、飛んできたSMTPのパケットを一網打尽に捕捉して、関係がありそうなものだけフィルタリングすればいい。実際、ちょっとしたネットワークの管理者なら、tcpdumpを使って、ワケのわからないパケットを捕まえて犯人（？）を見つけるといった経験をお持ちの方は少なくないだろう。そして実際、あちこち飛びまわる電子メールを確実に捕捉するには、この方法しかない。

しかし、tcpdumpを使ったことがある人なら、このようなツールで目的のパケットを捕まえるには、まずありとあらゆるパケットを捕まえてそれが目的としているパケットなのかどうかを調べなければいけないということ、そして、tcpdumpのようなツールが悪意で使われればどうなるかということも十分にご承知のことと思う。

先日、その存在が認められたCarnivoreも、原理的にはtcpdumpのような装置だ。とにかくすべてのパケットをその怪物の口にほうり込み、じっくり味わったあとで不要な通信ならペッと吐き出す、オイシイ通信なら飲み込む、という仕掛けになっているらしい。もちろん、悪用されればひど

いことになる。犯罪とは無関係の通信も、とにかくFBIのCarnivoreの口に入ってしまうわけだから、一般の通信利用者のプライバシーは何から何までFBIに筒抜けということになる。

FBIは当然、「Carnivoreは、令状に記載された範囲の通信以外は、決して記録しない」と言っている。このあたりは日本もアメリカも同じだ。つまり「警察を信用してほしい」というわけである。

盗聴装置はオープンソースで

別にアメリカ人がエライというつもりはないのだけれど、アメリカの盗聴法は、日本よりも手続き的な制限が厳しい。また、警察が捜査令状を申請しても、裁判所で徹底的に調べられ、市民のプライバシーを侵害する恐れのある盗聴捜査は決して許可されない。そしてアメリカのパンピーも「警察に知られて困るようなコトはしてないもんネ」なんて間抜けなことは決して……いや、めったに言わないらしい。中にはやっぱり、「FBIがどうかしたの？」なんていうノーテンキな人もいるのは事実らしいのだけれど。

しかし、7月にCarnivoreの存在が公になると、ACLUだとかEPICのような有名な団体をはじめ、アメリカの市民社会が「無関係の通信まで、FBIにチェックされる可能性が強い」猛然と攻撃に出たのである。

その攻撃の内容が面白い。Carnivoreシステムのソフトウェアをオープンソースにしろ、というのである。これは思いつきなんかじゃない。この主張は、アメリカの国会レベルでちゃんと議論されている。下院法務委員会の憲法小委員会は、有名な暗号学者のMatt Blaze氏を証人に呼んで、Carnivoreをオープンソースにするべきかどうかという公聴会を開き、ここでMatt Blaze氏は「Carnivoreのような複雑、かつ高度なセキュリティが要求されるシステムは、オープンソースにすることによりシステムを強化することが期待できる」とし、Carnivoreはオープンソースにするべきだと主張した。

そして、人権団体のEPICは、連邦地方裁判所に対して情報公開法に基づいて技術情報の開示を要求、裁判所はこの要求にすみやかに答えるようにと命じたという。日本じゃ、盗聴装置の外見を公開して「公開しました」なんて言ってるんだから……。

で、FBIは当然、公開したら、悪用されるかもしれない、というような理由で抵抗しているそうだ。しかし、tcpdumpだってsnifferだって、ソースが完全に公開されているというのに、いまさら悪用もヘッタクレもないだろうにと思う。日本には「やましいことをしてないから、盗聴されてもかまわない」なんて言うような人も少なくないらしいけど、やっぱり「公開してやましいことがないんだったら公開すればいいじゃない？」と言いたいぜ。

●

盗聴法については、言いたいことは山ほどあるけど、それはここでは遠慮しておこう。しかし、重要なことは、ソフトウェアというやつは、意図的な不正はともかく、必ずバグがあり、誤動作するという事実だ。万一、当局に悪用の意図がなくても、Carnivoreのような敏感な情報にアクセスするソフトウェアの動作は100%保証されていなければならない。そして、そんな保証なんて誰もできない。

だからこそオープンソースにしなければいけない、と考える人たちが増えている。コンピュータ仕掛けの文明に生きる現代人にとって、透明で安全な社会を実現するひとつの解決策として、オープンソースという選択肢が、少なくとも海の向こうでは現実のものになろうとしているのである。

Profile

やすだ ゆきひろ

生業はテクニカルライター。原稿書きのかたわら、(株)市民電子情報網のボランティア社長兼技術スタッフとして、NGO向けプロバイダのネットワーク運営に携わる。

ドクターShiodaの

ギョーカイSnapShot

PC業界今月の話題

文：塩田紳二
Text : Shinji Shioda

8・10 Netscape 6PR2発表
8・2 カルデラのSCO事業部買収
8・2 Red Hatがエリクソンと提携
7・31 Windows 2000 SP1

今月はちょっと大変で2週間もアメリカに来ている（つまり、この原稿はアメリカでの仕事の合間に書いているわけです）。いままで1週間程度だったのですが、2週間ともなると持ってきた荷物も多いが、イベントの資料などで帰りも荷物がいっぱいという感じ。もっとも、大半は、こっちで贅沢三昧の買い物をした結果なのであるが……。

インテルゆれてる

さて、次世代のCPUであるPentium 4のメインメモリにRambusメモリを使おうと考えていたインテルだが、ここにきて、ようやく厳しい状況に対応し始めたようである。というのは、インテルの予想通り、Rambusがはやることはなく、いまだにRambusメモリ（RDRAM）が高価なまま値段が下がらないのである。

ひとつには、インテル自身のチップセットのバグで、パソコンでのRDRAMの利用に「ケチ」がついたし、メモリメーカーは自分たちが主導権を握れないので、どちらかというとDDRDRAMを推したい。しかも、DDRなら既存のSDRAMと同じデバイスですむため、RDRAMよりは作りやすく、コストも下げやすいのである。

これに対してRambus社は、DDRなどにも使われる技術の基本特許を取得。DDR陣営に対して、特許使用料の要求という圧力をかけはじめた。メモリメーカーは、日本の大手電機メーカーなどほとんどが大企業。これに対してRambus社は自社工場も持たないベンチャー企業である。特許も小さな会社が大企業につぶされないための作戦と考えれば無理もない。どちらにせよ、次世代のメモリはDDRになりそうにしても、その次はどうなるかがわからないといった感じだ。

そういう状況を反映してか、現状の見方は今年中にRDRAMへの移行を行うのは無理だろうし、来年もどうかわからないというところ。しかし、それではインテルも困る。そこで、いままでは言わなかったが、次世代のPentium 4でもDDRを使えるようにしようという話が出始めた。

そうなると話はこじれてくる。インテルがDDRを採用か、という話が広まると、Rambusの株価がどーんと下がってしまった。インテルとRambusの間には契約があり、このために実際にはインテルは、ほかのメモリシステムを採用しにくい状態だという。

インテルとしては、世の中の大勢の選択としてDDRが選ばれたような状況が望ましく、自らRambusからDDRに乗り換えたような形にはしたくない。そこで、やんわりとそうした大勢を作る方向になっているみたい。たとえば、インテルの開発者向けのWebページにあるベンチマークテストの結果では、同クロックのCPUを使う815Eチップセット（PC133メモリ）と820チップセット（RDRAM）は、815Eを使ったマシンのほうが速くなっている（http://www.intel.com/procs/perf/PentiumIII/からたどれる）。

Rambusが問題を解決するには、メモリの価格を下げるしかないのだが、そのためには誰かに大量にメモリを買ってもらわねばならない。それには、ほかのメモリよりも高性能であることを見せなければならないが、このテスト結果はちょっと致命的。

一応、インテルはRambusに対してのフォロー計画（RambusとDDRの価格差をメーカーに補填するらしい）はあるようだが、なにか「別れる前の親切な彼」って気がするのだが。

マイクロソフト値引きする

マイクロソフトは、次期コンシューマー向けのWindowsであるWindows Millennium Edition（Me）の価格を30ドル値下げして59ドル95セントにすると発表した。これは、以前に発表したWindows 98からのアップデートパッケージの価格である89ドル95セントを改訂したもの。ところがWindows 95からのアップグレードには利用できず、95ユーザーは30ドルよけいに払う必要がある。Windows 95はいまだに動いているマシンが多数あり、多くのユーザーはWindows 98に換える必要があるとは思っていない。また、企業内のWindows 95ユーザーは、一般家庭向けのMeに魅力も感じていない。

マイクロソフトのこの値引きは、一般家庭の比較的新しいマシンを使っているユーザーをねらったもの。いまだにWindows 95ユーザーがいることからわかるように、パッケージとしての出荷本数を見れば、Windows 98、同Second Edtionと減ってきているのだ。

実はWindows Meには、次世代のWindowsへの移行の足がかりという重大な使命がある。たとえば、レガシーデバイス（プリンタポートやシリアルポートなど）を排除し、高速に起動するマシンに対応している。また、MS-DOSモードのサポートもない。つまり、Windows Meが動作するマシンは、次のコンシューマー向けNT（Windows 2000 Personal）に移行が可能であり、逆にこのWindows Meが動作できないマシンや、アップグレードが不可能なアプリケーションを動かさざるを得ないマシンは、移行はかなり難しい。

今後登場するマシンは、ほとんど対応可能なはずだが、過去に出荷したマシンや古いアプリケーションを使っているシステムは移行が難しいため、マイクロソフトとしては、早く新しいマシンに移ってもらいたい。家庭ユーザー向けに簡易ビデオ編集の機能などを取り込んではいるが、多くのユーザーにとっては、移行を促すほどの魅力は感じられない。

そこで登場したのが値引きである。だいたい300～400ドルも出せば、OSがプレインストールされているちゃんとしたマシンが手に入る。この値段に対して、89.95ドルは、ちょっと高い。Windows 95が使われていたころに比べてハードウェア価格はかなり下がっている。95ユーザーから見れば、Windows 98ユーザーよりも30ドルもよけいに払って、危険を犯してアップグレードするよりも、買い換えのほうが良く見えるかもしれない。

逆にWindows 98ユーザーは、安くアップデートできるという優越感を味わいつつ、マイクロソフトの売り上げに貢献できるわけだ。

だけど、そうそううまくいくんでしょうかね。英語環境だと、Linuxでもオフィス系ソフトがそろってきているので、Linuxって選択肢もあるんじゃないでしょうかね。

コンパックのiPaq H3600が店頭から消えた？

コンパックが米国で発売中のiPaq H3600というPocketPC（Windows CEマシン）がどこにいってもない。筆者も米国取材のついでにシリコンバレーやロサンゼルスのショップを回ってみたが、どこも売り切れで、ある店では入るとその日のうちに売れてしまって、次はいつ入るかわからないとのこと。

ついにWindows CEの人気爆発か、というとそうでもないらしく、相変わらず米国では、Palmがはやっているとのこと。

実は、このH3600ではLinuxが動くのだ。コンパックは、DEC時代にPDAでLinuxを動かす研究に取り組んでおり、そのためのマシンまで試作していた。Linuxバイナリなどは、http://www.handhelds.org/にある。もし、iPaqを入手できた人がいたら試してほしい。

最近では、こうしたPDA分野へのLinuxの進出が加速度的に行われている。MIPSやARM、SHなどモバイル向けRISCプロセッサへのLinuxのポーティングが進んでおり、これらを応用した製品などもめだってきたからだ。IBMも試作品ではあるが、ARMプロセッサを使った腕時計型のコンピュータを発表したが、これにもLinuxが使われている。

またレッドハットは携帯電話の大手メーカー、エリクソンと提携して組み込み分野へ進出の予定とか。今年は、ハンドヘルドマシンに人気が集まるという予測もあり、この分野でもLinuxはブームになるかも。

人気沸騰のコンパックiPaq H3600
http://www6.compaq.com/products/handhelds/pocketpc/index.html

日刊アスキーLinux on Linux magazine

http://www.linux24.com/

日刊アスキー Linuxの舞台裏

～IBMのカンファレンスに行ってきました～

8月14日から17日にかけて、米国ラスベガスにて、IBMのデベロッパー向けカンファレンス「Solutions 2000」が開催された。おりしも米国ではLinuxWorld Conference＆EXPO 2000 San Joseが開催されており、IBMのLinux関連の発表（たとえばRed Hatとの提携話など）はSan Joseで行われていたわけだが、ラスベガスでも、Linux互換のAPIを実装したIA-64用次世代AIX（IBMのUNIX）である「AIX 5L」の発表（http://www.ibm.com/servers/aix/news/に関連情報あり）など、発表やデモンストレーションがあったのでお伝えしたい。

（日刊アスキー Linux編集部・吉川大郎）

LinuxとIBM

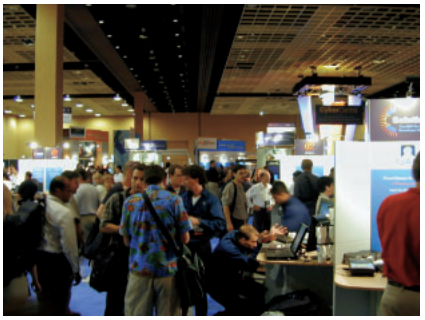
マルチプラットフォーム戦略（サポートしているどのプラットフォームでも、同一の環境を実現する）をとるIBMが、Linuxのサポートを表明した時点で、さまざまなIBM製品がLinuxに対応するのはある意味当然のことだ。しかし今回のSolution 2000の取材を通して、現在のLinuxは「IBMがサポートするOSのひとつ」という以上の扱いを受けているといった印象を持った。

たとえば展示会場には、「WebSphare Village」、「Server Village」、そして「Linux cafe」と、3つの大きなエリアが設置されていた。EJB（Enterprise JavaBeans）の実行環境を提供するWebアプリケーションサーバで、IBMのeビジネスのブランド名として用いられている「WebSphare」や、それを支えるハードウェア＝「Server」とともに、Linuxの名前で1エリアが作られていたわけだ。

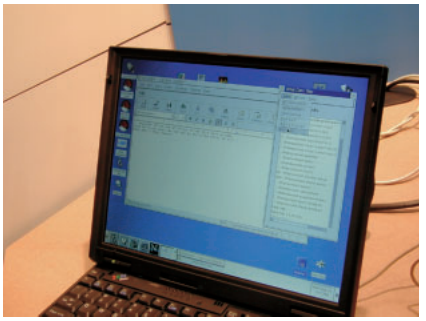
また、IBMがプレス向けに配ったバインダーに綴じられている資料も「Software」、「Hardware」、「Linux」といった分類になっていた。本来ならば「Software」の一部に入ってもおかしくないのに、Linuxだけ特別扱いというわけだ。「Linuxが急拡大」などという文言は今やなんの新鮮味もないが、こうして他のOS以上に取り上げられている姿を見ると、Linuxがいかにフォーカスされているのかがわかる。

IBM製アプリケーションツールキットがLGPLに

それでは、2点ほどSan Joseで発表



展示会場。WebSphere Village、Server Village、Linux cafeのほか、パートナー企業各社による展示が行われている。初日はレセプションパーティのような形で夕方開場。場内には食事や飲み物も用意された。



Via VoiceのLinux版。アプリケーションの起動からワードプロセッサへの入力、クリップボードまで、操作のすべてを音声による指示で行うことができる。日本語は入力メソッドなどの問題で、英語ほど簡単に移植されるというわけではないという。

SMP構成のコンピュータにおける並列処理のための通信テクノロジー「CC-NUMA」を採用したエンタープライズ向けサーバ「NUMA-Q」にもLinuxが搭載されていた。

されていないプレスリリースがあったのでご紹介しよう。

まず1点目が、アプリケーション開発ツールキット「SashXB for Linux」のソースコードがLGPLライセンスのもと、GNOME.orgにて公開されたというものである（GNOME.orgでは、「SashXB for GNOME」として公開されている）。URLはhttp://oss.software.ibm.com/developerworks/opensource/sashxb/。SashXB for Linuxは、WDE（Weblication Development Environment IDE）と銘打たれているもので、JavaScriptやXML、HTMLなどを使って、ネットワークベースのデスクトップアプリケーションを、CDE（Common Desktop Environment）に則った形で構築するためのツールだ（IBMでは、こうした最近のWeb技術を融合した形のデスクトップアプリケーションのことを、「Weblications」と呼んでいる）。

このツールは、個人が使うものも当然作成できるわけだが、企業内のクライアントソフトウェアを作成する場合にも有効だろう。IBMのWebサイトでは、SashXB～で作成されたアプリケーションを、WebSphereやLotus Dominoのクライアントとしても使用することができる点を明記している。

SashXB for Linuxでは、HTMLのレイアウトエンジンにMozilla.orgの「Gecko」を、XMLのパーサにはGNOMEの「Xerces」を使用している。なお、同ソフトは、IBMのalphaWorks（実験的なソフトウェアがいくつも置いてある、IBMのアルファ版専用サイト・http://www.alphaworks.ibm.com/）がWindowsプラットフォームのものを公開していたのだが、今回Linuxに移植された。

次に、IBMのPCサーバ「Netfinity」シリーズについての発表だ。これは、IBMのPCサーバであるNetfinityをエンタープライズレベルに引き上げるための計画「X-Architecture」が、Linuxにも対応したという発表である。X-Architectureは、サーバの自己診断機能、IA-64アーキテクチャ、PCI Hot Plugやそのほかの新しいI/O、SAN（Storage Area Network）、クラスタリング、セキュリティといった、エンタープライズ用途に不可欠な技術をNetfinity上に実装していこうというもので、こうした技術がNetfinityにLinuxを搭載した場合でも使用できるようにしていくという。こうすることで、Linux搭載のNetfinityが、エンタープライズ分野で活躍する下地が作られることになる。また、それに伴い、Netfinityのサーバマネジメントツールである「Netfinity Director」や、LEDなどでサーバの異常を知らせる「Light Path Diagnostics」といった機能も、Linuxで使用可能になる。

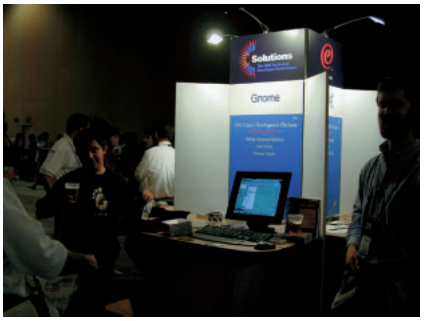
以前、Linuxをインストールした複数のNetfinityを接続したシステムを構築しているISPを取材したことがある。そのシステムでは、ISPの仕事を行うNetfinityにはLinuxがインストールされていたのだが、システム全体を管理するNetfinityは、サーバ管理ツールなどがWindows NTにしか対応していなかったため、Windows NTがインストールされていた。X-Architectureでは、Netfinityがエンタープライズに利用されるわけだから、そのシステムを管理するサーバマネジメント機構も同時にLinuxになるというわけだ。

●

今回のSolutions 2000は、IBMが自社製品をe-businessに向けた形で体系化したe-businessフレームワーク「WebSphere Foundation」が発表されて初めてのカンファレンスだった。このフレームワークの中でLinuxはただのプラットフォームに過ぎないわけだが、今後の成長に対してIBMが力を入れていくという点において、XML、SOAP、Java、といった技術と同じくらい重要な存在のひとつといえるだろう。

S/390、RS/6000、Netfinityをサポートする SuSE。展示会場では、SuSEのほか、Red HatやTurboLinuxなどのディストリビューションが展示を行っていた。

「IBM Small Business Pack for Linux」。グループウェアである「Lotus Domino」、RDBMS「DB2」、EJBによるアプリケーションサーバ「WebSphere」が、オールインワンでバンドルされている製品。



GNOMEもGNOME 1.2を展示。会場ではHelix GNOME Preview 2のCDを配っていた（というよりも、目が合った人の手にムギュッと突っ込んでいた）。静かに話をしているブースが多い中、異様なテンションで盛り上がっていたのが印象的。GNUステッカーも同時にもらった。

正しきタコへの道

初級Linuxer養成講座

第13回　入力と出力

大量のデータを一気に処理することを得意とするLinuxを使いこなすには、「入力と出力」の考え方に慣れておくことが重要である。WindowsなどのGUIに偏重したOSを使っていると忘れてしまいがちな「入力と出力」について、簡単にさらってみる。Linuxに限らず、コンピュータを何かの役に立てようとするときには、入力と出力について考えることがとても重要なのだ。さらに、Linuxならではの入出力機能「リダイレクト」を紹介しよう。

文：竹田善太郎
Text : Zentaro Takeda

オンラインゲーム、チャット、掲示板といった**リソース喰い**のインターネットアプリケーションには興味がない筆者だが、それでもほとんどの作業を自宅でしているので、月々の電話料金はけっこうな額になる。インターネットへの接続時間は毎月80時間前後なので、基本料金などを含めて2万円を超えることはめったになく、同業者の中ではかなり少ないほうだとは思うのだが、**メールを1通受信するだけなのに10円もの通話料をとられる**という状況には、どうにも我慢がならなかった。高いはずの携帯電話ですら、たとえばiモードでのメール受信なら1円単位の金額で済むのである（でもiモードのメールでは仕事にならないが）。

数年前の営業開始直後に、**高額な加入料金**を支払って加入した地域のCATVは、今年の春になってやっとインターネット接続サービスを開始したものの、加入者の大半が住んでいるはずの集合住宅への対応は**教科書どおり**のお粗末なもので、受信者負担で建物側の設備を更新しなければサービスは受けられないという。マンションの管理組合に提議して、設備更新の決議にまで持ち込むのも面倒だ。高額な加入料金や月々数千円にもなる視聴料金は、将来のインターネットサービス開始に期待して投資してきたつもりだったのだが、どうやら**すべて無駄**になってしまったようだ。

そんななか、8月上旬には当地域でもNTTのIP接続サービス（商品名「フレッツ・ISDN」）が始まることになったので、受け付け開始と同時に申し込みをしたのだが、NTTからの連絡は2週間先になるという。それから手続きが始まったとしても、実際に開通するのは1カ月くらい先になりそうだ。あいかわらずNTTの事務処理は異様に時間がかかる。

「あてにしないで気長に待とう」と思っていた矢先に、今度は「今年度中にISDN回線を使った一般家庭向けのSDSLサービスを始める」という記事が一部の新聞に掲載された。一般紙のすっぱ抜き記事なのであまり確度の高い情報ではないのだが、上下方向各1Mbps程度の常時接続サービスを、月額1万円以内で始めることを検討しているということだ。このニュースが本当なら、「フレッツ・ISDN」は提供が本格化する前にすでに**時代遅れのサービス**ということになってしまう。もっとも、DSLによるサービスは、回線の条件や交換機などの制約が多いそうなので、提供不可能な地域もあるだろう。そのような場所ではフレッツ・ISDNしか選択肢がない、ということになる。しかし、将来のサービスの展望をユーザーに一切示さず、**企業機密**という名の厚い壁の内側から市中のようすをこっそりとうかがいつつ、新しいサービスを**小出しに**してくるNTTの商法は、ユーザーとしてはとても歯がゆい。

データの流れ

GUIのアプリケーションを使っているときには、あまり意識することはないのだが、現在使われているほとんどのコンピュータ（そしてプログラム）は、**データを食べて、データを排出する**という動作をすることで、世の役に立っている。もちろん、一方的にデータを取り込むばかりで、何の結果も出してくれないプログラムとか、外部からのデータは一切受け付けないで、データを延々と吐き出し続けるプログラムというものも存在するの

だが、きわめて特殊な例といえるだろう。

プログラムに与える（プログラムが取り込む）データのことを、専門用語では**入力（input）**と呼んでいる。また、プログラムが吐き出すデータは、**出力（output）**と呼ばれる。単なるデータ（情報）の出し入れなのに、「力」を入れたり出したりするとは大げさな気もするが、もともと、入力、出力という言葉は、モーターなどの動力機械に対して使われていたものらしい。それが、エレクトロニクスの分野でもそのまま使われるようになったのだろう。

プログラムに入力するデータを用意するのは、人間（ユーザー）の責任で行うことになる。キーボードから文章を打ち込むのも「入力」だし、あらかじめ用意されていたテキストファイルを流し込むのも「入力」である。あるいは、MP3エンコーダに音楽CDのデータを与えるのも「入力」である。ともかく、コンピュータは原則として自分でデータを作り出すことはできないので、ユーザーが何らかの形で入力するデータを用意しなければならないのだ。

一方の「出力」については、ディスプレイ画面、プリンタなどの「出力装置」にデータが送られて、ユーザーが読んだり見たりできるようになる。あるいは、結果をファイルに保存しておいて、あとで利用できるようにすることもある。音声データを扱うプログラムなら、入力したデータの出力は「スピーカー」に対して行われることもあるだろう。

Linuxのコマンドラインから利用できるプログラム（コマンド）の多くは、入力元と出力先をユーザーが**自由に切り替え**られるようになっているのだ。GUIのプログラムでも、入力ファイルや出力先は、ユーザーがメニューやダイアログボックスなどから選択できるようになっていることがあるが、基本的にはプログラムを作った人が決めた出力先しか選べない。

プログラムを**単体で使っている**場合にはこれでも特に問題ないのだが、あるプログラムの結果を別のプログラムでさらに加工しようとする場合、使い方に大きな違いが出てくるのだ。

標準入力と標準出力

Linuxでは、コマンドによって切り替えが可能な入力と出力を**標準入力と標準出力**と呼んでいる。これは、英語の「standard input」、「standard output」という言葉を日本語に直訳したもので、初めて目にする人には何のことか想像もつかない言葉だと思うが、Linux（UNIX）の世界ではなかば**常識**として使われている用語である。

もう少し分かりやすい言葉に言い換えると、標準入力とは**データの入り口**、標準出力は**データの出口**ということになる。プログラムを機械部品の一種、たとえば水道につなげて使う浄水器のような機器にたとえると、標準入力は水が入ってくる栓、標準出力は水が出てゆく栓ということになる（**図1**）。水道の場合は、出入り口の栓にさまざまな部品をつけることで、水の取り入れ口や出口を自由に変更できるわけだが、Linuxのコマンドも、同じようにデータの取り入れ先や吐き出し先を自由に変更できるのである。

■出力の切り替え

では、Linuxの「標準入力」と「標準出力」とはどのようなものなのか、実例を見てみよう。まず、「標準出力」から始める。通常、Linuxのコマンドをコマンドラインから起動すると、コマンドの処理結果はディスプレイの画面上に表示される。これは、Linuxのプログラムの「標準出力」は、なにも指定しない場合にはコンソール画面に接続されているからである（**図2**）。たとえば、簡単な例を見てみると、ファイルの内容を出力するコマンド「cat」を使うと、ファイルの内容を画面に出力できる。

```
$ cat file1.txt
contents of textfile
```

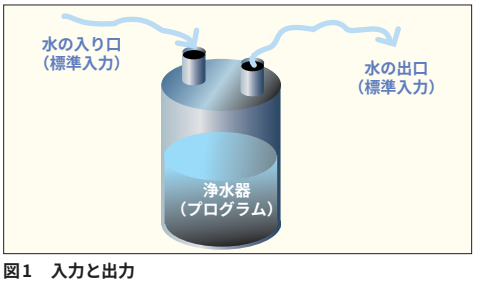


図1　入力と出力
機械や電気の分野でも、液体やエネルギーが装置に入ってくることを「入力」、出てゆくことを「出力」と呼んでいる。Linuxのプログラムを「浄水器」にたとえると、水が入る栓がデータの「入力」、出てゆく栓がデータ「出力」に相当する。

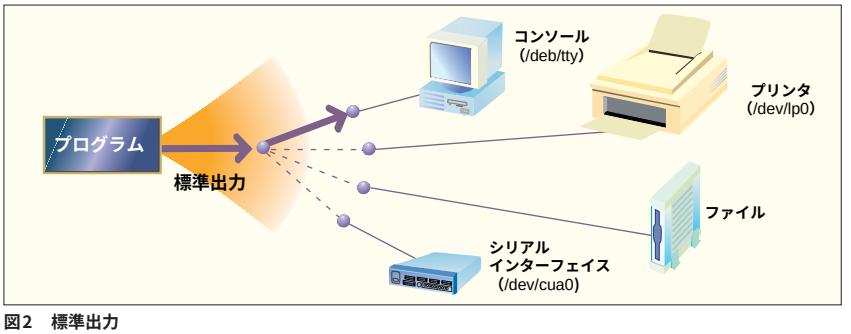


図2　標準出力
各プログラムには「標準出力」というデータの出口が1つ用意されている。何も指定しないと、標準出力に流れ込んだデータはコンソール画面（モニタ画面やターミナルエミュレータのウィンドウ）上にそのまま表示される。

これだけ見ると、「catコマンドとは、ファイルの内容を画面に出力するコマンドである」と思えるかもしれないが、catコマンドのオンラインマニュアルを見てみると、まったく違うことが書かれている。

```
$ man cat
```

を実行してみると、「cat — ファイルを連結して、**標準出力へ表示**する」というような説明になっているはずだ。「コンソール」とか「画面」という文字は見当たらない。catコマンドは、画面への出力専用に作られたコマンドではないのである。

では、catコマンドの出力を、画面以外に切り替えるにはどうすればよいだろう。Linuxのコマンドラインでは、このような場合に**リダイレクト文字**というものを使うことになっている。たとえば、file1.txtというファイルの内容をcatコマンドで表示して、表示結果をfile2.txtというファイルに振り向ける場合は次のようなコマンドラインになる。

```
$ cat file1.txt > file2.txt
```

上のコマンドラインの中の「>」がリダイレクト文字である。これを見れば**ああ、あれのことか**と思い当たる読者も多いだろう。本誌のほかの記事の中でも、コマンドの結果をファイルに保存するような場合に、頻繁に使われているからだ。

ちなみに、メールの本文中やWebページの掲示板の中で、発言の一部を特定の個人に振り向けたいときなどに、「**＞ ○○さん**」というような表記をしているのをよく見かけると思うが、これはまさに、リダイレクト文字の機能にならった表現である。

ところで、「リダイレクト」とは英語では「手紙などを別のあて先に送りなおす」とか「宛名を書き換える」というような意味になる。日本語で簡単に言うなら、**転送**とか**回送**という言葉になるだろうが、Linuxの世界ではカタカナ語の「リダイレクト」がそのまま使われている。標準出力（そして後述する標準入力など）の出力先（入力元）を切り替える操作のことを**リダイレクトする**と表現しているのだ。

GUIアプリケーションでも、処理の結果を自分の好きなファイルに保存することは可能で、たとえば**名前を付けて保存**などのメニューを選択して、ダイアログボックスの中でディレクトリやファイル名を入力する。しかし、このような操作のことを「リダイレクト」と呼んではいけない。単に、保存先のファイルを切り替えているだけで、たとえば、画面に直接出力したり、プリンタなどの周辺機器に出力したりすることはできないからだ。

リダイレクトが**エラい**のは、ファイル以外の「デバイス」にも出力を振り向けられる点である。たとえば、プリンタが正しくインストールされていて、テキストファイルの印刷ができるようになっているLinuxの環境なら、file1.txtの中身を直接プリンタに送ることができる。

```
$ cat file1.txt > /dev/lp0
```

ここで、「/dev/lp0」とは、Linuxにおいてプリンタの**窓口**に相当する「デバイスファイル」というものだが、普通のファイルと同じように、このデバイスファイルに対してデータをリダイレクトすると、それに関連付けられている装置（ここではプリンタ）に対してデータが送られるようになっている（**図3**）。

ちなみに、最近のプリンタでは、テキストファイルを直接印字しようとしても、きれいに出力できなかったり、無意味な文字が出力されてしまう機種

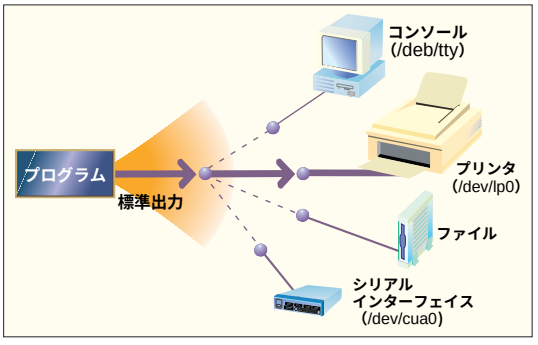
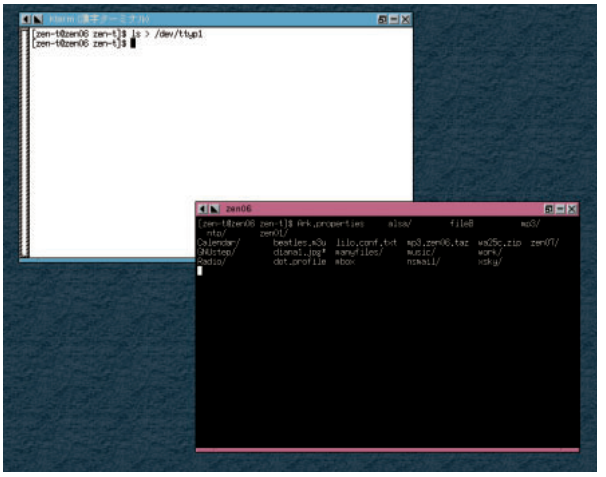


図3　標準出力の切り替え
「標準出力」は各プログラムに1つずつしかないのだか、その接続先をユーザーの操作で切り替えることができる。シェルのコマンドラインの末尾に「> /dev/lp0」を追加すると、コマンドの標準出力はプリンタにつなぎ換えられる。



画面1　lsコマンドの出力を別の「コンソール」に出力する
lsコマンドの標準出力を「/dev/pty1」、「/dev/pty2」などにリダイレクトすることで、コマンドを実行中のコンソール（ktermなどのターミナルエミュレータ画面）とは別のコンソールに表示させることもできる。単なる「いたずら」にも使えるのだが、たとえば実行中のプログラムの進行状況を、別のウィンドウで監視したいような場合に便利である。

が多いので、このような方法でプリンタを使うことはお勧めできないのだが、少なくとも大昔の**文字しか印字できない原始的プリンタ**の時代には、このような簡単なコマンドでプリンタを使うことができたのだ。

もう少し面白い例としては、コンソール画面へのリダイレクトが考えられるだろう。たとえば、X Window System環境で複数の画面（ktermなどのコンソール画面）を開いている場合、あるウィンドウ上で実行したコマンドの出力を、別のウィンドウに表示することができる（**画面1**）。

```
$ cat file1.txt > /dev/ttyp1
```

ここで、「/dev/ttyp1」というのは、プリンタの例と同じくデバイスファイルの一種なのだが、これは特定のハードウェアを指しているのではなく、X上の「ターミナルウィンドウ」（ktermやxtermなどのアプリケーション）のひとつを表しているものと考えておけばよい（実際にはそうとはいえない場合もあるのだが、ここでは難しいことは抜きにしたい）。この「/dev/ttyp1」というデバイスのことを**仮想コンソール**などと呼んでいる。ちなみに、Xで＜誰がどの仮想コンソールを使っているか＞を調べるには**w コマンド**を使えばよい（**リスト1**）。

リストのように、ログインしているユーザーの一覧が表示されるが、2列目の「TTY」の項目が、そのユーザーが使っている「仮想コンソール」を表している（Linuxのディストリビューションによっては、「ttyp?」ではなく「pts/?」という形式で表示されることもあるが、最後の数字が仮想コンソールの番号を示している）。同じユーザーが複数のターミナルウィンドウを開いているような場合は、「ttyp1」、「ttyp2」、「ttyp3」というように、ターミナルウィンドウ1つずつにそれぞれ仮想コンソールが割り当てられているのがわかるだろう。ちなみに、「時計」や「カレンダー」など、ユーザーからの入出力が一切ないプログラムや、メニューで操作するタイプのGUIアプリケーションには、仮想コンソールは割り当てられないことが多い。

さらに特殊な例では、すべての入力を破棄してしまう/dev/nullデバイスがある。

```
$ cat file1.txt > /dev/null
```

上のコマンドを実行しても、見た目上はなにも起こらない。画面には何の文字も出力されないまま、コマンドの実行は終了する。catコマンドの出力はどうなってしまったかというと、/dev/nullというデバイスファイルにリダイレクトされたのだが、このデバイスは、データの入力を受け付けるだけで**なにもしないデバイス**なのである。お金を無駄遣いしてしまうことを**お金をドブに捨てる**と表現するが、/dev/nullはこのような場合の**ドブ**に相当するものと考えればよいだろう。

一見無意味な/dev/nullであるが、たとえば、プログラムの動作チェックをしたり、プログラムを無人運転させるような場合、画面上への出力を行わないようにするのに使われている。あるいは、**上司の悪口を/dev/nullへ向かって叫ぶ**というような使い方をする人もいるらしい。

```
$ echo "この△○××！！！" > /dev/null
```

もっとも、コマンドライン上にはしっかり悪口が表示されるのだから、このような使い方にはあまり意味はないのかもしれないが、ストレスの解消にはなるかもしれない。

出力のリダイレクトに利用できるファイルやデバイスファイルは、上で触れたもの以外にも多数あるのだが、これについてはおいおい覚えていけばよい。

■入力の切り替え

プログラムへの入力も、出力と同じように「リダイレクト」によって切り替えることができる。コマンドラインをある程度使える読者ならわかっていると思うが、入力の切り替えに使う「リダイレクト文字」は、出力の場合の逆向きの＜である。今まで、catコマンドは

```
$ cat ファイル名
```

という形で入力ファイルを指定したの

```
$ w
  9:50am  up 84 days, 12:25,  2 users,  load average: 0.08, 0.02, 0.01
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
zen-t    ttyp0    -                Mon 1pm 18:35   0.20s  0.20s  -bash
zen-t    ttyp1    -                Mon 1pm 18:20   0.00s  0.01s  w
zen-t    tty1     zen07             9:50am  0.00s  0.15s  0.03s  -bash
```

リスト1

だが、同じことを、リダイレクト文字を使って記述すると次のようになる。

```
$ cat < ファイル名
```

たとえば、file1.txtの内容を標準出力に表示したい場合には、

```
$ cat < file1.txt
```

とすればよい。入力と出力の両方をそれぞれリダイレクトすることもできる。

```
$ cat < file1.txt > file2.txt
```

上のコマンドラインは、file1.txtの内容をfile2.txtというファイルに出力する。つまり、

```
$ cp file1.txt file2.txt
```

とした場合と、ほぼ同じことである。

入力の場合も、出力と同様にデバイスファイルをリダイレクト先に指定することができる。たとえば、サウンドカードをインストールしているマシンなら、

```
$ cat < /dev/dsp0 > sound.au
```

というコマンドを実行するだけで、サウンドカードのライン入力やマイク入力からの音声を、PCM形式のファイルに保存することができる（サウンドカードの設定や音量設定などは別途必要になるが、ここでは触れない）。録音を停止するにはCtrl+Cキーを押せばよい。

このように、Linuxでは周辺機器との間の入出力など、一見高度な処理も、リダイレクトの機能を使えばコマンドラインだけで簡単にできるのである。

フィルタ型コマンドと非フィルタ型コマンド

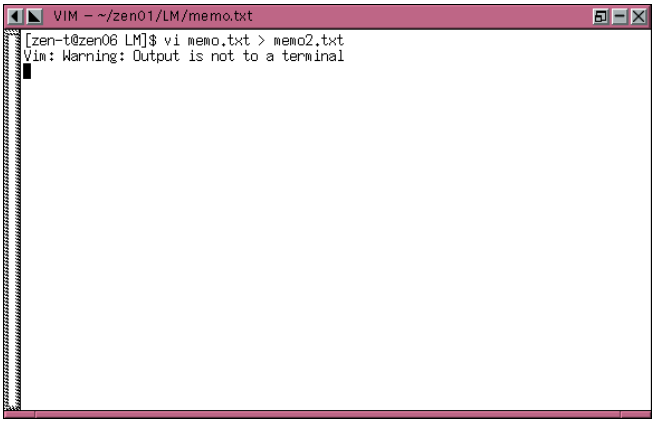
Linuxのコマンドラインでは、すべてのコマンドでリダイレクト文字が使えるわけではない。**リダイレクトが使えないコマンドもある**のだ。たとえば、テキストエディタのviのようなアプリケーションでは、コマンドの起動時に編集するファイルを指定できるだけで、出力先をコマンドライン上でリダイレクトしたり、デバイスファイルからの入力を編集するというような使い方はできない。

正確には、コマンドラインで出力をリダイレクトできないこともないのだが、実際に試してみるとわかるように、コマンドを起動しても画面にはなにも表示されないばかりか、編集などの操作もできなくなってしまう（**画面2**）。これは、テキストエディタの画面がファイルにリダイレクトされてしまって、本来表示させたいコンソール上に表示されなくなってしまうからだ。もちろん、この結果作成されるファイルは、本来なら画面に表示されていたはずの「ステータス表示行」や「コントロール文字」などがごちゃまぜになった、**まったく無意味なもの**になる（**画面3**）。

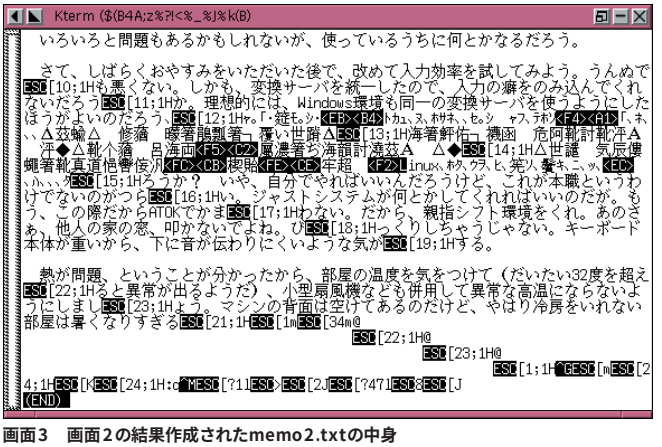
このように、テキストエディタなどのアプリケーションでは、標準入力や標準出力はあまり意味を持たない。一方、ファイル中の文字列検索をしてその結果を表示する**grepコマンド**などは、入力や出力を切り替えることで、さらに便利に使えるようになる。

```
$ grep "検索文字列" < file1.txt > result.txt
```

上のコマンドラインでは、file1.txtの中から「検索文字列」という文字列の含まれている行を探し、見つかった行をresult.txtというテキストファイルに



画面2　viの「出力先」をリダイレクトしてしまった例
「memo.txtを編集して編集結果をmemo2.txtに保存する」つもりで、viコマンドの出力をリダイレクト文字でリダイレクトするように指定してしまうと、viから警告メッセージが表示され、画面上には編集画面が一切表示されなくなってしまう。この状態から抜け出すには、viの「終了」コマンドである「:q」とEnterキーを押すしかない。



画面3　画面2の結果作成されたmemo2.txtの中身
memo.txtの中身らしい文字列も含まれているが、画面表示に使われる制御文字などの「ゴミデータ」が混じっているので文字化けを起こしてしまっている。もちろん、このファイルの中身は「使い物」にはならない。

保存する。

grepコマンドのように、標準入力と標準出力をユーザーが切り替えて、流し込むデータや結果のデータを保存するファイルを自由に設定できるコマンドのことを、Linuxの世界では**フィルタ型コマンド**あるいは単純に**フィルタ**と呼んでいる。

日本でも都市部では必須の設備になりつつある「浄水器」は、英語では「フィルタ」と呼ぶらしいが、この浄水器を思い浮かべてもらえばフィルタ型コマンドの働きも理解しやすいだろう。蛇口につけるタイプの小型の浄水器ではなく、流し台の下に設置するような大型の浄水器には、円筒形の本体に2つの「栓」がついている。一方は水道水が入る口、すなわち「入力」用の栓、もう一方は浄化された水が出てくる「出力」用の栓である。これらの「栓」は、(製品によって若干違うかもしれないが)一定の「規格」に従って作られていて、市販のホースや接続用金具を使って、水道管や流し台の蛇口に接続できるようになっている。浄水器の用途や設置場所に応じて、たとえば「出力」を蛇口ではなく瞬間湯沸し器に接続したり、自動製氷機に接続するなど、「出力をリダイレクト」することができる。入力については、浄水器ではあまり切り替える必要はないかもしれないが、井戸水と水道水など、複数の水源があるような場合には、同じように接続先のホースをつなぎ換えるということも考えられるだろう。

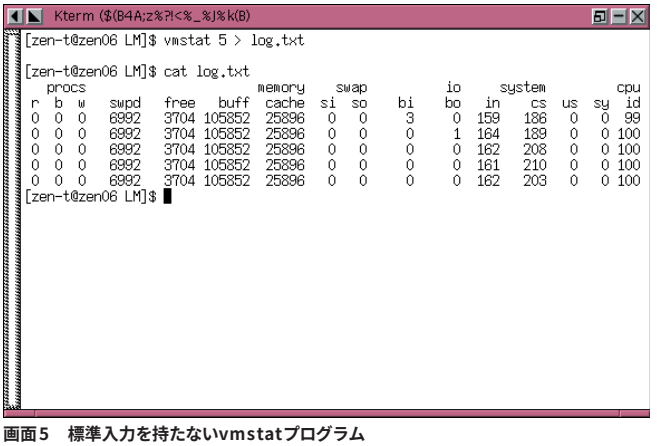
このように、フィルタ型コマンドとは、入力と出力を自由に切り替えられるコマンドのことを指すのだが、Linuxのコマンドの中には、標準入力はあるが標準出力のリダイレクトはできない(あるいは「意味がない」)ものや、外部からの入力は一切なくて、出力のみ切り替え可能ものなど、入出力のどちらか一方しかリダイレクトできないタイプのものがある。入力のみ切り替えできるコマンドとしては、たとえば、テキストファイルをページ単位で表示する「lessコマンド」がある(**画面4**)。出力のみ切り替え可能なコマンドにはいろいろあるが、システムの動作状態を表示する「vmstatコマンド」などがその例になるだろう(**画面5**)。前出のviのような、入力も出力も**リダイレクト無用**のコマンドも含めて、入出力のうち少なくともひとつがリダイレクト不可能なコマンドを**非フィルタ型コマンド**と呼ぶことがある。ただし、出力か入力のどちらか一方でもリダイレクトできれば、フィルタ型コマンドの仲間とみなす場合もあるので、この用語は厳密なものではなく、参考程度にすべきだろう。

しかし、自分が使おうとしているコマンドが、リダイレクトに対応しているかどうかは常に意識しておく必要がある。そうでないと、たとえば、入力のリダイレクトに対応していないコマンドにリダイレクト文字を使ってデータを入力しようとして、目的どおりの結果が得られない、などのトラブルに見舞われることになる。あるコマンドがフィルタ型かそうでないかを見分ける**統一的な方法はない**ので、オンラインマニュアルをよく読んで、「標準入力(出力)」、「standard input(output)」、「stdin(stdout)」などの文字列に注意して、コマンドの入出力データをどのように指定すればよいのかを調べなければならないだろう。

Linuxプログラムの「標準入出力」の基本的な使い方は以上のとおりだが、次回はさらに便利な「リダイレクト」の使い方と、標準入出力の応用である「パイプ機能」について説明しよう。

画面4　標準入力のみ切り替え可能なプログラム「less」

長いテキストデータをページ単位でじっくりと読めるようにする「less」コマンドは、「標準入力」を切り替えることで、ファイルの内容を表示したり、別のコマンドの出力結果を表示したりする際に利用できる。しかし、出力をリダイレクトする意味はまったくなく、viの例と同じように、利用価値のないファイルができてしまう。



画面5　標準入力を持たないvmstatプログラム

Linuxシステムの現在の状態を表示してくれるvmstatプログラムでは、出力をリダイレクトすれば「ログファイル」を作ることができる。しかし、外部からデータを入力することはないので、「標準入力」は持たない。

InterBase 6.0

ミッションクリティカルなシステムで培われた高いスケーラビリティを誇るデーターベース「InterBase」がオープンソースになって公開されました。今回から始まるこの連載では、InterBase 6.0について、徹底解説していきたいと思います。

第1回　InterBaseの概略とインストール

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

数多くの基幹系情報システムで使われているインプライズ社のリレーショナルデータベースInterBaseの最新バージョンがオープンソースとなって登場しました。まずは、InterBaseの概略とインストールまでを紹介しましょう。

InterBaseとは

InterBaseは米国インプライズにより開発、販売、サポートされてきたリレーショナルデータベース（RDB）です。日本では、現在でもInterBase 5.6についてはインプライズによって販売およびサポートが行われています。InterBaseは1986年に最初のバージョンがInterBase Corporationによって開発されています。当時のInterBase Corporationは、元デジタルイクイップメント（DEC、現在のコンパック）にてRDBMS（Relational DataBase Management System：リレーショナルデータベース管理システム）を担当していたメンバーが、管理者を必要としない手軽なRDBMSの構築に興味を抱き、スピンアウトして設立された会社でした。オラクルも同じように、旧DEC社のメンバーによって設立された会社ですので、製品の源流は一緒なのかもしれません。

その当時、InterBaseはVMSやMVSなどのDECのOS上で稼働するものでしたが、HP-UXやSun OS、AIXなどのUNIXに対応していきました。InterBase Corporationはアシュトンテートに買収されましたが、ボーランド（現インプライズ）によるアシュトンテートの買収により、インプライズの製品となりました。

いろいろな会社に吸収されてきた製品ですが、現在まで当初の設立メンバーが、製品のアーキテクチャの改良を行ってきました。そのため、最初のリリースから目標とされてきた、使いやすく、管理者のいらないRDBMSとしてのコンセプトは今も受け継がれています。

InterBaseの導入例

InterBaseはトランザクションログを使用しない（ユーザーに見えない形では使用していますが）履歴型アーキテクチャ（別名バージョニング）を当初より採用していたため、非常にパフォーマンスに優れ、OLTP（Online Transaction Processing：オンライントランザクション処理）向きの製品として数多くの企業に採用されてきました。古くはフィラデルフィア証券取引所、現在ではFirst National Bank of Chicago、City of New Yorkなどで使用されています。変わった例としては、湾岸戦争で使用された米軍の戦車に搭載されています。日本では、ANAグループ、ソニーテクトロニクスなどで使用されており、ハードウェアや商用パッケージの組み込みエンジンから、ワークグループレベルのデータベースサーバまで幅広く使われています。このように非常に高いスケラビリティを持っているのもInterBaseの特長のひとつです。

オープンソース

昨年のInprise Conference(インプライズのプライベートカンファレンス)でInterBase 6.0のベータ版が配布され、そろそろInterBase 6.0の発売時期と思われていた2000年1月3日に、インプライズよりInterBase 6のオープンソース化がアナウンスされました。長い間製品として販売してきたソフトの最新バージョンをオープンソース化することに対する、IT業界およびLinux業界の興味は大変高く、このようなビジネス形態が与える影響について分析したアナリストも多くいました。

InterBaseはオープンソース化に伴い、7月15日に一部のエンジニアがスピンアウトし、InterBaseのソース、コミュニティ、サードパーティとの関係、ニュースグループなどを管理する別の会社、InterBase Software Coporationが設立されようとしています。私もずっとこの流れを見てきたわけですが、この会社がどのような会社であるのか現状ではわかりません。おそらく、近いうちに詳細な発表があるのではないかと思われます。

1月3日のオープンソースの発表以降、InterBaseの開発チームによってソースコードの切り出しと、他社が所有権を有する部分の切り出し、容易なビルド環境の実現などの作業が行われ、7月25日にInterBase 6.0のWindows、Solaris、Linux用のソースコードが公開されました。Linuxプラットフォーム以外のソースコードが公開されたことは、他のプラットフォームへの移植を考えると非常に有意義なことでしょう。

1月3日の発表からソースコードが公開された7月25日まで、なぜこんなに長い時間がかかったのかと疑問を持たれる方もおられると思います。InterBaseのソースコードの管理方法は少し変わっていて、marionというソースコード管理のシステムを利用しています。marionはInterBaseを利用したソースコードのバージョン管理システムで、すべてのプラットフォームのソースコードがデータベースに格納されているそうです。新しいバージョンを作成するときは、データベースのコピーをとって行っているそうです。InterBase 3.xの時代には30以上のプラットフォームをサポートしていましたので、非常に複雑なシステムなのでしょう。筆者は個人的な興味から、OpenStep、MVSなどのプラットフォームのソースを見てみたいと思いますが、今回公開されたソースの中にはそういったプラットフォームの部分は残念ながら省かれてしまっています。

すでにInterBaseのコミュニティである、IBDI（InterBase Developer Initiative）のメンバーによって、FreeBSD 4.0、Netwareなどへのポーティングが始まっています。また、公開されたコードのメンテナンスが行われていますので、もうしばらくすると安定したビルドがあがってくるのでしょう。

InterBase 6.0の特徴

InterBaseは非常にコンパクトなRDBMSですが次のような特徴を持っています。

- **ANSI SQL-92エントリレベルのフルサポート**
- **トリガー、ストアドプロシージャのサポート**
- **Blob、多次元配列のサポート**
- **データベース全体を1つのファイルに格納**
- **マルチ言語のサポート（列ごとにキャラクタセットを選択可能）**
- **ユーザー定義関数（UDF）のサポート**
- **クライアントにイベントを通知するイベントアラータ**
- **ガベージコレクション機能**
- **チューニングレスで最高のパフォーマンスを実現するクエリオプティマイザ機能**
- **シャドウデータベース機能**
- **オンラインバックアップ機能**
- **100% Javaで書かれたJDBCドライバであるInterClientの搭載**
- **InterBaseの状態を監視し、必要に応じてInterBaseやOSの再起動を自動的に行うGuardian機能**

なお、今回リリースされたInterBase 6.0の新機能としては、次のようなものがあります。

- **SQL-3のエントリーレベルのサポート**
- **新しいクライアントツール（IBConsole、現在はWindows版のみ）**
- **分割された識別子（Delimited Identifier）**
- **新しいInterBase Express**
- **Read-Onlyデータベース**
- **新しいデータ型（SQL DATE、TIME、TIMESTAMP）**
- **64ビットIntegerのサポート**
- **組込用の新しいAPIセット（Service API、Install API、License API）**

InterBase 5.xまでに、ふだん使用していくのに必要な機能がすべて入っていましたが、InterBase 6.0からは新しいANSI SQLの規格である、SQL-3の一部の機能をサポートしています。以前から、サーバで実行されるプログラムであるストアドプロシージャや、データの格納・変更・削除などの処理時に実行されるイベントであるトリガー機能などは非常に柔軟性に富んでいました。InterBase 6.0でもこれらの機能はさらに進化しているようです。

また、InterBaseは組み込み用途でも使用していくことを考慮しているため、最低限の機能しか提供していません。たとえば、あらかじめ用意されている関数が少ないなどの欠点がありますが、UDF（ユーザー定義関数）を利用することで、自分の使いたい関数を作成することができます。Windows版ではDelphi、Visual C++、C++Builderを使用することでUDFを作成することができますし、Linux版ではgccやFree Pascal Compilerを使用することでUDFを作成することができます。必要なものを自分で作成することができるのも、特徴のひとつです。

InterBase独自の機能としては、イベントアラータがあります。使い方は**リスト1**のように、トリガーの中でイベントを登録しておき、あとはクライアント側でイベントを受けられるようにしておくだけです（DelphiやC++Builderにはイベントアラータ用のコンポーネントが付属しています）。たとえば、クライアントAがデータを追加したら、**リスト1**のイベントによって追加のイベントが発行され、クライアントBやクライアントCにデータが追加されたことが伝えられます（**図1**）。この機能により、クライアントBやクライアントCがポーリングを行うことなく、クライアントAのデータ追加を知ることができます。

データベースの信頼性を高める機能として、データベースのミラーを作成するシャドウデータベースという機能があります（**図2**）。**リスト2**のようにシャドウを設定するSQL文を実行することで、シャドウデータベースを作成することができます。この機能を利用することで、RAIDなどを使用することなくシステムの信頼性を高めることができます。マスターデータベースとシャドウデータベースへの書き込みは、InterBaseのサーバプロセスによって自動的に行われます。InterBaseには自動ツーフェイズコミットという機能があり、アクティブのトランザクションが複数のデータベースにまたがる場合、InterBaseが自動的にツーフェイズコミットモードに入り、複数のデータベースの一貫性を保ちます。ツーフェイズコミットモード機能を利用するために特別のコードや設定を行う必要はありません。この機能はシャドウデータベースへの書き込み時にも使用され、ツーフェイズコミットを利用することにより、マスターデータベースとシャドウデータベースの一貫性を保ちます。

InterBaseは非常にコンパクトでありながら、このようにたくさんの特徴があります。これらの特徴は、商用データベースとして販売されてきた実績ともいえるでしょう。

InterBaseのトランザクションについて

InterBaseはマルチジェネレーションアーキテクチャという履歴型を採用しているユニークなRDBMSです。RDBMSでは通常、SET TRANSACTIONコマンドを使用することでトランザクションを開始し、COMMITコマ

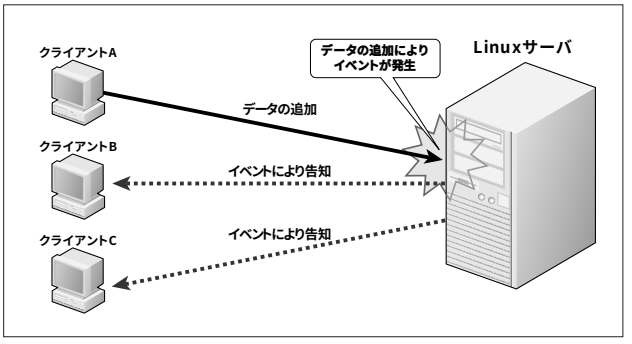


図1　イベントアラータの仕組み

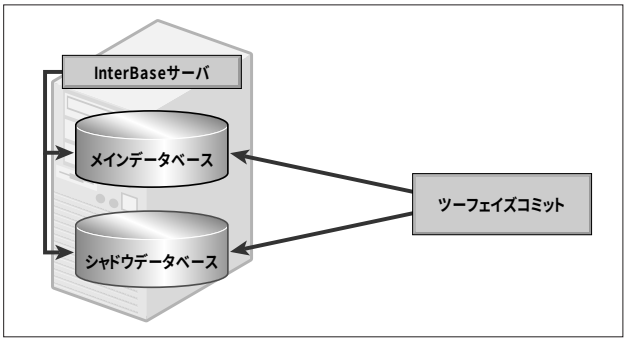


図2　シャドウデータベースの仕組み

▼リスト1　イベントアラータのイベント

```
CREATE TRIGGER POST_NEW_ORDER FOR SALES
AFTER INSERT AS
BEGIN
        POST_EVENT "new_order";
END
```

▼リスト2　シャドウを設定するSQL文

```
/*シャドウの設定*/
CREATE SHADOW 1 AUTO "/home/backupdb/salesdb.shd";
```

ンドにてトランザクションの実行を行います。エラーが発生した場合には、ROLLBACKコマンドでトランザクションを元の状態に戻します。InterBaseでも他のRDBMSと同じようにトランザクションを実行できるのできるのですが、InterBaseの履歴型アーキテクチャがトランザクション処理をわかりにくくしています。

■InterBaseのトランザクション管理

InterBaseはトランザクションログを使用することなく、複数のユーザーからのトランザクションを実現します。よく、トランザクションログを使用しないと、トラブル時にトランザクションのロールバックができないと言われますが、InterBaseはこういった機能を使用せずにトランザクションの一貫性を実現しています。

InterBaseでは、1から始まる連続した番号でトランザクションを管理します。この番号をトランザクション番号と言い、トランザクションが行われるごとにトランザクション番号が1ずつ増加し、2^{32-1}まで連続的に振られていきます。ユーザーがデータベースのバックアップユーティリティであるgbakを使用して、バックアップ／リストアを行うと、この番号は1に戻ります。

テーブルの作成時、InterBaseは自動的にトランザクション番号を管理するための列を追加します。データの更新が行われると、このトランザクション用の列にトランザクション番号が格納され、どのトランザクションが、どのテーブルのどの列をコントロールしているかを判断していきます。トランザクションがコミット、またはロールバックされると、このトランザクション番号を削除し、トランザクションが終了したことを判断できるようにします。

InterBaseはこのトランザクション番号を使用することで、複数のユーザーが同時にデータの読み取りと書き込みをすることを実現しています。あるユーザーがデータの更新を行っている場合、更新される行にトランザクション番号が格納されます。このとき、同一行を他のユーザーが読み取ろうとすると、InterBaseはトランザクション中であることを判断して、前の履歴を渡すかどうかを判断します。前の履歴を使用するかどうかはアイソレーションレベルによって判断します。

InterBaseはトランザクションの処理時、処理中のトランザクションをlimboトランザクションとして扱い、COMMIT、またはROLLBACKの実行により初めてアクティブな履歴として扱います。もし、トランザクション処理中に電源が落ちるなどのトラブルが発生した場合、実行していたトランザクションはlimboトランザクションのままとなりますので、データの不整合は発生しません。このlimboトランザクションはデータベースのスイープ時、バックアップ／リストア時、データベースの読み込み時にロールバックされます。電源が落ちた場合、電源が復帰すればInterBaseは自動的にlimboトランザクションをロールバックしてくれますので、すぐに業務を開始することができます。また、gfixというユーティリティを使用することで、limboトランザクションをコミットすることもできるようになっています。

		SNAPSHOT/READ COMMITTED		SNAPSHOT TABLE STABILITY	
		UPDATE	SELECT	UPDATE	SELECT
SNAPSHOT/	UPDATE	△(注)	○	×	×
READ COMMITED	SELECT	○	○	○	○
SNAPSHOT TABLE	UPDATE	×	○	×	×
STABILITY	SELECT	×	○	×	○

○：衝突することなしにデータ処理を実行
×：データ処理の衝突が起こり、エラーとなる
注：同一範囲をアクセスするときのみ衝突

表1　InterBaseのトランザクションのパターン一覧

InterBaseの持つアイソレーションレベル

InterBaseのトランザクションのパターンを**表1**にまとめました。InterBaseはSNAPSHOT(Repeatable read)、READ COMMITTED、SNAPSHOT TABLE STABILITYという3つのアイソレーションレベルを持っています。SNAP TABLE STABILITYを使用することで、排他処理が可能になります。

■排他処理時の対応

InterBaseは排他処理時に、ロックが解除するまで待つか待たないかのオプションが用意されています。デフォルトではWAITモード（ロックが解除されるまで待つ）です。また、トランザクション処理の宣言時にRESREVING句を使用することで、排他処理を行いたい行（テーブル）を指定することができます。RESERVING句を指定しない場合、InterBaseは排他処理を行う必要があるテーブルをサーバ側で判断し、排他処理を行います。

リスト3はトランザクションの使用例です。

InterBaseは同時に2つ以上のデータベースにわたるトランザクション処理が行われる場合、自動的に分散トランザクションモードとして働きます。すべてのデータベースが正しくトランザクション処理を実行できる場合のみコミット処理が行えます。InterBaseはこのようにツーフェイ

ズコミットをサーバ側で行ってくれますので、プログラマーは特別なコードを書くことなく、複数のデータベースにわたるトランザクション処理を実現することができます。

■履歴型アーキテクチャと同時更新

複数のユーザーが同一行を更新するようなトランザクションを行った場合に、InterBaseはどのように対応するでしょうか。

前述したように、InterBaseは履歴型アーキテクチャを採用しており、トランザクションはトランザクション番号にて管理しています。たとえば、同一行を複数のユーザーで更新した場合、つまりあるテーブルの1つの行が複数のユーザーによって更新されているとします（**図3**）。トランザクション番号144のトランザクションがコミットされたあと、ユーザーAにより同一行が更新され、145というトランザクション番号が振られたとします。このとき、同時にユーザーBも同一行に対してトランザクション処理を実行し、146というトランザクション番号が振られたとします。ユーザーAのトランザクションがコミットされると、トランザクションは問題なく実行されます。しかし、ユーザーBのトランザクションはトランザクション番号144の実行後の履歴をベースにしていますので、ユーザーBのトランザクションを実行してしまうと、ユーザーAのトランザクションとの矛盾が生じてしまうため、InterBaseはユーザーBのトランザクションをエラーとして処理します。このとき、InterBaseは「the proposed update conflicts with the current state of the row.」というエラーを返します。

ユーザーBのトランザクションですが、エラーになる時期はWAITモードかNO WAITモードかで変わります。WAITモード時はユーザーAのトランザクションが終了するまでトランザクション処理を待っていますので、ユーザーAのトランザクション終了後に、ユーザーBのトランザクションが実際に実行されたときにエラーになります。また、もしユーザーAのトランザクションがロールバックした場合、問題なくユーザーBのトランザクションが実行されます。NO WAITモード時はユーザーBがトランザクションを開始しようとロックした時に、ロックの衝突が発生しますので、ただちにエラーとなります。

InterBase 6.0 for Linux

InterBaseのLinux版は、InterBase 4.0 for Linuxのフリーダウンロードから始まっています。このビルドは、当時のInterBase UNIX版の日本における総代理店であった、リオスコーポレーションによって作成され、InterBase(当時米国インプライズの100%子会社)によって検証されたものです。アスキーから販売されている「データベースLinux」を読まれた方も多いのではないかと思いますが、こちらの本を書かれた著者の方は実際にポーティングに携わった方であるとお聞きしています。InterBase 4.0の次のバージョンであるInterBase 5.0 for Linuxからは有料の製品となり、glibc 2サポートとなりました。InterBaseは非常にコンパクトな設計であるため、Linuxとの親和性は非常に良く、同じマシンで動作させた場合、Windows NTよりも高いパフォーマンスを得られるなどの特徴がありました。また、他のオープンソースの製品と異なり、長い間有料の製品として販売されてきたものですので、こういった製品がオープンソース化され、自由に使用できる意味は非常に大きいと思います。InterBaseは筆者の最も好きなデータベースですので、オープンソース化によってより一層浸透していくのを期待しています。

さて、InterBase 6.0のLinux版には2つの種類が用意されています。Classic版とSuperServer版です。この違いは後ほど説明しますが、動作環境は次のようになっています。

カーネル： 2.2.12以降
ライブラリ： glibc 2.1以降
CPU： Pentium II以降を推奨
メモリ： 64Mバイト以上を推奨

▼リスト3　トランザクションの使用例

```
SET TRANSACTION
      READ WRITE
      ISOLATION LEVEL SNAPSHOT TABLE STABILITY
      RESERVING
      Stock,StockToBeOrdered for PROTECTED WRITE
```

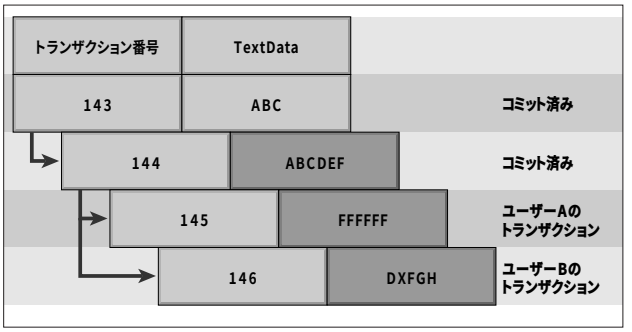


図3　同時更新の例

カーネルが2.2.12以降、ライブラリがglibc 2.1以降なので、Red Hat Linux 6.2J、Vine Linux 2.0、LASER 5 Linux 6.2、TurboLinux Workstation 6.0などで問題なく稼働します。

CPUはPentium II以降を推奨していますが、それ以前のマシンで動作させてもまったくストレスを感じさせません（筆者はPentium 133MHzで使用しています）。RDBMSですので、どの程度のパフォーマンスを要求するかによってCPUやメモリは異なってくるでしょう。

■InterBase 6.0 for Linuxのインストール

Linux版にはClassic版とSuperServer版の2種類が用意されています。

Classicアーキテクチャ版
InterBaseCS_LI-V6.0-1.i386.rpm
SuperServerアーキテクチャ版
InterBaseSS_LI-V6.0-1.i386.rpm

このほかにtar.gz形式も用意されていますので、Slackware系の方も使用できるようになっています。本誌付属CD-ROMには、RPM形式で2種類を収録しています。インストールは単純にrpmをインストールするだけです。GNOMEなどのデスクトップを使用されている方はGUIからインストールを行うこともできます（**画面1**）。

まず、ユーパーユーザーモードにします。

```
$ su
Password:
```

画面1　GNOMEのファイルマネージャからのインストール

次に、RPMパッケージのインストールを行います。

```
# rpm -ivh InterBaseCS_LI-V6.0-1.i386.rpm
```

InterBaseのインストールを行うと/opt/interbaseに製品がインストールされます。それでは正しく起動できるかどうか確認してみましょう。InterBaseにはisqlというコマンドラインツールが付いています。このコマンドラインツールは、SQL文の実行（バッチ的な実行も含む）が行えるツールです。このツールを使ってサンプルのデータベースにアクセスしてみましょう。

まず、isqlがあるディレクトリに移動します。

```
# cd /opt/interbase/bin
```

次に、isqlを実行します。

```
# ./isql -User SYSDBA -Password masterkey
```

-Userはユーザー名、-Passwordはパスワードです（ここでは、デフォルトで設定されているユーザー名とパスワードを使用）。

次に、サンプルデータベースに接続します。

```
SQL> connect '/opt/interbase/examples/employee.gdb';
```

データベースに接続できたところで、データベースに格納されているテーブル一覧を表示してみましょう。

```
SQL> show tables;
```

次に、データベースのプロパティを表示してみます。

```
SQL> show database;
```

最後に、SELECT構文でEMPLOYEEテーブルを表示してみます。

```
SQL> select * from employee;
```

isqlを終了するには、「quit;」と入力します。

```
SQL> quit;
```

画面2が、実際にデータベースに接続している画面です。

現在のところLinux版にはコマンドラインツールしかありませんが、Kylix（インプライズが開発中のC、C++、Delphiをサポートする開発ツールのコードネーム）が登場すればWindows版に付属しているGUIツールのほとんどが移植されるでしょう（**画面3**）。

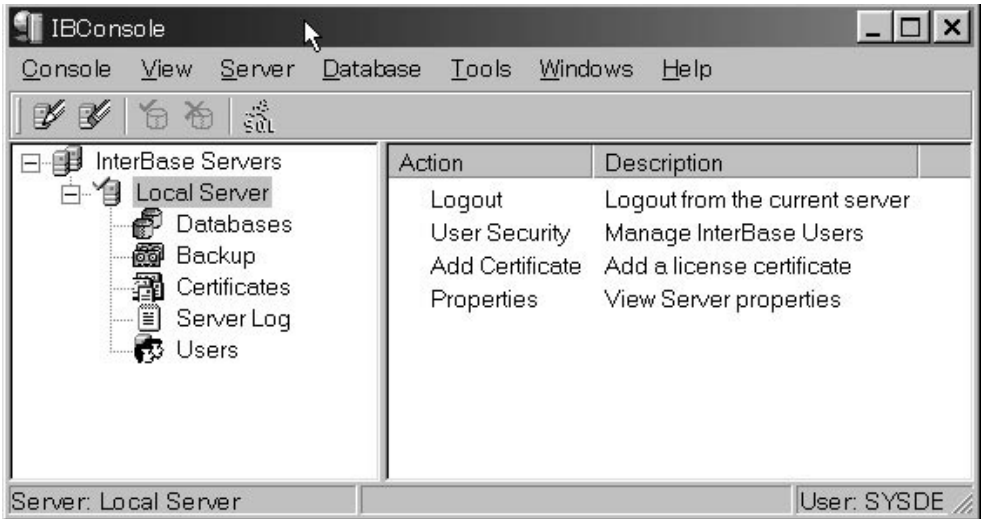
SuperServerとClassicの違い

InterBaseではバージョン4.0の時代（Netware版を除く）までの設計をClassicアーキテクチャと呼んでいます。Classicアーキテクチャはシングルスレッドアーキテクチャで、1台のクライアントがサーバに接続されると1つのプロセスを作成して管理しています。接続数分だけプロセスが作成され、ロックマネージャがトランザクションの衝突を防ぐように管理しています。この方法を使用した場合、接続数が多くなるとパフォーマンスが悪くなるだけでなく、ロックオブジェクト（ロック情報を管理しているファイル）が大きくなります。ロックオブジェクトが増えると、さらにパフォーマンスが悪くなります。ただし、安定して稼働させるためにはこの方法が優れていると思われます。

一方SuperServerアーキテクチャは、マルチスレッドを利用したアーキテクチャです。クライアントがサーバに接続すると、新規のスレッドを作成して管理します。複数のプロセスを使用せずに1つのプロセスで管理できるため、Classicアーキテクチャよりもパフォーマンスが優れています。

```
# cd /opt/interbase/bin
# ./isql -User SYSDBA -Password masterkey
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect '/opt/interbase/examples/employee.gdb';
Database:  '/opt/interbase/examples/employee.gdb', User: SYSDBA
SQL> show tables;
       COUNTRY                         CUSTOMER
       DEPARTMENT                      EMPLOYEE
       EMPLOYEE_PROJECT                JOB
       PHONE_LIST                      PROJECT
       PROJ_DEPT_BUDGET                SALARY_HISTORY
       SALES
SQL> show database;
Database: /opt/interbase/examples/employee.gdb
        Owner: SYSDBA
PAGE_SIZE 4096
Number of DB pages allocated = 247
Sweep interval = 20000
SQL> select * from employee;

 EMP_NO FIRST_NAME      LAST_NAME           PHONE_EXT   HIRE_DATE DEPT_NO JOB_C
ODE JOB_GRADE JOB_COUNTRY                     SALARY FULL_NAME

======= ============== ==================== ========= =========== ======= =====
=== ========= ============== ====================== ========================
===========

      2 Robert          Nelson              250       28-DEC-1988 600     VP
            2 USA                           105900.00 Nelson, Robert

      4 Bruce           Young               233       28-DEC-1988 621     Eng
            2 USA                           97500.00 Young, Bruce

                     (....途中略....)

    145 Mark            Guckenheimer        221        2-MAY-1994 622     Eng
            5 USA                           32000.00 Guckenheimer, Mark


SQL> quit;
```

画面2　データベースに接続



画面3
IBConsole(Windows版のGUIツール)

Linux版は5.0まではClassicアーキテクチャを使用していました。これは、Linuxのスレッド管理がミッションクリティカルな業務で使用するには貧弱であったため、信頼性のあるClassicアーキテクチャを採用したためだと思われます。InterBase 6.0のLinux版では、従来のClassicアーキテクチャと、SuperServerアーキテクチャの両方が用意されています。どちらを使用するかは個人の判断になると思いますが、信頼性を求めるのであればClassicアーキテクチャを使用されることをお勧めします。

InterBaseの接続形態

InterBaseはさまざまな接続方法をサポートしています。インプライズの製品であったため、Delphi、C++Builder、JBuilderなどの開発ツールとは非常に親和性が優れています。また、現在インプライズが開発しているLinux用の開発ツールであるKylixでも、InterBaseへのネイティブ接続がサポートされる予定となっています。これ以外にもまだまだ多くの接続方法があり、ヨーロッパの多くの開発者によってFree Pascal、PHP3、ColdFusionなど、さらに多くのツールのサポートが予定されています。

InterBaseへの接続方法の一例をあげてみましょう。

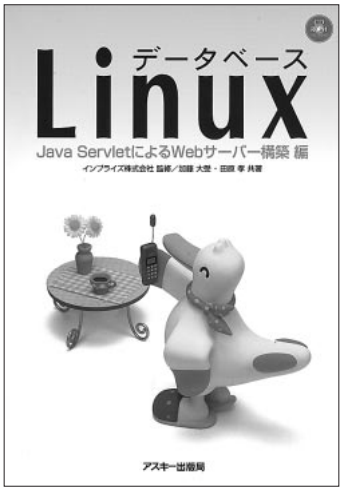
- **API接続(プリプロセッサを使用しての接続)**
- **InterBase Express**
  (Delphi、C++Builder、Kylix(予定))
- **ODBC接続(InterBase 6.0対応版はまだ開発中)**
- **InterClient(InterBase用のJDBCドライバ)**
- **IBPerl(Perl用のドライバ)**
- **BDE/SQL-Link(Delphi、C++Builder)**

次回に向けて

次回はInterBaseの使い方について細かく説明していきます。SQLの話が中心となるでしょう。少し堅苦しい話になりがちですが、RDBMSを使ううえで必要な知識ですので、この機会に覚えていただけると幸いです。

今回の記事は、アスキーより刊行している『データベースLinux Java ServletによるWebサーバ構築編』からの抜粋に加筆したものです。

『データベースLinux Java ServletによるWebサーバ構築編』



インプライズ株式会社監修
加藤大受・田原　孝著
アスキー刊
ISBN4-7561-3379-7
3800円+税

Javaプログラミング入門
Javaで3Dゲームを作ろう

今回は、クラスの概念をもう少し詳しく解説しつつ、前回のball3.javaの解説から始めましょう。ボールを動かしたり、壁などに当たって反射させたりする仕組みや、影の付け方などを解説していきます。さらに、マウス操作で動くラケットを追加します。

Programming

第4回　ラケットとマウス操作

文：おもてじゅんいち／かざぐるま
Text : Junichi Omote／Kazaguruma

前回、クラスの構成を説明するために掲載した**図1**（前回の**図5**）について、賢明な読者の方は不正確ではないかと思われたかもしれません。つまり、Ballクラス、Backgroundクラスは、アプレットクラスの内部に含まれるのではないかと思われた方もいらっしゃると思います。実はこの図は見る観点によっては正しくもあり、間違ってもいるのです

前回は、また初心者の読者のことも考慮して、「クラスの継承」ということの説明で、できるだけプログラムリストから類推しやすいようにしたため、**図1**のようにしています。

細かい解説をすれば、この**図1**はプログラムリストの4つのクラスについての記述、

```
class ..... {
}
```

の構造にできるだけ合わせて、そのうえで「クラスの継承」ということをイメージ化したものです。しかし、実はこのプログラムがアプレットとして実行するときには、Ballクラスや Backgroundクラスはアプレットクラスに含まれるといえるのです。

クラスとインスタンス

クラスとは、プログラムを構成する部品であると簡単に言ってきました。しかし、もう少し厳密に言うと、プログラムの実行時に実行されるのはクラスではありません。

クラスとは実行時にどのように動くかといったことをこと細かに記述した、いわば手順書のことで、そのクラスの記述に従って、メモリ上にプログラムが作成されます。これを「インスタンス」または「インスタンス化されたオブジェクト」と呼びます。「実体（オブジェクト）」という日本語がぴったり合うでしょう。

つまり、プログラミング時にはクラスとして書いたものが、実行時にそれぞれインスタンスとして実体化されて動きます。

Javaのようなオブジェクト指向言語以前の旧式（？）のプログラミング言語では、プログラムコードとデータが

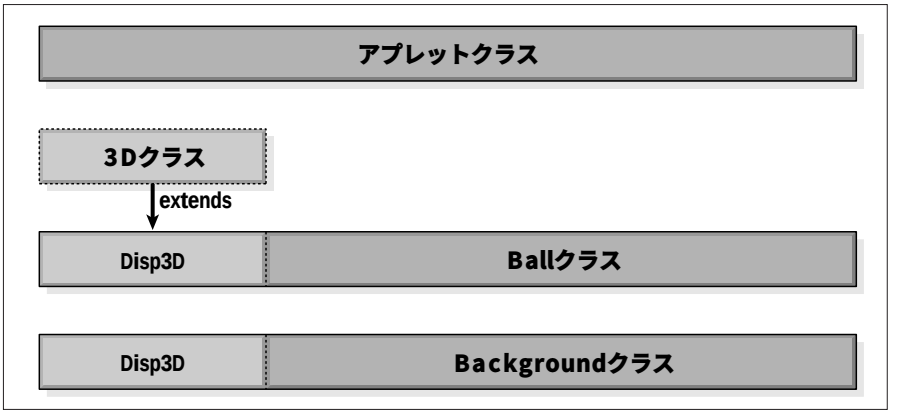


図1　クラス構成（ball3d.java）

明確に分離されていて、データはいくつでも複製したり削除したりすることができましたが、プログラムコード自体は起動時から終了時まで固定して存在していました。

オブジェクト指向言語では、そのようなプログラムとデータの区別自体がなくなって、プログラム部分でさえ、クラスとして記述しておけば、必要なときに必要なものだけインスタンス化して使うことができます。

では、各クラスのインスタンス化はいつ行われるのでしょうか。

実は、Javaアプレットではアプレットクラスだけが起動時にシステムによってインスタンス化されます。ですから、そのほかのクラスはこちらでインスタンス化してやらなければなりません。それが、アプレットクラスのinit()メソッド内の、

```
ball = new Ball(400,400);
bg = new Background(400,400);
```

という部分です（**リスト1**）。

newというキーワードがまさに「クラスをインスタンス化」するためのもので、「新しいメモリ領域を割り当てる」という意味からnewという名前になったのでしょう。

これでアプレットクラスに続き、BallクラスとBackgroundクラスもインスタンス化されたわけですが、あと1つ残った3D処理用のDisp3Dクラスのインスタンス化はどうするのかというと、これはスーパークラスとしてBallクラスとBackgroundクラスに継承されているため、自動的にそれぞれの一部として一緒にインスタンス化されることになります。

こうしてインスタンス化された実体（オブジェクト）の構成は、**図2**のようになり、**図1**のクラス構成とは少し異なったものになるのです。

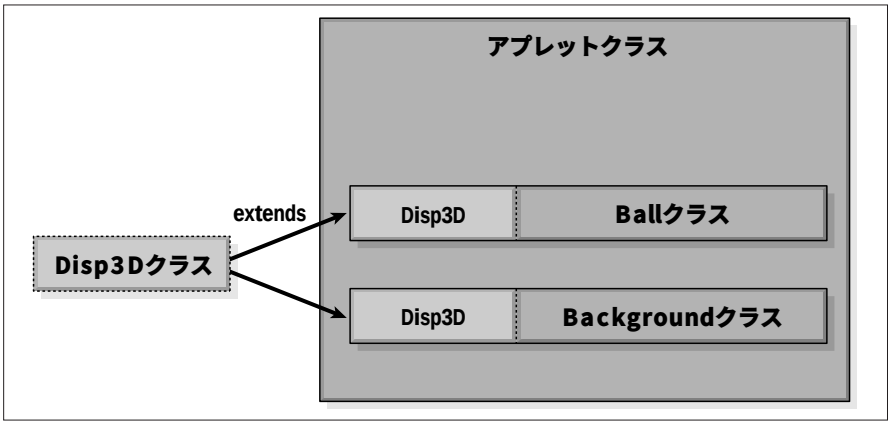


図2　オブジェクト構成（ball3d.jaba）

▼リスト1　クラスのインスタンス化（ball3d.java抜粋）

```
//---------------------------
// Applet Class
//---------------------------
public class ball3d extends Applet implements Runnable {

        Thread          t;
        Ball            ball;
        Background      bg;
        Image           buffer;
        Graphics        bufferg;

        public void init() {
                buffer = createImage(400,400);

                ball = new Ball(400,400);          //インスタンス化
                bg = new Background(400,400);      //インスタンス化

                t = new Thread(this);
                t.start();
        }

        public void run() {
                try {
                        while(true) {
                                ball.process();            // ボールを動かす
                                repaint();
                                Thread.sleep(40);
                        }
                }
                catch(Exception e) {}
        }

        public void update(Graphics g) {
                paint(g);
        }


        public void paint(Graphics g) {
                if(bufferg == null)       bufferg = buffer.getGraphics();

                bg.disp(bufferg);                          // 背景描画
                ball.disp(bufferg);                        // ボール描画

                g.drawImage(buffer, 0, 0, this);
        }
}
```

クラス間の連携

ついでに復習も兼ねて、アプレットクラス内でインスタンス化されたBallクラスとBackgroundクラスが、どのようにしてアプレット自体と連携を取って動いているかを見てみましょう。

アプレットクラスのrun()メソッド内の、

```
ball.process();
```

で、ボールが進みます。壁などに当たればそれなりに反射してくれます。ボールの速度は一定にしたいので、このメソッドを呼ぶのはrun()メソッドのwhileループ内である必要があります。ここなら、

```
Thread.sleep(40);
```

のおかげで、一定の時間間隔でくり返されますので、ボールの速度も一定になってくれるのです。

一方、アプレットクラスのpaint()メソッド内で、BallクラスとBackgroundクラスの描画メソッドを呼び出しています。

このときに、描画先が必要となります。ここではダブルバッファリングを採用しているため、実画面ではなく、バッファ領域buffergをパラメータとして渡しています。

```
bg.disp(bufferg);
ball.disp(bufferg);
```

気をつけなければならないのは、描画は向こう側（画面の奥）にあるものから順に行わなければならないということです。実行すればわかることですが、壁を描画するbg.disp()メソッドを先に呼び出さなければ、ボールが壁に隠れてしまいます。

また、この2つの描画メソッドは、ball.process()のようにアプレット側のタイミングで行うのではなく、システムが「描画しなさい」と言ってくるタイミングで行うために、paint()メソッド内に記述しています。これによって描画がスムーズに行われます。逆にball.process()がpaint()メソッド内にあると、どんなタイミングで呼び出されるかわからないために、ボールの速度が一定ではなくなってしまいます。

このように、どのように動くか、どのように描画するかなどの詳細は各クラスの内部に任せて、アプレットクラスからは、そのメソッドを実行するタイミングや、描画先がどこであるかだけを制御するようにします。

ボールを動かすためには

このゲームの心臓部はなんといっても、3D空間で動くボールでしょう。反射を含むこのボールの動きのプログラミングは、いろいろなシミュレーションやアクションゲームなどに応用する

▼リスト2　Ballクラス（ball3d.java抜粋）

```
//--------------------------
// Ball Class
//--------------------------
class Ball extends Disp3D {
        int             vx,vy,vz;
        int             x,y,z;                  // 3D座標
        int             gx,gy,gr;               // 2D座標、直径
        int             sgx,sgy,sgr;            // 影2D座標

        Ball(int w, int h) {
                super(w,h);
                init();
        }

        void init() {
                x = 0;
                y = 0;
                z = 0;

                vx = 13;
                vy = 15;
                vz = 15;

                gr = 40;
                process();
        }

        void process() {
                x += vx;
                y += vy;
                z += vz;

                int lim = 500 - gr / 2;

                if ( x > lim )          vx = -vx;
                if ( x < -lim )         vx = -vx;
                if ( y > lim )          vy = -vy;
                if ( y < -lim )         vy = -vy;

                if ( z > lim )          vz = -vz;       // ブロック面
                if ( z < 150 )   {      vz = -vz;       z = 150; }
                                                        // 手前の面

                gx = d2x(x,y,z);
                gy = d2y(x,y,z);
                gr = d2r(100,z);

                sgx = gx;
                sgy = d2y(x,500,z);
                sgr = gr / 2;
        }

        public void disp(Graphics g) {
                g.setColor(Color.black);
                g.fillOval(sgx-gr/2, sgy-sgr/2, gr, sgr);

                g.setColor(Color.red);
                g.fillOval(gx-gr/2, gy-gr/2, gr, gr);
        }
}
```

ことができますので、ぜひその理屈をマスターしてください。

プログラミングでの「動き」とは、ようするに座標の変化であるといえます。2Dであれば(X,Y)、3Dであれば(x,y,z)の各数値を次々と変化させていけば、ボールは飛んで行きます。そして、この座標を変化させるのは速度です。速度というのは一定時間（たとえば1秒間）で進む距離のことですから、コンピュータ処理的には現在の各座標に、速度の値を足してやるだけでよいことになります。その足し算が一定時間ごとに行われれば、等速運動ということになるのです。

現実には加速度や摩擦により、速度自体が刻々と変化するのですが、このゲームでは重力や空気抵抗を考えないので速度は一定です。

小学生のときに算数で学んだような「速度」だと、方向によらず1つの数値で表していたのですが、このゲームは3Dですから(x,y,z)の各数値の変化が必要で、速度もx,y,z軸それぞれ用意しなければなりません。ここでは各方向の速度成分を（vx,vy,vz）として管理することにします。つまり、x方向の速度vx、y方向の速度vy、z方向の速度vzとそれぞれを別々に考えます。

Ballクラスを抜粋した**リスト2**を見てください。init()メソッドで、ボールの位置を表す座標x、y、zと各方向を表す速度vx、vy、vzの初期設定を行っています。grはボールの直径です。

process()メソッドの最初の3行でボールの座標を変化させています。

```
x += vx;
y += vy;
z += vz;
```

前述したように、単に速度の値を各座標に足し算しているだけということがわかると思います。

壁にぶつかったら……

座標に速度を足していくだけでは、ボールは直線運動をするだけで、すぐに画面から飛び出してしまいます。そこで、壁などに衝突するときに反射させたいのですが、ここでも速度を各方向に分けたことが大いに役立ってくれるのです。

図3は、反射のときの各方向の速度の変化を表したものです。この図は2次元ですが、3次元でも原理は同じです。

Aは水平の壁での反射ですが、反射前と反射後のx、y方向の速度がそれぞれどうなっているかをよく見てください。**図3**ではvxは変わらず、vyは方向だけが反対になっています。Bの場合では逆にvyが変わらず、vxの方向が逆になっています。

つまり、反射とはあるひとつの方向の速度だけが逆になり、他のすべての速度成分の値は変わらないということなのです。

図3のAの場合、y方向に進んでいるときの障害物での反射であれば、

```
vy = -vy;
```

また、Bのx方向に進んでいるときの障害物での反射であれば、

```
vx = -vx;
```

とするだけで、立派に「反射」が行われるのです。

現実の世界では、反射に伴う速度の減衰や、反射面での摩擦などが複雑にからみあうのですが、ここではそれらを全部無視すると割り切っているので、これらの式だけでOKです。

3Dへの拡張も同じように考えれば、z方向に進んでいるときの障害物での反射は、

```
vz = -vz;
```

となります。

では、Ballクラスの反射を処理する部分を見てみましょう（**リスト2**）。process()メソッド内の5行目からが反射を処理しています。

たとえば、

```
if ( x > lim )    vx = -vx;
```

は、「もしボールのx座標がlimを超えたら、速度vxを反転させる」という意

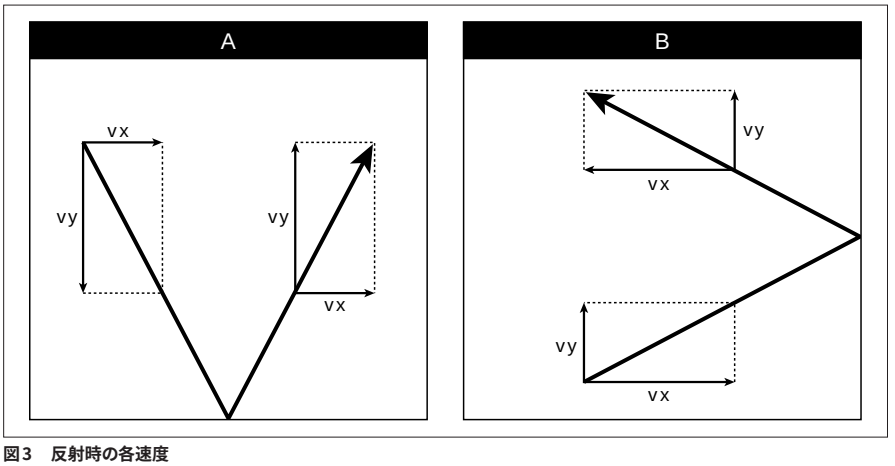


図3　反射時の各速度

味になります。limは中心から壁までの距離に設定されています。さらに左右上下の壁についても、同様の反射処理を行っています。

次の、

```
if ( z > lim )    vz = -vz;
```

は、画面奥の面との衝突を表します。このプログラムはブロックくずしゲームなので、いずれここにブロックを消す処理が入ります。

その下の、

```
if ( z < 150 ) ……
```

は、手前の面、つまりラケットが動く面で、ラケットでの反射処理をすることになります。

中心から左右上下の壁までの距離が500なので、手前の面から奥の面までの距離も500でよさそうなのですが、実際に描画してみると、それでは少し深くなりすぎて奥の面が小さくなってしまうため、奥行きを浅くしてあります。そのために、zが0でなく150の地点が手前の面にしています。

process()メソッドの残りの部分は、最初の部分で速度を足し算することと、各方向の反射によって計算されたボールの座標を、画面表示用に2D座標に変換しています。

gx、gy、grは画面上のボールの表示座標で、3D→2Dの変換のためにDisp3Dクラスのメソッドを使っています。BallクラスはDisp3Dクラスを継承しているので、自分自身のメソッドのように使っているわけです。

sgx、sgy、sgrはボールの影の座標を計算しています。単純なことですが、この影の存在がより3Dらしさを出しているといえます。影は性質上、床面にしか現れないものですから描画はじつに簡単で、ボール本体と同じ座標でy座標のみを床面の座標にすればよいのです。つまり、sgyのみ、

```
sgy = d2y(x,500,z);
```

というように、y=500（床面のy座標）と固定して計算するだけです。さらに、遠近感からくる影の直径の変化も、ボール本体と同じです。

```
sgr = gr / 2;
```

実は擬似的な処理ですが、影を楕円にするために縦の直径を横の直径の半分にしています。

難解な物理の公式も……

どうでしょう、単純な足し算と引き算くらいしか使っていませんが、これがボールの動きをつかさどる、このゲームの心臓部であることには違いありません。

これまで解説してきたことは入門編ゆえ、重力や空気抵抗を無視した比較的簡単な動きでしたが、中学校の物理で習った考え方を利用するだけで、たとえば重力の効果を加えたりできます。

「加速度」という言葉を思い出してください。一定時間ごとに（process()メソッド内で）、y方向の速度vyを減らして行くと、vyに加速度を加えるということになます。重力も下向きの加速度の一種ですから、速度を刻々と変えていけば画面上で重力を表現できるわけです。

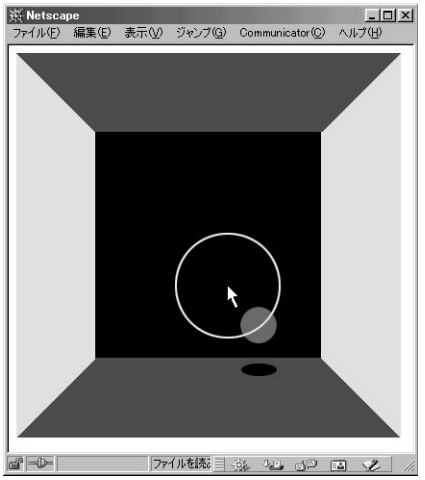
そういえば物理の授業で、加速度とか重力とかを習ったことを思い出しませんでしたか。そのときに出てきた公式はとりあえず暗記するだけで、式のもつ意味などを考える機会はあまりないものです。しかし、本ゲームのBallクラスで行っていることは、まさにその加速度や速度を表しているのです。物理で習った公式と意味は同じなのです。かつて習ったあの複雑な公式はわれわれ人間用のもので、コンピュータに処理をさせるときはこのようにけっこう単純だったりするのです。

物理の公式を利用などといっても最初からあまり構えないで、公式よりむしろ物理で習った「考え方」を利用するようにしましょう。

ラケットを動かす

次に、もうひとつの動く部品であるラケットを登場させましょう（**画面1**）。今回は描画が複雑にならないように、ちょっと手抜きして簡単な円を描画するだけなのですが、動きは当然マウスに従わなければなりません。

ラケットもまた独立した部品なので、Racketクラスを作ります。**リスト3-1**の網掛け部分が、前月のball3d.javaからの変更点で、このプログラムの最後にRacketクラス（**リスト3-2**）を追加します。前月から変更・追加したのは、主にRacketクラスのインスタンス化



画面1　ラケットとボール

と、マウス処理についての部分です。Disp3D、Ball、Backgroundクラスには何も変更はありません。

ボールと違って、ラケットは人間が動かすわけですから、やはり動きのための計算と描画は別々に行います。描画は、他のクラスと同じようにアプレットクラスのpaint()メソッド内で行い、動きの計算は、アプレットクラスのmouseMoved()メソッド内からRacketクラスのset()メソッドを呼び出して行います。

▼リスト3-1　ball3d_racket.java（変更点）

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

//---------------------------
// Applet Class
//---------------------------
public class ball3d_racket extends Applet
          implements Runnable,MouseMotionListener  {

          Thread              t;
          Ball                ball;
          Background          bg;
          Racket              rc;
          Image               buffer;
          Graphics  bufferg;

          public void init() {
                    addMouseMotionListener(this);

                    buffer = createImage(400,400);

                    rc = new Racket(400,400);
                    ball = new Ball(400,400);
                    bg = new Background(400,400);

                    t = new Thread(this);
                    t.start();
          }

          ...
          （省略）
          ...


          public void paint(Graphics g) {
                    if(bufferg == null)bufferg = buffer.getGraphics();

                    bg.disp(bufferg);           // 背景描画
                    ball.disp(bufferg);         // ボール描画
                    rc.disp(bufferg);           // ラケット描画

                    g.drawImage(buffer, 0, 0, this);
          }

          //---- Mouse Event
          public void mouseMoved(MouseEvent me) {
                    int       x = me.getX();
                    int       y = me.getY();
                    rc.set(x,y);
          }

          public void mouseDragged(MouseEvent me) {
          }
}

//---------------------------
// 3D Class
//---------------------------
...（省略）...

//---------------------------
// Ball Class
//---------------------------
...（省略）...

//---------------------------
// Background Class
//---------------------------
...（省略）...
```

アプレットでのマウス処理

アプレットクラスでマウスの移動を検出するには、MouseMotionLitenserという基本クラスを使うため、冒頭に、

```
import java.awt.event.*;
```

という記述と、アプレットクラス宣言のときに、

```
inplements MouseMotionListener
```

という記述、さらにinit()メソッド内に、

```
addMouseMotionListener(this);
```

という記述が必要になります。この3つでマウス処理を行う準備はできましたが、もうひとつ、このアプレットではマウスをラケットとして扱うので、Racketクラスをインスタンス化しておきます。

```
rc = new Racket(400,400);
```

さて、マウスの操作でラケットを動かすためには、マウスが動いたかどうかを検出すればよいので、そのためにmouseMoved() メソッドを用意します。これはマウスが動いたときに自動的に呼び出されるメソッドです。

meというパラメータが、マウスの座標等を持っているので、

```
x = me.getX();
y = me.getY();
```

というように、getX、getYというメソッドによってx座標、y座標を取り出すことができます。そして、得られたマウスの座標を、set()メソッドによってRacketクラスに渡します。

```
rc.set(x,y);
```

ここではラケットの描画は行わず、座標位置の指定だけを行います。

なお、そのあとにマウスのドラッグ処理を行う、mouseDragged()というメソッドがあります。今回はマウスのドラッグの処理は必要ないのですが、MouseMotionListenerクラスでは、mouseMoved()と、mouseDragged()の2つのメソッドは必ず用意しなければならないという決まりなので、空のメソッドとして形だけ用意しておきます（用意されていないとコンパイルエラーになる）。

Racketクラス

Racketクラスの内部は比較的単純で、set()メソッドによってマウスの動きから座標を取得して記憶しておき、disp()メソッドでラケットを描画しているだけです。

ラケットの絵がリングだけなので、せめて少しは立体的に見せるためにちょっと凝った描画をしていますが、基本的にはいくつかの円を色を変えて描画しているだけです。

さて、このball3d_racketアプレットを実行してみると、3D空間で飛び跳ねるボールと、マウスに追随して自由に動かせるラケットが登場します。やっとゲームらしくなりましたね。

とはいえ、まだボールがラケットに当たったか、空振りしたのかを判定するようにはなっていませんから、ラケットの位置にはおかまいなしにボールは反射しつづけます。

当たり判定

こうなればやっぱり、ラケットにボールを反射させねばなりません。そのためには、ボールが手前の面にきたときに、うまくラケットに当たったか、ラケットのない部分に当たったか（つまり空振り）を判定する必要があります。当然、後者の場合は反射せずにそのままボールが消えていくのが自然でしょう。

プログラム的に言い換えれば、ボールが手前の面の位置に来たときに、ボールの(x,y)座標が、ラケットの円内に含まれているか、円の外に含まれているかということになります。つまりラケットの中心の座標と、ボールの座標との距離が、ラケットの半径よりも小さければ反射、大きければ空振りということになります。

判定のタイミングは、ボールが手前の面に来たときですから、**リスト4**のBallクラス、process()メソッドの、

```
if ( z < 150 )  attack();
```

という記述により、とにかくボールが手前の面に来たときにattack()メソッドが呼び出されます。ですから、このattack()メソッド内で当たり判定を行えばよいのです。

さて、このattack()メソッドをどう作ればよいかを次回までの宿題としたいのですが、実際に「ボールの座標と、ラケットの座標の距離が……」と考え

▼リスト3-2　ball3d_racket.java（追加分）

```
//---------------------------
// Racket Class
//---------------------------
class Racket extends Disp3D {
          int                x,y,r;
          int                gx,gy,gr;

          Racket(int w, int h) {
                    super(w,h);

                    x = 0;
                    y = 0;
                    gx = d2x(x,y,0);
                    gy = d2y(x,y,0);
                    r = 200;
                    gr = d2r(r,0);
          }

          public void set(int x, int y) {                    // 位置設定
                    gx = x;
                    gy = y;
          }


          public void disp(Graphics g) {                     // 表示
                    int       rr = gr;

                    g.setColor(Color.darkGray);
                    g.drawOval(gx-rr/2, gy-rr/2, rr, rr);
                    rr-=2;
                    g.drawOval(gx-rr/2, gy-rr/2, rr, rr);
                    rr-=2;
                    g.setColor(Color.yellow);
                    g.drawOval(gx-rr/2, gy-rr/2, rr, rr);

                    rr-=2;
                    g.setColor(Color.white);
                    g.setXORMode(Color.white);
                    g.fillOval(gx-rr/2, gy-rr/2, rr, rr);
                    g.setPaintMode();
          }
}
```

ると、「Ballクラス内部（attack()メソッド）で、どうやってラケットの座標を知ることができるのだろうか」という大きな問題にぶつかってしまいます。

このままではあまりに難しすぎるので、**リスト4**に、そのヒントを載せておきます。

BallクラスとRacketクラスは、それぞれアプレット内に同列に存在しますから、アプレットからそれぞれのメソッドは呼び出せますが、BallクラスからRacketクラスのメソッドを呼び出すことができません。そのためには、Ballクラス内でRacketクラスのインスタンスがどれであるかを知る必要があります。

リスト4の網掛け部分がそのためのからくりです。つまり、Ballクラスはアプレットクラスから、Racketクラスのインスタンスをパラメータとして受け取っています。アプレットクラスinit()メソッドの、

```
ball = new Ball(400,400,rc);
```

の3つめのパラメータが、Racketクラスのインスタンスで、これをBallクラスではBall()というコンストラクタ（クラスがインスタンス化される時に自動的に呼び出されるメソッド）で受け取っています。

Ballクラス内部では、これをrcとして保持していますので、今後いつでもRacketクラスのメソッドを呼び出すことができるというわけです。さて、これでラケットの座標がわかりそうです。

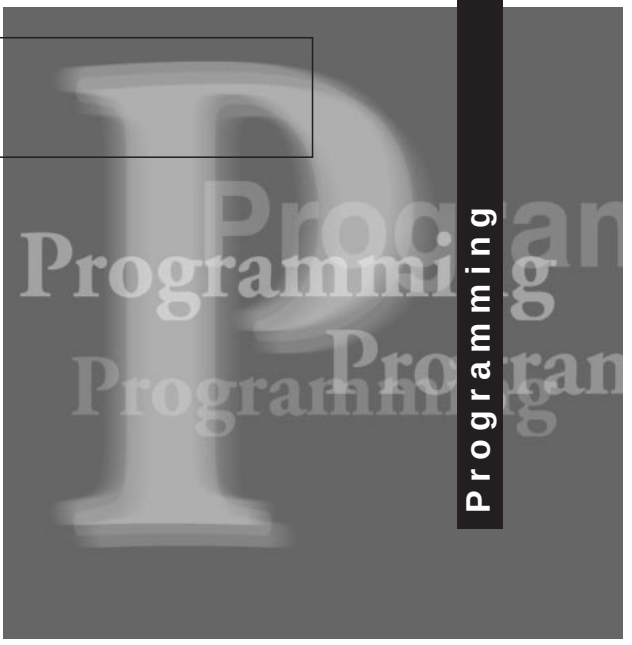
▼リスト4　ボールとラケットのアタリ判定

```
//---------------------------
// Applet Class
//---------------------------
public class ball3d_racket extends Applet
          implements Runnable,MouseMotionListener  {

          …
          （省略）
          …

          public void init() {
                    addMouseMotionListener(this);

                    buffer = createImage(400,400);

                    rc = new Racket(400,400);
                    ball = new Ball(400,400,rc);
                    bg = new Background(400,400);

                    t = new Thread(this);
                    t.start();
          }

          …
          （省略）
          …


//---------------------------
// Ball Class
//---------------------------
class Ball extends Disp3D {
          int               vx,vy,vz;
          int               x,y,z;                    // 3D座標
          int               gx,gy,gr;                 // 2D座標、直径
          int               sgx,sgy,sgr;              // 影2D座標
          Racket    rc;                               // ラケットのインスタンス

          Ball(int w, int h, Racket rc0 ) {
                    super(w,h);
                    rc = rc0;
                    init();
          }
          …
          （省略）
          …

          void process() {
                    x += vx;
                    y += vy;
                    z += vz;

                    int lim = 500 - gr / 2;

                    if ( x > lim )              vx = -vx;
                    if ( x < -lim )             vx = -vx;
                    if ( y > lim )              vy = -vy;
                    if ( y < -lim )             vy = -vy;

                    if ( z > lim )              vz = -vz; // ブロック面
                    if ( z < 150 )              attack(); // 手前の面

                    gx = d2x(x,y,z);
                    gy = d2y(x,y,z);
                    gr = d2r(100,z);

                    sgx = gx;
                    sgy = d2y(x,500,z);
                    sgr = gr / 2;
          }
```

自分で解決したときの喜び

あとは、読者の皆さんにお任せします。これまで学んだことから、attack()メソッドを作り上げ、見事ラケットでボールを打ち返せる「ゲーム」に仕立て上げてください。次回までに、頭の中でうなって考えるのもよし、あれこれと書いては実行し……とただひたすらトライするのもいいでしょう。いずれにせよ、できたときの喜びは格別なものですし、そのときに失敗したり学んだことは、確実に身につくことでしょう。

プログラミング工房



先月号では、I/Oアクセスを行うプログラムを作成し、ユーザープロセスから直接デバイスをアクセスする方法を紹介した。しかし、やはりカーネルから行うのが由緒正しきUNIXのあるべき姿であろう。そこで、今月号からの数回で、カーネルのプログラミングについて解説してみたい。

第10回　カーネルのプログラミング(1)

文：藤沢敏喜
Text : Toshiki Fujisawa

カーネルプログラミング

カーネルレベルのプログラムは、プログラミングのなかでも難しい分野である。特に、Windowsのようにソースコードが公開されていないOSで、デバイスドライバを作成しようとすることは、たいへん難しい。WDM（Windows Driver Model）の仕様など、資料自体は読みきれないほど膨大に存在するのだが、デバイスドライバを作成するために本当に必要な情報はあまりなく、探し出すのに苦労する。そして、実際にプログラムを書こうとすると、OSの内部で何が行われているのかがわからないため、いったん問題に突き当たると、それを解決することがきわめて困難な状況に陥ってしまう。

一方、LinuxやFreeBSDなど、カーネルのソースが公開されているOSでは、参考となるデバイスドライバのソースが多数あり、GPLやBSDライセンスに基づいて自由に使用することができる。

また、開発時に問題が生じても、カーネルのソースコードがすべて公開されているため、カーネル自体にデバッグコードを埋め込んだりと、柔軟なデバッグを行うことができるのである。

UNIXから派生したOSのアプリケーションは、シンプルに設計されていることが多いが、カーネルもまた小さく単純な機能のモジュールの集まりとして設計されている。

もちろんLinuxも、Linus氏が“KISS”、つまり“Keep itSimple, Stupid (or Small)”という概念を最重要視して設計したと述べているように、カーネル自体は可能な限りシンプルな構造になっている。

したがって、カーネルといってもちょっと大きなひとつのプログラムであり、カーネルのプログラミングは、極端に難しいというものではない。

今回作成するデバイスドライバ

今回の記事の目的は、カーネルのプログラミングの解説を行うことであるが、解説を目的とした簡単なデバイスドライバを作るだけではあまりおもしろくない。そこで、現在のLinuxカーネルではサポートされていない機器のデバイスドライバを、実際に開発してみることにする。

現状のLinuxカーネルで十分なサポートが行われていない機器のうち、技術的におもしろいのはやはりUSB機器であろう。Windows 98の普及とともに、非常に多くのUSB機器が発売されたため、LinuxでもUSBマウスやUSBキーボードなど一部のUSB機器はサポートされている。しかしながら、多くのUSBデバイスは最新のLinuxカーネルでもまだ使えるようになっていない。

最近では、デジタルカメラやスキャナを始め、魅力的な機器はそのほとんどがUSB対応になってきた。また、現在のUSBと比較して、速度が48倍になる「USB 2.0」の

規格も定められ、6月15日には日本でもUSB 2.0に関する会議が開催された。会議には多くの開発者が参加していて、USBデバイスドライバの開発は今後ますます重要になっていくと思われる。

USBのデバイスドライバの作成は、USBに関する多くの知識を必要とし、デバイスドライバのなかでも開発が困難な部類のものであるが、読者もぜひ開発に挑戦してみてほしい。

なお、今回作成するドライバは、SCSIからUSBへの変換を行う機器のデバイスドライバであるため、技術的には多少難しい。ページ数の限られた雑誌の記事で、その詳細を説明するには無理がある。そのため、この連載では難しいところは思いきって省き、初心者が読んでもデバイスドライバの開発のようすが理解できるような形で進めていきたい。

「必要」は「開発」の母

SCSIは古くから存在する規格であり、多くの機器で採用されている。筆者の自宅にも外付けのハードディスク、MO、多連装のCD-ROM、テープドライブ、そしてフラットベッドスキャナなど多数のSCSI機器が存在する。このため、自宅のPCのほとんどにはSCSIボードが搭載されている。

しかし、最近あるノートPCを入手してからは、ノートPCに直接フィルムスキャナを接続してスキャンしたいという強い欲求が生じた。もちろんPCMCIAのSCSIカードを使えばよいのではあるが、SCSIケーブルは太くて短いため、フィルムスキャナまで接続するのが物理的に難しい。また、PCMCIA部分に接続する部分のコネクタが壊れやすいという欠点もある。

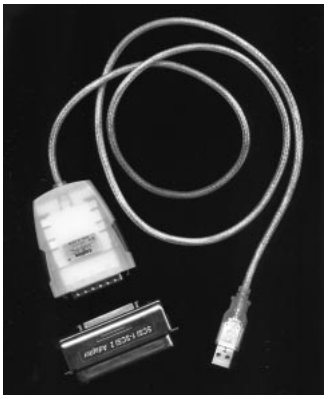


写真1　USB-SCSI変換ケーブル

写真2　USB-SCSI変換ケーブルのパッケージ

そんなとき、PCショップで見つけたのがSCSIのコネクタをUSBに変換するケーブルである（**写真1**）。箱にはスキャナが動作するとは書いてなかったが、7千円程度と安かったこともあって思わず衝動買いをしてしまった（**写真2**）。

家に帰ってWindowsで実験してみると、SCSIでスキャンするよりも若干遅いものの、実用的には十分な速度でスキャンをすることができた。

こうなると、やはりLinuxなどのOSでも使用したくなる。早速カーネルハックに取り組み、現在はLinuxでもとりあえずスキャンが可能な状態にすることができている。

なお、このデバイスドライバの作成はそれなりに時間がかかったが、なんとしてでもこのデバイスを使いたいという強い思いが開発の原動力になった。デバイスドライバの「開発」に必要なのは技術力ではなく、そのデバイスが「必要」だという強い思いなのではないかと思う。技術は開発の過程でなんとかなるものであり、とにかく取り組んでみるという気合いが重要だと感じる。

USBの概要

今までのPCには、PS/2キーボード・マウス、シリアルポート、パラレルプリンタポート、ジョイスティックなど、各デバイスによって物理的な形状も、電気的な特性もまったく異なるコネクタが用意されていた。

このように多数のコネクタがあることは、PCの初心者にとっては敷居が高いだけでなく、PCの裏側がケーブルだらけになって見苦しいという問題があった。また、大きさの制約が厳しいノートPCでは、上記のようなコネクタを配置することは非常に難しい。

このような状況を改善するために、考えられたのがUSB規格であり、次のような特徴を持っている。

- **USBから500mAまでの電源供給が可能**
- **12Mbpsと比較的高速**
- **電源を入れたまま抜き差しが可能**
- **ケーブルが細い（特に低速モード時）**
- **1台のPCに128台までの機器を接続可能**
- **動画やオーディオ用として帯域保証の転送が可能**
- **マウスやHDDなどのプロトコルが公開されている**

このように、使用するユーザーにとっては、今までのインターフェイスと比べ、たいへん使いやすいものとなって

いる。

特に、電源供給が可能なので、USBフロッピーディスクなどのデバイスでは、USB以外のケーブルを必要とせず便利である。また、イーサネットと違って、転送のコントロールはPCがすべて握っている。このため、データを転送する帯域を保証することが可能になり、動画や音声を送っても、画像や音が途切れたりすることがない点も優れている。

しかしながら、USBは上記のようなさまざまな特徴を可能とするため、かなり複雑な仕様になっている。そのため、デバイスドライバを書く立場からすると、既存のシリアルやパラレルを使ったプログラミングと比べると、開発するのにはかなりの知識と時間が必要である。

USBの仕様

■USBの基本仕様

USBの基本仕様は、http://www.usb.org/から入手することができる（**画面1**）。

トップページから［DEVELOPERS］の画面を選択し、さらに［Documents］のページへ行くと、

```
[Universal Serial Bus Revision 1.1 specification]
```

というリンクがあり、ZIP形式で固められた仕様書一式を入手することができる。

このなかにあるUSB-1.1の仕様はPDFファイルで提供され、311ページもある。この英文を全部読むのはかなりたいへんであるが、デバイスドライバのプログラミングに必要なのは、後述の「GetDescripter」など一部分であるので、そんなに恐れることはない。

画面1　USBインプリメンターフォーラム

■USBクラス

USBには、デバイスの用途に応じて「クラス」が定義されている。たとえば、マウスやキーボードなどの人間とのインターフェイスを行うデバイスクラスである「HID（Human Interface Device）クラス」や「オーディオクラス」、そして大容量の記録媒体の仕様を定義した「マスストレージクラス」など、多数のクラスが定義されている。

これらのクラスもまた、先ほどのWebからその仕様書が入手でき、たとえば上記のHID、オーディオ、マスストレージについては、**表1**のようなドキュメントが用意されている。

これらのドキュメントはかなり詳細に書かれており、マスストレージクラスの場合、その概要から通信に使うSCSIコマンドの詳細まで、プログラムを書くために必要なドキュメントはほぼ完全な形で提供されているといってもよいだろう。

USBアナライザ

マウスやキーボード、そして大容量記憶デバイスなど、その仕様が個々の製品によってさほど違わないものは、前述のようにクラス仕様が定義されている。

一方、「USB-SCSI変換装置」のような特殊でマイナーなデバイスは、その仕様がメーカーごとに違っている。つまり、ベンダーが独自に通信プロトコルを決定している。このような場合、その仕様は公開されていないことがほとんどである。

通信プロトコルがわからなければ、デバイスドライバを

HID Class
HID Usage Tables Document 1.1
Human Interface Devices 1.1
HID Point of Sale Usage Tables 1.01
Audio Class
Audio Device Document 1.0
Audio Data Formats 1.0
Audio Terminal Types 1.0
USB MIDI Devices 1.0
Mass Storage Class
Mass Storage Bulk Only 1.0
Mass Storage Control/Bulk/Interrupt(CBI)
Specification 1.0
Mass Storage Overview 1.1
Mass Storage UFI Command Specification 1.0

表1　HID、オーディオ、マスストレージクラスのドキュメント

書くことは不可能であるが、このような場合によく使われる手法としては、通信しているケーブルの信号を直接観測する方法がある。

たとえば、シリアルポートで通信するデジタルカメラなどの場合、シリアルポートが2つあるPCに、FreeBSDのパッケージにある「snooper」というソフトをインストールすることにより、そのPCをRS-232Cアナライザとして使用することができる。これを使えば、画像転送の方法を調査することが可能だ。

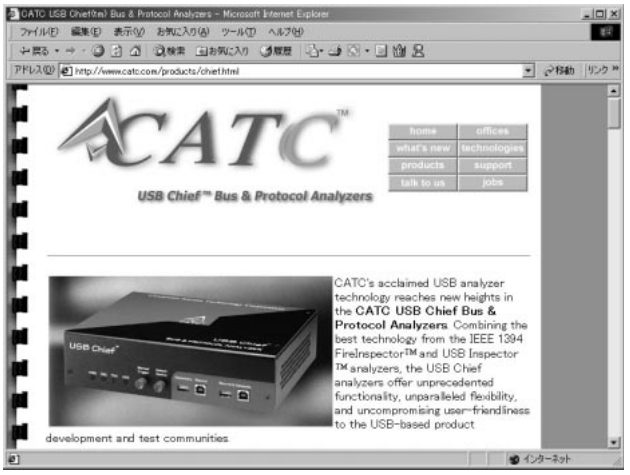
パラレルポートの場合、オシロスコープやロジックアナライザなどを使えば、それなりに信号を観測することができる。

一方、USBはホストとデバイスでは非対象な通信を行うため、snooperのようなソフトの開発は、PCIバスにUSBデバイスとして動作する基板を増設しない限り困難である。また、USBケーブルを流れる信号は、かなり複雑なものとなっているため、オシロスコープやロジックアナライザではまったく歯が立たない。

このため、USBバスの信号の状態を観測するためには、「USBアナライザ」という装置が必要になる。USBアナライザに関しては、CATC（http://www.catc.com/）という会社の製品がデファクトスタンダードになっている。CATCでは、数年前から数種類の製品を提供していて、たとえば**画面2**のような製品がある。

この製品はA5判程度の大きさで、出張時にも簡単に持ち運べるほど小さく軽いものではあるが、趣味で買うには高すぎるのが欠点である。

なお、USBアナライザが手元になくても、USBのデバイスドライバを書くためには、デバイスが動作しているときのUSBアナライザログが得られれば十分役に立つ。もし、USBアナライザを使用できる立場にある友人がいるならば、ドライバを書きたいデバイスを手渡し「このUSBデバイスをWindowsで動作させたログをとってほしい」とお願いするとよいだろう。ログをとるだけであれば数分の作業で可能なので、引き受けてもらえるかもしれない。



画面2　USBアナライザ

USBデバイスをテストするプログラム

上記のようなUSBアナライザがなくても、USBデバイスの動作をそれなりに調べるツールが、http://www.usb.org/で配布されている。このツールは、「USBComp.exe」というファイル名になっている。Windows 98 Second EditionのPCにダウンロードして実行すると、インストーラが起動していくつかのプログラムがインストールされる。

このプログラムは、もともとUSBの仕様を満足するかどうかをチェックするためのツールであるが、USBのコマンドを単独で送ることができたり、USBの構成を調べたりすることができる。

たとえば、このツールに含まれる「USB Check Version 3.2」というプログラムを使うと、デバイスにどんなインターフェイスがあるかや、使用できるエンドポイントの番号を調べたりと、デバイスドライバを作成するうえでかなり役に立つ。

また、「HID View」というプログラムを使うと、USBマウスのボールの動きに合わせてどんなデータが送られて来るかなどを観測することもできる。

ホストコントローラ（UHCI、OHCI）

USBは基本的にはシリアル通信であるので、特定のタイミングで信号をHIGHにしたり、LOWにしたりするだけである。しかしながら、1m秒ごとに同期するための決まったパターンを送出する必要があったり、その1m秒の間に複数の機器宛のデータパケットを効率よく詰め込むなどの複雑な作業が必要になる。

この機能をすべてソフトウェアで実現することはたいへん難しいため、タイミングがクリティカルな部分のUSB通信はすべてハードウェアが行っている。

USBデバイス側では、通信を実現するハードウェアは比較的楽であり、いくつかのFIFOバッファ（通信バッファ）と、セットアップのための通信を理解するハードウェアで構成できる。

しかしながら、USBホスト、つまりPC側は最大127台

のデバイスを相手にしなければならないため、その処理はかなりたいへんだ。そのため、この処理をする部分のハードウェアは必然的に複雑なものとなる。

PC側のデバイスを処理するハードウェアとしては、「UHCI」（Universal Host Controller Interface）と呼ばれるチップと、「OHCI」（Open Host Controller Interface）と呼ばれるチップの2種類がある。

OHCIはMacintoshやPCIバス用のUSB拡張ボードに多く用いられている。一方、UHCIはインテルが開発したもので、数多くのPCで採用されている。

今回、筆者が開発に使ったノートPCもUHCIチップを搭載している。このUHCIの仕様書は、インテルのWebサイト（http://www.intel.com/）で「UHCI」をキーワードとしてサーチすると手に入れることが可能である（**画面3**）。

この仕様書は、ハードウェアのしくみの解説から始まり、コントロールレジスタのアクセス方法までの詳細が記述され、ホスト側のデバイスドライバを理解するには欠かせないものである。

なお、UHCIの部分はUSBアクセスの最も下層の部分であり、linux-2.4.0-test5カーネルではこのあたりのソースコードはかなり完成されたものとなっているようである。したがって、UHCI部分の詳細な理解は、今回の記事で作成するようなデバイスドライバを開発するうえでは、必ずしも必要のないものである。

しかしながら、デバイスドライバをデバッグしていると、どうしても解決不可能なバグが出てくることもある。今回の開発でも、不可解なエラーが返ってくるというバグが出て、それがなかなか解決できなかった。しかし、Linuxカーネルの UHCI部分のソースコードと、UHCIのレジスタアクセス方法のドキュメントを読むことにより、そのバグを解決することができた。

このように、ハードウェアをアクセスするデバイスドライバでは、そのハードの詳細仕様がないと、効率的な開発を行うことは難しい。しかし、USBに関してはホスト側の仕様が公開されているため、Linuxでもかなりスムーズに開発が進んでいるのではないかと思う。

USBでの通信プロトコル

ネットワークで用いられるTCP/IPプロトコルでは、通信する相手との上下関係はなく、対等な条件で通信が行われる。このため、イーサネットでの通信では、各通信ポイントがイーサネット上に同時に出力を行い、データが衝突する場合がある。イーサネットではこの衝突（コリジョン）を検出し、巧妙な方法でこの衝突による弊害を避けるようにしている。

一方、USBではすべてのアクセス制御をホストPC側で行い、デバイスはホストPCが指示するままに動作する。そのため、イーサネットのようにデータが衝突することはなく、帯域が保証された通信を行うことができる。しかしながら、ホストがすべてのコントロールを行うといっても、それなりのプロトコルが必要になる。

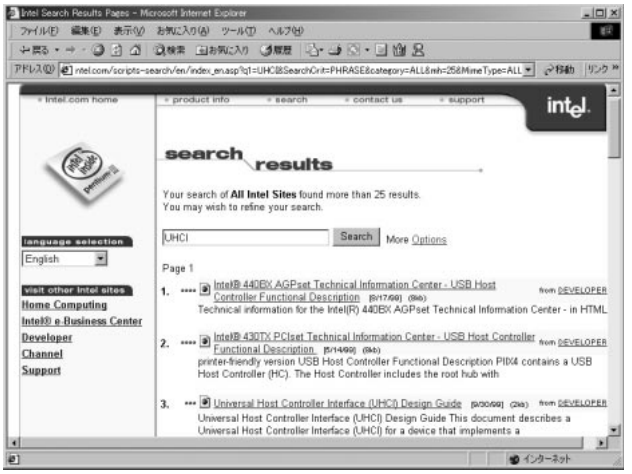
そのため、デバイスドライバを書くうえでは、このプロトコルのだいたいのしくみを知っておく必要がある。概要をざっと説明してみよう。

■Bus Enumeration

TCP/IPの世界では、IPアドレスで各ノードを区別し、DHCPというIPアドレスを自動的に割り当てるしくみがある。

USBでも同様に、各デバイスに1から127までのアドレスが動的に割り当てられる。このしくみは「Bus Enumeration」と呼ばれ、下記の手順で行われる。

1. **デバイスがUSBハブのいずれかのポートに接続される。**
2. **ハブは新規デバイスの接続をホストへ報告する。**
3. **ホストは新規デバイスに一時的にアドレス0を割り当て、GetDescripterコマンドを送る。**
4. **GetDescripterに対するデバイスの応答を確認して、1から127までのアドレスのうち現在使われていないアドレスを新規デバイスに割り当てる。**



画面3　「UHCI」をキーワードにした検索結果

■ Endpoint

イーサネットは1本のケーブルであるが、TCP/IPプロトコルでは、IPアドレスのほかにポート番号と呼ばれる番号で識別される仮想的な通信路を確保できる。たとえば、telnetは23番、POP3は110番というようにポート番号が定義されているのである。

イーサネットと同じように、USBもまた物理的な通信路は1つである。しかし、デバイスによっては、仮想的な通信路をたくさん使いたいことがある。たとえば、デジタルカメラの場合、マスストレージクラスを実現するために、

Column

VMwareとカーネル開発

本文に書いたように、今回のデバイスドライバを作る動機となったのは、ソニーのノートPC VAIO PCG-Z505GR/Kを入手したからである。そのため、今回のデバイスドライバのテストを行ったのもこの機種である。PCG-Z505GR/Kは、販売ランキングの上位にあるので注目している読者も多いのではないかと思う。

開封して、まず真っ先に行う作業は「プリインストールされているWindows 2000の完全消去である」と書きたいところだが、筆者の場合、Windows用のデバイスドライバを作ったりしなければならないこともあるため、Windowsを完全に抹消するわけにもいかない。また、今回のように、USB機器を買ったときの初期動作を確認したりするのにも、Windowsは必要である。

さて、PCG-Z505GR/Kの場合、工場出荷状態ではCドライブが8Gバイト（FAT32）、Dドライブが4Gバイトになっているので、DドライブにLinuxをインストールするという方法がある。

しかし、Windowsには2Gバイトもあれば十分なので、オプションの専用CDドライブとリカバリCD-ROMを使ってパーティションを切り直し、Windows 2000を再インストールした。

次に、FreeBSD用に8Gバイトのパーティションを用意し、雑誌の付録に付いていたFreeBSD-3.3（PAO）のCD-ROMから直接ブートしてインストールを行ったが、まったく問題なくインストールを行うことができた。

この機種の場合、USBフロッピーディスクドライブしか使えないため、カーネルが立ち上がったあとは、修復用fixitフロッピーディスクが読めなくなってしまう。したがって、CD-ROMブートでfsckやviを使ってディスクを修復できることは、非常に重要なポイントである。また、ビデオチップが、XFree86ではまだサポートされていない、NeoMagicのNM2380なので、http://www.xig.com/からFreeBSD版Accelerated-Xを購入して使用することにした。

その次に、残りの2GバイトにLinuxをインストールしようと、本誌7月号の付録CD-ROMにあった、

- **Vine Linux 2.0**
- **Kondara MNU/Linux 1.1**
- **Official Red Hat Linux 6.2J**
- **Plamo Linux 2.0**

という4つのディストリビューションをすべて試してみたのだが、どのインストールプログラムもPCG-Z505GR/Kの専用CD-ROMドライブを認識することができなかった。

このノートでLinuxを使うのはあきらめようかとも思ったが、ふとhttp://www.vmware.com/の存在を思い出し、Windows 2000用のデモ版をダウンロードして使ってみた。VMwareのインストールはたいへん簡単であり、バーチャルマシンからはCD-ROMを通常のATAPIとしてアクセスできるため、先ほど用意した2GバイトのパーティションにVMware上でVine Linux 2.0をまったく問題なくインストールすることができた。もちろん、いったんインストールしてしまえば、VMware上でなく直接Linuxを起動することも可能である。

なお、Linux版のAccelerated-Xは購入しなかったため、LinuxでXアプリケーションを使う場合は、VMwareでLinuxを動作させ、Windows上のX-Serverを使用していた。しかし、最近になってWebを検索したところ、http://www.cl.cam.ac.uk/~and1000/vaioF-series.htmlで、NM2380を搭載したPCでXFree86を使う方法が書かれていることを発見し、LinuxでもXFree86を使用できるようになった。

さて、ここからが本題である。Windows 2000上のVMwareで、LinuxやFreeBSDを実際に使ってみると、けっこう快適であることがわかった。筆者の場合、Windows上で開発をしなければならないことも多いのだが、EmacsやCVSなど、UNIX上の道具がないと仕事にならない。

そのためWindowsマシンとFreeBSDマシンをネットワークで結び、Windows上でXサーバを動作させて使っていた。したがって、新幹線の中などでは、クロスケーブルで2台のノートPCを接続して開発を行っていたのであるが、2台のモバイル生活は結構つらい。こんなときVMwareを使えば、1台のマシンだけで済むようになり、たいへん快適なのである。

さらに、今後VMwareでUSBなどのデバイスがサポートされるようになると、今回の記事のようなUSB機器のドライバを開発する際にもブート時間が短くなるなど、いろいろと便利になるはずだ。また、新しいファイルシステム開発のような分野では、ディスククラッシュを気にすることがなく実験ができるなど、カーネル開発の分野でのVMwareの応用はかなり期待できるのではないかと思う。

しかしながら、Windows 2000上でLinuxやFreeBSDを使うのは、GNUやBSDの魂が「札束」に買われてしまったようで、とても嫌な気分ではある。そうはいっても、VMware上では、USBフロッピーディスクやWinModemなど、LinuxやFreeBSDからは直接使えない機器も使うことができるのだ。

そこで、「VMwareというハードウェアのマシン」を買ったと思って我慢することにして、Windows 2000版のVMwareライセンスを購入してしまったのである。

ReadとWriteで2つの通信路を使うが、撮影などのコントロールを行うため、さらに上り下り2つの通信路を必要とする場合などが考えられる。この最終的に通信したいポイントまでを結ぶ通信路のことを、USBでは「エンドポイント」と呼んでいる。

■ GetDescripter

ホストがデバイスと通信するときには、いくつのエンドポイントがあるかや、各エンドポイントの最大転送サイズ（バイト）がどれくらいあるかなどを調べる必要がある。この転送サイズは、USBバスの混み具合によって動的に変化させたいこともあるため、数種類の設定が可能になっているデバイスもある。

このようなデバイスの情報を得るのがGetDescripter命令で、この命令をデバイスに送ることにより、そのデバイスの構成を得ることができる。

ホストPCは、得た情報によって現在の適切な構成を選択し、それを選択することが可能になっている。

次号で作成するデバイスドライバ

新しい魅力的なUSBデバイスが次々と発売され、メモリスティックのようなノートPCの内蔵デバイスまでもが、ノートPC内部のUSBハブ経由で接続されるようになってきた。

このため、LinuxでのUSBデバイスドライバプログラミングは、ますます重要なものとなるはずである。

次号では、冒頭で紹介したSCSI-USB変換装置を使って、SCSIフィルムスキャナを制御するデバイスドライバを作成しながら、USBデバイスドライバの開発方法について解説してみたい。

Column

USBマスストレージクラス

マウスやキーボードなどのポピュラーなUSBデバイスに関しては、USBを通じて通信をするためのプロトコルが定められ、http://www.usb.org/で一般に公開されている。

また、USBフロッピーディスク、USBハードディスク、USB-MOなどの記録媒体を接続するためのプロトコルも「USBマスストレージクラス」として定義され、これも公開されている。

USBマスストレージクラスには、割り込み転送とバルク転送の両方を用いる方式（CIB転送）と、バルク転送のみを用いる方法の2種類が定義されているが、バルク転送のみを使うほうが簡単なので、こちらの方式を採用しているデバイスが多いように見受けられる。なお、マスストレージクラスではUSB通信時のプロトコルとしてSCSIを使用している。

最近発売されるデジタルカメラのほとんどの機種では、USBインターフェイスが採用されているが、USBマスストレージクラスを採用している機種はあまり多くないようである。たとえば、キヤノンのIXY DIGITALは、アプリケーションによる転送（TWAIN）のみなので、WindowsやMacintoshのドライブとして認識することはできない。このような機種では、専用のプロトコルを使用しているため、Linuxから使用できるようにすることはたいへん困難である。

一方、富士写真フィルムのFinePix4700Zは、Linux-2.4.0-test5ではUSBマスストレージクラスとして認識されて、/dev/sd?が割り当てられ、ファイルを正常に読むことができた。しかしながら、このカメラはRead Onlyでしかマウントできない仕様となっているため、書き込みを行うことができないのが残念である。

ただし、マスストレージクラスだからといってLinuxで使えるとは限らない。たとえば、筆者が所有しているハギワラシスコムのスマートメディアリーダ、Flash Gate-IIをLinux-2.4.0-test5へ接続するとカーネルがパニックを起こしてしまう。

このデバイスは、Windows 98、Windows 98 Second Editionはもちろん、Windows 2000、Windows Me（β版）、MacOS-9.0.4ではOS標準のドライバでまったく問題なく動作する、いわばリファレンス的なUSBデバイスであるので、Linuxカーネルのエラー処理に問題があると思われる。

ところで、今回USBデバイスドライバの開発を行うのに使用したノートPCのVAIO PCG-Z505GR/Kには、メモリスティックスロットが搭載されている。この機種の数世代前（PCG-Z505DX）ではIDE接続だったので、PCG-Z505GR/Kも同じだろうと思っていたのだが、Linuxをブートしたところ、内部では内蔵USBハブを介してUSB接続になっていることがわかり、たいへんびっくりした。

なお、このメモリスティックスロットは、linux-2.4.0-test5では、USBマスストレージクラスとして認識され、/dev/sdaに割り当てられる。つまり、SCSIディスクとまったく同じようにアクセスできるので、「mount -t vfat /dev/sda1 /mnt」というコマンドでマウントすることができる。筆者が試した範囲では、マウントにちょっと時間がかかるのが気になるが、正常に読み書きすることができた。

以上のように、LinuxにおけるUSBマスストレージクラスデバイスのサポートは、一部にそれなりに使えるデバイスもあるものの、まだ十分とはいえないのが現状である。

しかし、マスストレージクラスは規格が明確であるので、USBアナライザさえあればLinuxでサポートすることはさほど難しくない。ほとんどのストレージデバイスが問題なく使えるようになるのに、そんなに時間はかからないのではないかと思われる。

特別講座

ステップアップC言語

リスト構造

■ダイナミックに増減するデータの保存

カーネルのなかでは、さまざまなデータを扱う必要がある。たとえば、USBデバイスの場合は、ケーブルが接続された瞬間に、そのデバイスのさまざまな情報を集めてカーネル内に保存しなければならない。

このような動的に変わるデータを保存するためには、デバイスの情報を構造体にまとめ、その配列を定義するという方法がある。しかし、この方法ではカーネル内に固定的にメモリが確保されてしまうため、メモリの無駄が生じてしまう。

また、要らなくなったデバイスのデータは削除することが望ましいが、配列として保存するとデータが歯抜けになり、無駄なメモリ空間が生じてしまう。

■リスト構造

このようなダイナミックに変わりうるデータを保存するには、「リスト構造」を採用することが多い。リスト構造とは、ある並びのデータをポインタで動的につなげていく構造である。Linuxカーネルでも、このリスト構造が多用されるので、リストの操作のために、

```
/usr/src/linux/include/linux/list.h
```

というインクルードファイルが用意されている。

■リスト構造を実現する構造体

このlist.hでは、リスト構造を実現する構造体が、次のように定義されている。

```
struct list_head {
    struct list_head *next, *prev;
};
```

ここで、「next」はこのリストの次のリストを示すポインタであり、「prev」はこのリストの前のリストを示すポインタである。

たとえば、List0、List1、List2、List3がこの順番に並んでいる場合は、下図のようにポインタがそれぞれのリストを指し示すことになる。

このリスト構造は、nextとprevの両方向のポインタをもつため、「双方向リスト構造」と呼ばれるが、用途によっては片方向だけのリスト構造も使われることがある。

■リストへの追加

このような双方向リスト構造に、新しいリストを追加するために使われるのが、list.hで定義されているlist_add関数である。この関数は、

```
list_add(new,head)
```

として呼ばれる。ここで、「new」は新しく登録するリストへのポインタで、「head」は新しいリストを登録しようとする位置にあるリストのポインタである。

また、この関数の内部では、

```
__list_add(new, head, next);
```

という下請けの関数が呼ばれるが、この下請け関数では、

```
next->prev = new;
new->next = next;
new->prev = prev;
prev->next = new;
```

という操作が行われる。

つまり、今まで双方向につながっていたリストの位置に新しいリストを割り込ませて、つじつまがあうように前後の双方向ポインタを書き換えているわけである。

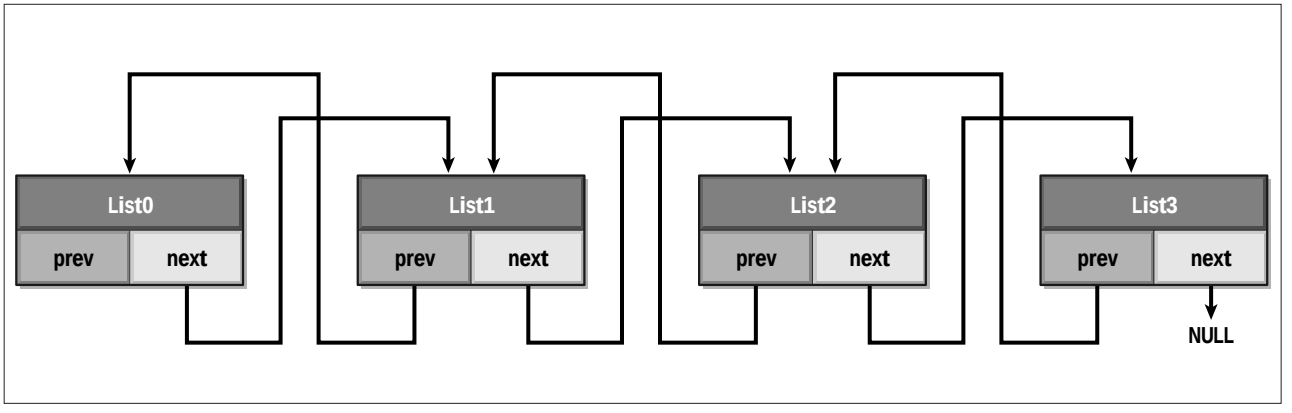
■リストの操作

list.hでは、リストの追加だけでなく、次のようにさまざまな操作が用意されている。

- **リストの初期化**
- **最後のリストの取り出し**
- **指定したリストの削除**
- **リストが空であるかの調査**
- **2つのリストの結合**
- **リストの内容の取り出し**
- **全リストへの操作**

カーネル内では、このリストの操作が多用されているので、カーネルを読むうえで、このリスト構造を理解することは重要である。

リスト構造への操作は一見複雑のように見えるが、実態は前後の参照をつなぎ変えるだけであるので、list.hをよく読めば理解することは難しくはないはずだ。



Rubyで行こう

本連載も10回目ということで、読者の皆さんのRuby経験値も高まってきているのではないでしょうか。今回は連載10回を記念して「十戒」というタイトルでRubyの「落し穴」を探ってみます。

第10回　十戒

文：赤松智也
Text : Tomoya Akamatsu

Rubyは人間が最も自然にプログラムできるように設計されています。少なくとも設計者のまつもとさんはそう言っています。しかし、人間はなかなかに複雑なもので、特に慣れないうちはいろんなところにつまずくものです。そういう意味では、どんな言語にも落し穴があると言ってもよいでしょう。PerlとかC++とか特に多そうですが、Rubyも決して例外ではありません。今回はRubyの落し穴を避けるためのルールをいくつか紹介します。

①変数の有効範囲を理解すべし

変数の名前で、変数の有効範囲がひと目でわかるのはRubyの大きな特長です。私の知る限りでは、このような特長を持つ言語はRubyしかありません。変数宣言は必要ないといわれるスクリプト系の言語でも、変数の有効範囲（スコープとも呼びます）に関しては宣言が必要です。

たとえば、Perlはlocalやmyという形でローカル変数を宣言する必要がありますし（宣言しなければグローバル変数）、PythonやTclは逆にグローバル変数のほうをglobalという形で宣言する必要があります（宣言しなければローカル）。まともにプログラムするならば、ほとんどの変数はローカル変数ですから、PythonやTclのほうが優れているといえますが、それでもローカル変数かグローバル変数かを区別するためには、宣言部分を探すか、いちいち覚えておくしかありません。

■変数のスコープ

一方、Rubyでは、変数のスコープはその名前で決まります（**表1**）。Rubyの方式では、宣言が不要なだけでなく、変数を見ればそのスコープがひと目でわかります。しかも、一番よく使われるローカル変数と定数は、先頭に記号が付かないのでプログラムが見やすく、グローバル変数を多用するような「良くない」プログラムは$だらけで見にくくなります。そのため、自然と悪いプログラムを書きたくなくなるのです。こういう性質の積み重ねが、よく言われる「Rubyの魔法」の秘密なのではないでしょうか。

しかし、なにごとも良いことばかりではありません。Rubyの変数にはちょっとした落し穴があります。まず、気をつけなければならないことは、変数の初期化についてです。

■変数の初期化

ローカル変数、クラス変数、それと定数は、使う前に必ず初期化が必要です。ローカル変数はそのメソッドの中で初期化されていないと、引数のないメソッド呼び出しと見

変数	種類	説明
foo	ローカル変数	小文字で始まる
$foo	グローバル変数	$で始まる
@foo	インスタンス変数	@で始まる
@@foo	クラス変数	@@で始まる（1.5系以降）
Foo	定数	大文字で始まる

表1　変数のスコープ

なされます。

```
def good
  print         # printメソッドの呼び出し
end

def bad
  print = 44
  print         # 変数printの参照
end
```

ローカル変数については、先頭の代入が宣言の代わりをしていると考えることができます。この宣言は実行時ではなく、コンパイル時に解釈されます。ですから、代入文が現れたことが重要で、実際にそれが実行されるかどうかは無関係です。

```
def bad2
  if false
    print = 44  # この代入は実行されない
  end
  print         # でも変数printの参照
end
```

クラス変数と定数に関しては、代入が宣言代わりなのは同じですが、初期化されていないと単にエラーになります。メソッドと取り違えたりということはありません。

■グローバル変数のスコープ

$で始まっている変数は、プログラム全体が有効範囲になります。ただし、これには例外があって、変数$_と$1などの正規表現のマッチ結果を示す変数たち（**表2**）は、$で始まるものの、実際はローカル変数です。

■インスタンス変数のスコープ

インスタンス変数は、現在selfが指しているオブジェクトに所属しています。ですから、同じselfを持つメソッド間では共有されますし、逆に同じメソッドでもselfが違えば異なる値を持ちます。たとえば、instance_eval()メソッドでselfが置き換えられると、インスタンス変数の値も変わってしまいます。

変数	説明
$~	最後のマッチに関するデータ
$1, $2..	n番目の()にマッチした文字列
$&	マッチした部分文字列
$`	マッチした部分より前の文字列
$'	マッチした部分より後ろの文字列
$+	最後の()にマッチした文字列

表2　正規表現のマッチ結果を示す変数

```
class Foo
  def initialize                # @varを15に初期化
    @var = 15
  end
end
class Bar
  def bar
    foo = Foo::new
    @var = 22
    puts @var                   # 22が出力される
    foo.instance_method {       # selfがfooになる
      puts @var                 # 15が出力される
    }
  end
end
```

■ローカル変数のスコープ

ローカル変数のスコープのルールは少々複雑です。まず、基本的なルールとして、ローカル変数はそれぞれの場所に固有のローカル変数スコープに所属し、外や中から参照されることはありません。ローカル変数スコープの種別は以下のとおりです。

・クラス定義

Rubyではクラス定義も実行文ですから、その中に文を含むことができますし、ローカル変数を使うこともできます。クラス定義が終われば、それらのローカル変数は消えます。

・モジュール定義

モジュール定義はクラス定義と同様です。

・メソッド定義

メソッド定義の中で使われたローカル変数はそのメソッドの中でだけ有効です。

・トップレベル

クラス定義、モジュール定義、メソッド定義の外側をトッ

プレベルと呼びます。loadやrequireで読み込まれたプログラムではそれぞれ別のトップレベルを持ちますから、プログラムをロードすることによってトップレベルのローカル変数を変化させることはありません。

これだけならさほど面倒ではないのですが、歴史的な経緯から、以下のような少しややこしいルールがあります。

(a) ブロックは新しいスコープを導入する
(b) このスコープからは外側のスコープが見える

このルールは手続きオブジェクトやスレッドにローカルな変数を提供するためのものです。(a) がなければ、手続きオブジェクトは一切ローカル変数が使えず面倒なことになりますし、(b) がなければ、外側の変数がアクセスできず結局クロージャでなくなってしまいます。また、普通の繰り返し処理が非常に面倒になるでしょう。『コールバック』の回（2000年6月号）で説明した、Cのコールバック関数のような使いにくさです。

そういうわけで、これらのルールの導入はある意味で必然だったわけですが、必然の組み合わせでも場合によっては、つまづきやすい結果になることがあります。それは、うっかりローカル変数をブロックの中で最初に使ってしまうと、その変数がブロックを出たときに消えてしまうことです。ずいぶんRubyに慣れた今でも、ときどきひっかかります。このプログラムを見てください。

```
lines = file.readlines
lines.each |line| do
  if /pattern/ =~ line  # lineがpatternに合致したら
    pat = line          # patに代入してループ終了
    break
  end
end
print pat, "\n"         # patを出力したいけど、あれれ
```

このプログラムはすごく自然に見えますが、ルール (a) により、変数patはdo～endの範囲内だけで有効ですから、printの場所ではもう参照できません。プログラムの先頭に、「pat = nil」のような代入（宣言）を行えば、意図どおりに動作します。

この問題は、メソッドの先頭でローカル変数の初期化を行うことで解決できます。メソッド全体で使うローカル変数すべてが、先頭で代入が行われていれば、人間もRubyもどれがブロックの中だけで有効なローカル変数かについて悩む必要はなくなるわけです。Ruby本にも「メソッドの先頭でローカル変数の初期化を行う」という「ローカル変数のオキテ」が紹介されています（47ページ）。

まつもとさんが自ら書かれた文に反論するのも変な話ですが、せっかく宣言の要らないRubyで書いているのに、メソッドの先頭で実質的なローカル変数の宣言を行うというのは、なんだかなあって思います。やはり使うところで必要に応じて宣言したいのです。

[ruby-talk:04397]によれば、まつもとさん自身もときどきこの落し穴にはまってしまうそうで、将来のバージョンでこの件をなんとかしたいと考えているようです。が、ルールがもっと複雑になるかもしれないので、すっきりした解答にはならないかもしれません。

■定数のスコープ

定数のスコープは以下のようになっています。

・定義されたクラスまたはモジュールの定義内

メソッド定義の内外を問わず参照できます。定義内に別のクラスやモジュールが定義されていれば、その内部でも参照できます。

・サブクラスまたはインクルードしたクラスまたはモジュール

例をあげると以下のようになります。

```
                # 有効な定数
C1 = 10         # C1
module C2       # C1,C2
  C3 = 20       # C1,C2,C3
end
module C4       # C1,C2,C4
  C5 = 30       # C1,C2,C4,C5
  include C2    # C1,C2,C3,C4,C5
end
class C6        # C1,C2,C4,C6
  class C7      # C1,C2,C4,C6,C7
  end
end
class C8<C6     # C1,C2,C4,C6,C8
```

```
  C9 = 40        # C1,C2,C4,C6,C8,C9
  include C4     # C1,C2,C3,C4,C5,C6,C8,C9,
end
                 # C1,C2,C4,C6,C8
```

■クラス変数のスコープ

1.5系で新しく導入されたクラス変数は、クラスまたはモジュールに所属する変数です。クラス変数はクラスまたはモジュール定義の中、メソッドが定義されるのと同じレベルで代入される必要があります。これが宣言の代わりになります。クラス変数は、宣言が行われたクラスまたはモジュールおよびその子孫の範囲内なら、メソッドの内外どちらでも参照できます。この場合の子孫とは、クラスの場合ならサブクラス、モジュールの場合なら直接または間接にそのモジュールインクルードしたクラスやモジュールのことです。

②変数を初期化すべし

たとえば、Cでは宣言したが初期化していない変数の値は不定です。なにが入っているか保証されませんし、実際によくわからない値（ゴミ）が入っています。変数の初期化忘れは、やっかいなバグの原因になります。それでなくても、使う変数に最初にどのような値が入っているべきかを明確にしておくことは重要です。

Rubyでは、ローカル変数、クラス変数、定数は、初期化しなければ使えませんし、グローバル変数とインスタンス変数は未初期化の場合は値がnilと決まっています。ですから、RubyではCのような類のバグは発生しません。「初期化すべし」というのは、バグを避けるためではなく、意図を明確にするためなのです。

③グローバル変数は使うべからず

グローバル変数には、以下の問題があります。

- **プログラムのどこからでも参照できるので、影響範囲が大きい**
- **同様の理由でだれが（プログラムのどの部分が）グローバル変数の値を変更したのか捕捉するのが難しい**
- **グローバルな値を変更してしまうと、スレッドが破綻する（ことが多い）**
- **プログラムが$だらけで汚くなる**

Rubyでのグローバル変数は、わかっている人が自己責任を持って使うためのものです。たとえば、ワンライナーで活用するのはともかく、ある程度以上のプログラムでは使うべきではないでしょう。

通常のグローバル変数は、インスタンス変数やクラス変数あるいは定数を用いることで、ほとんどの場合には代用できます。

$で始まる特殊変数は、ほとんどすべてグローバル変数を用いないで、その機能を用いることができるようになっています。たとえば、入力レコード区切り$/は、getsやreadlineメソッドの引数として明示的に指定できます。

④空白を一貫性を持って使うべし

Rubyは、空白やかっこの使い方で意味が変化することがあります。もちろん便利なのでですが、ときどき不安になります。たとえば、空白については以下のような例があります。

```
（あ）foo+2
（い）foo +2
（う）foo + 2
```

上の例は、メソッドの第1引数と演算子の区別があいまいになるものです。（あ）と（う）は変数fooと2の足し算、（い）はメソッドfooを+2を引数に呼び出したものと見なされます。では、以下の違いがわかりますか？

```
foo * bar
foo *bar
```

このようなあいまいさを呼ぶ記号は、「+ −＊ / % & [{」です。これらの記号を使うときには、両側に空白を置くか、両側に空白を置かないように気をつける必要があります。演算子の両側の空白に一貫性を持たせるというのは、基本中の基本ですから、まっとうなプログラムを書いていればそれほど問題にならないかもしれません。

⑤かっこを適切に使うべし

空白と並んで重要なのはかっこです。かっこは、式のグループ化と、メソッド呼び出しの引数を囲むことの両方の目的があります。おまけにRubyではメソッド呼び出しの

引数のためのかっこは省略できますから、少々話がややこしくなります。たとえば、「puts (1+2).abs」と書いてあったら、以下のどちらにも解釈できるような気がします。

(a) puts((1+2).abs)
(b) (puts(1+2)).abs

ここでのRubyのルールは「メソッド呼び出しのように見えるものはメソッド呼び出し」です。ですから、識別子の後ろにかっこがきたものは、すべてメソッド呼び出しと見なされます。よって、これは必ず（b）と解釈されます。

（b）は一見すると、「(1+2).abs」を引数にするようですが、実際に引数になるのは、「(1+2)」の部分だけで、absメソッドはputsメソッドの戻り値に対して呼び出されます。このルールはPerlと同じです。

これだけを見ると自然といえば自然なルールですが、プログラムを以下のように段階的に書き換えていくと、結構引っかかります。

```
puts a           # aを出力
puts a+b         # bの値も足さなきゃ
puts (a+b).abs   # 絶対値がほしいな（あれれ？）
```

⑥こまめに-wオプションを使うべし

皆さん、-wオプションを使ってますか？ -wオプションは以下のチェックを行ってくれます。

- **未初期化のグローバル変数へのアクセス**
- **未初期化のインスタンス変数へのアクセス**
- **あいまいな第1引数**
- **識別子と引数かっこの間の空白**
- **既存のメソッドの置き換え**
- **プライベートな属性の定義**
- **到達不能な実行文**
- **演算結果が無視される演算子**
- **strftimeに空白フォーマット文字列**
- **クラス変数、定数の重複定義**

すでに延べた②、③、④のルールはチェックしてくれるようです。これだけチェックしてもらえれば、かなりの間違いを検出できます。特にチェック項目の最初の4つは、タイプミスや文法の誤解などのプログラムの単純な間違いを発見してくれそうです。

-wオプションを使うと警告が出る標準添付のライブラリもあるので、いつも使うとうっとうしいのですが、数多くの問題を見つけてくれることは確かですから、ときどきは使ってみることをお勧めします。

⑦参照を理解すべし

「三つ子の魂百まで」という諺もありますが、慣れ親しんだプログラミング言語の影響は大きいものです。BASICやPerlに慣れた人は、この「参照」に引っかかるようです。逆に、LispやSmalltalk、Pythonから来た人や、Cのポインタを理解している人にとっては、さほど問題になりません。

重要なポイントは、「Rubyではすべての値は参照である」ということです。変数に入っているのはオブジェクトそのものではなく、オブジェクトへの参照、言葉を換えれば、あるオブジェクトへ向かう矢印に過ぎません。

変数aからbへの代入を行うということは、変数aが参照しているオブジェクトを変数bからも参照させるということです。ここでオブジェクトのコピーは行われません。

ところがBASICやPerlでは、変数はデータの容器であり、代入はある容器に入っているデータを別の容器にコピーすることです。

この違いが問題になるのは、オブジェクトの状態が変化するときです。Perlタイプでは、代入にあたってコピーが行われるので、元のオブジェクトを書き換えてもコピーにはなんの影響も与えません。

```
#!/usr/bin/perl
$a = "abc";      # 変数$aを"abc"に
$b = $a;         # 代入（コピー）
chop($a);        # $aの末尾"c"を削除
```

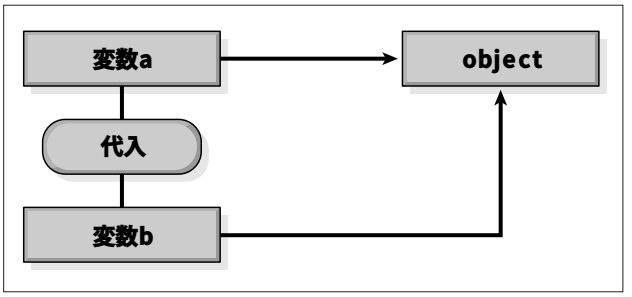


図1 Rubyの代入

```
print $a;        # $aの値は"ab"
print $b;        # $bの値は"abc"
```

Rubyタイプでは、代入によって同じオブジェクトへの参照が発生します（**図1**）。

この結果、Perlタイプに慣れた人には驚きの結果が起きることがあります。

```
#!/usr/bin/ruby
a = "abc"        # 変数aを"abc"に
b = a            # 代入（同じオブジェクトの参照）
a.chop!          # aの末尾"c"を削除
print a          # aの値は"ab"
print b          # bの値も"ab"
```

aもbも同じオブジェクトを参照しているので、両方が同時に変化するのは当然といえば当然なのです。chop!メソッドのようなオブジェクトそのものを変化させるメソッドの多くに!マークが付いているのは、「気をつけてね」という意味です。以下のようにすればこの問題には引っかかりません。

```
#!/usr/bin/ruby
a = "abc"        # 変数aを"abc"に
b = a            # 代入（同じオブジェクトの参照）
a = a.chop       # aの末尾"c"を削除したものをaに代入
print a          # aの値は"ab"
print b          # bの値は"abc"
```

こんな単純な例では引っかからない人も、ちょっとひねりを加えると引っかかってしまうことがあります。たとえばこんな例です。

Hashオブジェクトは、キーが見つからないときに返すデフォルトの値を設定できます。「Hash::new(default)」とすれば、デフォルト値を設定したハッシュを作ることができます。ここで、「hash = Hash::new([])」として、デフォルトに配列を指定します。

さて、キーに対応した値をすべて覚えさせるために、少々不自然ですが、以下のようなメソッドを用意します。

```
def hash_store(hash, key, value)
  hash[key].push(value)
end
```

これで、valueに設定した値が配列に各キーごとに次々追加されるはずです。やってみましょう。

```
hash_store(hash, 1, 2)
hash_store(hash, 2, 3)
hash_store(hash, 1, 4)
```

hashはどうなったかというと、こうなっています。

```
p hash[1]        # [2,4]を期待するが[2,3,4]
p hash           # なぜか空のまま
```

デフォルトは配列への参照ですから、pushメソッドはデフォルトの配列に一生懸命値を追加するだけで、ハッシュの中身はなにも変化していなかったのです。

たとえば、hash_store()を以下のように変更すれば、望みどおりになるでしょう。

```
def hash_store(hash, key, value)
  hash[key] += [value]
end
```

これならハッシュの更新もきちんと行われます。もっとも毎回配列の生成が行われるので効率は良くありません。

Column

Perl/Ruby Conference

2000年11月29日～12月1日、国立京都国際会館で「Linux Conference 2000 Fall」と併催で、「Perl/Ruby Conference」が開かれます。Perlと共同ではありますが、Rubyに関する最初の本格的なカンファレンスが開かれることになります。

Perl/Ruby Conferenceでは基調講演、テクニカルセミナー、ビジネスセミナー、チュートリアルが行われ、国内外の著名人が招聘されます。まつもとさんは、きっと呼ばれるでしょうね。現時点では参加費などは未定ですが、それほど高いものにはならないようです。

平日の京都というのが辛いところではありますが、私も参加するべく準備を始めようと思います。京都で読者の皆さんにお会いできることを楽しみにしています。見かけたら声をかけてください、と言いたいところですが、私は自分の本名をまだ明らかにしてないんですよねえ。

以下のようにすれば効率が良くなります。

```
def hash_store(hash, key, value)
  if hash.key?(key)
    hash[key] << value
  else
    hash[key] = [value]
  end
end
```

⑧むやみにコピーをするべからず

先に話した参照について理解が進むと、次に陥りやすいのは「なんでもコピーする」くせです。オブジェクトの代入や引数渡しなどが、参照がそのまま渡るだけだという事実が理解できると、今度は「じゃあ、渡された先でオブジェクトが書き換えられたらどうしよう」と不安に陥ります。そして、以前に慣れ親しんだ言語と同じようにしようと、dupとかcloneなどのメソッドを使って、代入や引数渡しのたびにオブジェクトのコピーを取ろうとします。freezeメソッドを使って、オブジェクトを更新できないようにしてから渡すことを考える人もいます。

でも、それって本当に有効なんでしょうか？ dupやcloneによるコピーはシャローコピーです。つまり、オブジェクトはコピーされますが、そのオブジェクトが参照していたものまではコピーされません（**図2**）。

いくらaのオブジェクトのコピーを作っても、それが参照しているbのオブジェクトが変更されてしまえば、結局は一緒です。freezeについても更新が禁止されるのはそのオブジェクトだけで、そこから参照されている先までは禁止されませんから、同じことです。

それだけではありません。Rubyは動的な言語なので、aというオブジェクトが所属しているクラスそのものを書き換えることも可能です。

このような理由で、コピーやfreezeがいつも有効であるとは限らないうえ、無駄なコピーはプログラムの実行を遅くします。コピーそのもののコストのほかに、コピーによって増加するオブジェクト数はガベージコレクションの効率を低下させます。

というわけで、むやみなコピーはさほど根拠のない、不安に対するおまじないのようなものです。すべてが参照であることを把握していれば、変更していいものと、いけないものがわかるでしょう。

⑨むやみに型チェックをするべからず

過去に「静的型言語」を経験した人は、型チェックを入れたくなるようですが、あまり勧められません。

静的型言語とは、変数や式の型がコンパイル時に決まるタイプのプログラミング言語です。逆に、Rubyのように、変数も式も型は実行時にしか決まらない、どの変数にどのクラスのオブジェクトが格納されるのか事前にはまったく予想できないタイプのプログラミング言語を「動的型言語」と呼びます。C++やJava、Eiffel、MLなどが静的型言語の代表です。

静的言語では引数や変数などに型がありますから、メソッド（関数）呼び出しや代入について、型のチェックが行われます。これがプログラムのコンパイル時に行われるので、型の間違いに関するエラーが早い段階で見つかります。型の宣言が面倒である代わりに、エラーが早く見つかるわけです。

一方、Rubyのような動的言語では、実際に実行してみるまで型の間違いは検出されません。プログラムの規模が大きくなれば、いつも全体が実行されるとは限りませんから、隠れたバグが発見されずに残る可能性も否定できません。

そこで、静的型に慣れた人は不安に陥り、以下のような形で型チェックを挿入したくなります。

```
def foo(var)
  unless var.kind_of?(Integer)
    raise TypeError, "argument must be an integer"
  end
  ...
end
```

これは必ずしもお勧めではありません。というのも、静

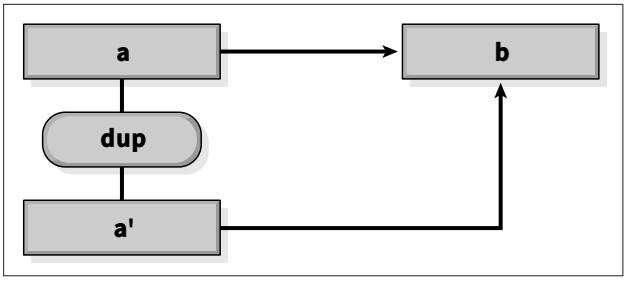


図2　シャローコピー

的言語では引数が適切かどうかは、ある型に適合するかどうかで決定されます。しかし、動的言語ではむしろ必要なメソッドを持っているかどうかで決まることが多いからです。そのようなメソッドの集合を、「インターフェイス」とか「プロトコル」と呼ぶことがあります。単純な例では、あるメソッドの処理に「オブジェクトの長さ」を知る必要があれば、引数はsizeメソッドを持つだけで十分であり、たとえば、Arrayに限定する必要はありません。

そのような型チェックを明示的に行わなくても、ほとんどの場合には「メソッドがない」、「引数の数が違う」のいずれかのエラーが発生するか、組み込みメソッドが行ってくれる型チェックに引っかかります。プログラム中での型チェックは、むしろプログラムを窮屈なものにしてしまう可能性があります。静的言語には静的言語の、動的言語には動的言語の良さがあります。RubyではRuby流がよろしいのではないでしょうか。

⑩クラスの設計はボトムアップすべし

私は、オブジェクト指向設計の専門家ではないので、あまり偉そうなことは言えないのですが、経験上Rubyはトップダウン設計には向かないような気がしています。

まず、目的の具体的なクラスを作って、そこから共通部分を括りだして、スーパークラスやモジュールを作り出すやり方のほうが、使いやすいクラスができると思います。クラスの設計についてはほかにも、

- **クラスを作りすぎない**
- **既存のクラスを活用する**
- **継承階層を深くしない**

なども言えるような気がします。この件については、私の好みもかなり含んでいますが。

まとめ

「Rubyの落し穴」と「注意すべきルール」をまとめて来ましたが、考えてみると②、③、④、⑤は「悪いプログラムを書きにくくする」という効果がありますし、⑦、⑧、⑨はどちらかというと考え方の違いのように思われます。

ですから、Rubyに慣れてしまった人には、これらのルールは不要かもしれません。Rubyを学び始めた人は、ここに挙げたようなルールに従うことで、かつて私が引っかかったような落し穴を避けることができるでしょう。

Column

今後のRuby 1.7

原稿執筆時点では、1.6はまだ公開されていません。開発版である1.5系が、次の安定版である1.6系として公開されると、ほぼ同時に次の開発版である1.7系の開発が始まります。

1.7系では、仕様を早期に安定させるために先送りされた機能を含めて、機能追加と改善が行われます。まつもとさんに聞いてみたところ、1.7系で検討したい項目は以下のようなものがあるそうです。ただ、これらは決定したものではなく、こんなことができたら良いなあ、というようなレベルのものだそうです。

■キーワード引数

省略可能な引数をラベルを付けて指定できる機能です。「Tk::button(parent: root, label: "hello")」のような形式になりそうです。

■インタプリタの書き換え

そろそろ現在のインタプリタの限界が見えてきたそうで、高速化のために全面的に書き換えることを検討しているそうです。

■バイトコード化

高速化に伴って、プログラムの内部表現を現在の構文木形式からバイトコード形式にするかもしれないそうです。ただし、やってみてメリットがなければやめるかも、ということです。

■GCの改善

GC（ガベージコレクション）のコストが結構高いようなので、GC部分を改善したいとのことです。あるいは[ruby-dev:9990]でアナウンスされたような世代別GCが取り込まれるかも。

■ANSI C化

まつもとさんはANSI Cスタイルがお嫌いだそうで、Rubyのソースは未だにold K&Rスタイルで書かれています。が、そろそろ潮時か、と言ってました。

■国際化

ライバルであるPerlやPythonは新バージョンでUnicode対応しています。Rubyは以前からマルチバイト文字列に対応していますし、UTF-8も扱えるので今まではやや先行していましたが、Unicode対応については優位性がしだいにあやしくなってきたようです。たしかに、Rubyでは文字とバイトの扱いが統一されているとは言いがたいですし、また「真の」国際化には単にマルチバイトエンコーディングが扱えること以上の機能が必要です。1.7では専門家にも使ってもらえるレベルの国際化をめざすとのことです。

1.7系が安定したら、1.8として公開されるか、あるいは繰り上がって2.0となるのかは未定だそうです。いずれにせよ楽しみですね。

賢く使うUNIX

これであなたもスマートなUNIX使い！

【最終回】タイムスタンプの活用

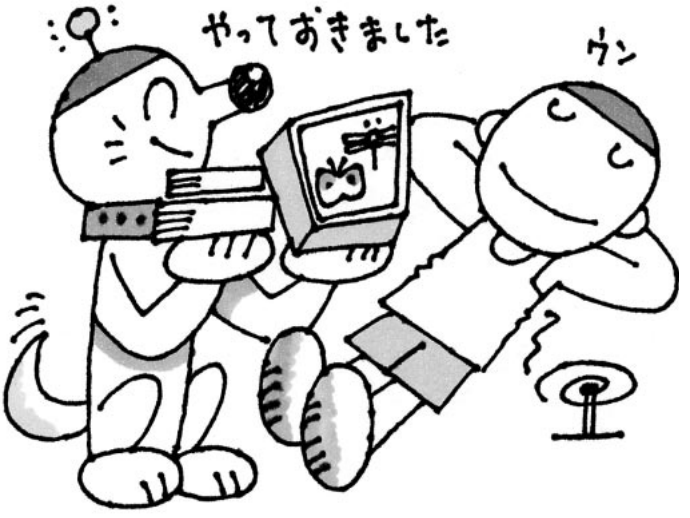


Illustration : Manami Kato

Linuxの標準シェルであるbashのコマンドラインを中心として、Linuxの便利な使い方について紹介してきた本連載もこれで最後。最終回である今回は、ファイルの修正時間が記録されたタイムスタンプを取り上げ、ls、cp、tar、touchなどタイムスタンプに関係したコマンドを解説し、findを使ったスクリプトを作成する。

今月のお題

・指定したファイルの3種類の時間情報をまとめて表示する
・最近n日以内に更新したファイルを新しいものから順に表示する

文：**大池 浩一**
Text : Kouichi Ooike

UNIX系OSを含むたいていのOSでは、ファイルを新たに作成したり、ファイルの内容を修正したりする際、ファイル情報の一部として最後に修正した日時が記録される。この修正時間のことを、手紙などの発送（あるいは受け取り）日時を記録するために押すスタンプになぞらえて、「タイムスタンプ」と呼ぶ。

われわれがふだん、lsなどの出力として目にするファイルの時間情報はこのタイムスタンプだ。DOSの場合はタイムスタンプだけだが、UNIX系OSのファイルシステムでは、さらに「アクセス時間」と「ステータス変更時間」を加えた3種類の時間情報が記録される。いずれもlsのオプション指定で表示可能だ。なお、Windowsのファイルシステム（VFAT）も3種類の時間情報を記録するが、その内容はUNIX系OSのものとは内容や精度が異なる。

タイムスタンプやその他の時間情報は、ファイルを扱うコマンドを使用すると自動的に更新されるので、通常はユーザーが意識する必要はない。たとえば、エディタでファイルを修正して保存すると、タイムスタンプが更新される。また、lessなどでファイルの内容を閲覧するとアクセス時間が更新され、chmod、chownなどでファイル属性を変更するとステータス変更時間が更新される。コマンドのオプション設定によっては、タイムスタンプを変更しないでファイルをコピーしたり、タイムスタンプだけを意識的に変更することも可能だ。

また、ファイルをコピーする際、コピー先ファイルが古い場合にだけ上書きするcpの-uオプションや、ビルド後に修正したファイルだけコンパイル対象とするmakeなど、さまざまなコマンドがタイムスタンプなどの時間情報を利用した処理を行う。

今回の記事の前半では、UNIX系OSのファイルシステムが利用する3種類の時間情報の内容や、Windowsなど他のOSとの違いについて説明し、lsでの時間情報の表示方法をはじめ、cpやtarを利用してタイムスタンプを変更せずにファイルをコピーする方法、タイムスタンプを意識的に変更するtouchの使い方を解説する。後半の「今月のお題」では、findによる時間情報の利用例として、指定したファイルの3種類の時間情報をまとめて出力するスクリプトと、指定したディレクトリ以下のファイルの中から、1週間以内に作成・更新したファイルを検索し、新しい順に並べて出力するスクリプトを作成する。

タイムスタンプなどの時間情報を活用しよう

初めに、UNIX系OSのファイルシステムで使われている3種類の時間情報、修正時間（mtime）、アクセス時間（atime）、ステータス変更時間（ctime）について説明しよう（**図1**）。なお、カッコ内の文字列は、それぞれの時間情報の名称としてマニュアルページで使われるので、あわせて覚えておきたい。

(1) 修正時間（mtime）……ls -l

ファイルが最後に修正（書き込み）された日時が記録された、いわゆるタイムスタンプ。ファイルを新規作成した場合はその日時、その後修正を加えた場合は修正日時が記録されている。

lsに-lオプションを付けると、ファイルのパーミッションやサイズなどとともに、この修正時間が表示される。また、-tオプションを付けた場合は、修正時間の新しいファイルから順番に表示されるようになる。これらのオプションは、「-lt」のようにまとめて指定可能だ。

たとえば、

```
$ ls -lt
```

とすると、カレントディレクトリのファイルやディレクトリの詳しい情報が、修正時刻の新しいものから順に表示される。

古いものから順に表示したい場合は、表示を逆順にする-rオプションを追加して、

```
$ ls -ltr
```

とすればいい。

ところで、lsの-lオプションで出力される時間情報には、2種類の異なる形式が存在する。時間情報の日時が6カ月以内の場合は、

```
Aug 23 04:01
```

のように、「月・日・時刻（秒なし）」という形式で出力されるのに対し、それより古い場合には、

```
Apr 30  1999
```

と、「月・日・年」という形式で表示される。新しいファイルほど、細かい時間を知りたいことが多いので、このような仕様になっているわけだ。

もし、時間情報を統一された長い形式で表示したいなら、--full-timeオプションを付けて、

```
$ ls -ltr --full-time
```

とする。この場合、時間情報は、

```
Wed Aug 23 04:01:33 2000
```

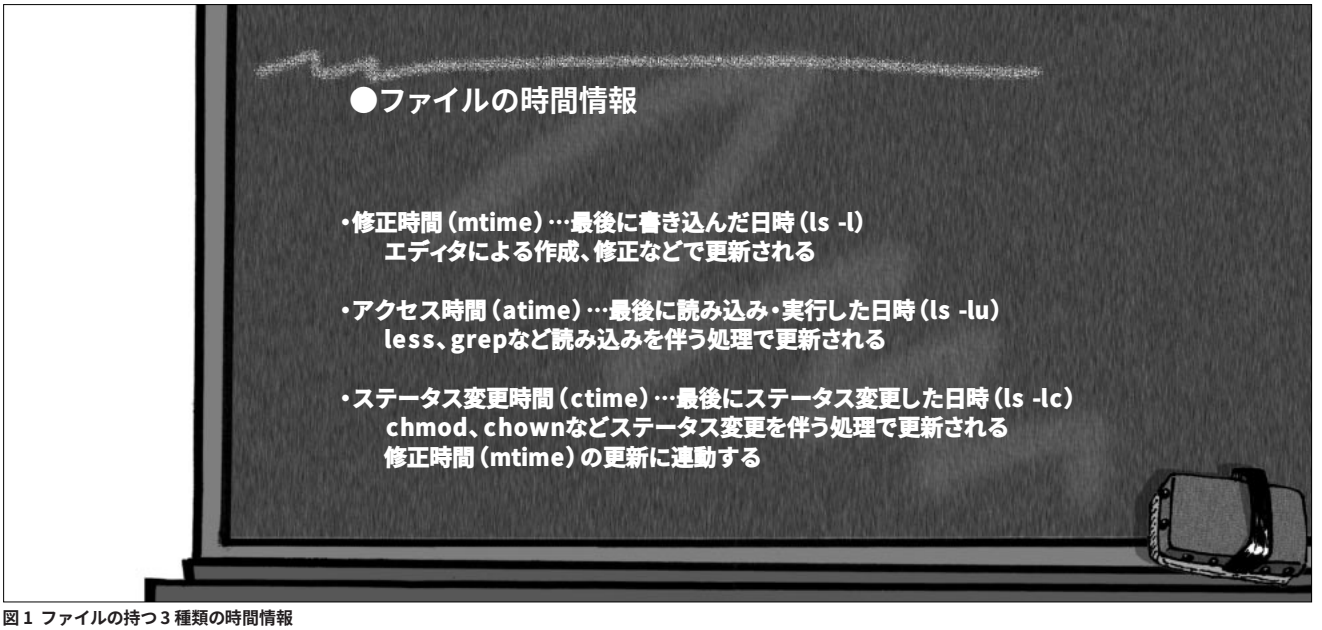


図1 ファイルの持つ3種類の時間情報

のように、「曜日・月・日・時刻（秒あり）・年」という順番ですべての情報が出力される。なお、--で始まる長い形式のオプションは、1文字オプションとまとめて記述できないことにも注意されたい。

(2) アクセス時間（atime）……ls -lu

ファイルが最後にアクセスされた日時が記録された情報。この場合、アクセスとは「読み込み」と「実行」を意味する。最初はファイルを作成した日時が記録されており、その後lessで内容を閲覧したり、grepで検索したりといった読み込みを伴う処理を行うと、その時点の日時で更新される。また、実行属性を持つファイルやシェルスクリプトを実行すると、実行時の日時が記録される。

lsでアクセス時間を表示するには、-lと-uオプションを組み合わせて「-lu」とすればいい。これで、アクセス時間の新しいものから順に表示される。また、逆順にする-rを追加すれば、アクセス時間の古いものから順に表示されるようになる。

たとえば、

```
$ ls -lur /usr/bin
```

とすると、/usr/binにあるファイル（ほとんどはコマンド）の詳しい情報が、アクセス時間の古いものから順に表示される。おそらく、先頭付近には見たことも聞いたこともないいようなコマンドが、インストールした日時とともに表示されているはずだ。

なお、先頭10個のコマンドだけ表示するには、

```
$ ls -lur /usr/bin | tail +2 | head
```

とすればいい。lsの出力の先頭行にはブロックサイズの合計表示が含まれるため、tailの＋2オプションで2行目以降を取り出し、headで先頭から10行分（lsの出力の2～11行目）を取り出している。

(3) ステータス変更時間（ctime）……ls -lc

ファイルのステータスが最後に変更された日時が記録された情報。最初はファイルを作成した日時が記録されており、chownでファイルの所有者を変更したり、chmodで属性（パーミッション）を変更したり、lnでハードリンクを行うなど、ステータスの変更を伴う処理を行うと、その時点の日時で更新される。

また、修正時間（mtime）が更新されると、ステータス変更時間も同時に更新されるため、両者が同一になっているファイルも多い。一方、アクセス時間（atime）の更新には連動しない。

lsでステータス変更時間を表示するには、-lと-cオプションを組み合わせて「-lc」とすればいい。これで、ステータス変更時間の新しいものから順に表示される。ほかの時間情報と同様、-rを追加して古いものから順に表示することも可能だ。

なお、ctimeの意味として「ファイルを生成した日時」と説明されることがあるが、これは完全な誤り。おそらく、Windowsのファイルシステム（VFAT）に用意された「作成日時」との混同と思われる（**コラム**参照）。

Column

Windowsファイルシステムの時間情報

「長いファイル名」に対応したWindows 95以降のファイルシステム（俗に「VFAT」と呼ばれる）では、ファイル情報の一部として、タイムスタンプをはじめとする3種類の時間情報を持つ。しかし、その内容はUNIX系OSの時間情報とは似て非なるものだ。

UNIX系OSの修正時間（mtime）に相当するのが「更新日時」で、DOSのファイルシステムと互換性がある（DIRでこれが表示される）。記録の最小単位が2秒なので、奇数秒の更新日時は存在しない。

残りの2つは、VFATで拡張された「アクセス日」と「作成日時」だ。アクセス日は、アクセス時間（atime）とほぼ同一の概念だが、名前の末尾が「日」であることに注意されたい。最小単位が1日なので、最後にアクセスした日付しかわからないのだ。また、作成日時は、文字通りファイルが作成された日時が記録される領域で、ファイルの更新後も変化しない（最小単位は1/100秒と細かい）。

UNIX系OSのステータス変更時間（ctime）が、「ファイルを作成した日時」と誤って説明されることがあるのは、VFATの作成日時と混同しているのだろう。なお、LHAなどのアーカイバを使うと、作成時のタイムスタンプ（更新日時）が展開時に復元されるため、作成日時より更新日時のほうが古いというわけのわからない状態になる。

UNIX系OSの時間情報とは大きく異なることがおわかりいただけただろうか。このほかにも、UNIX系OSのファイルシステムでは時間情報をUTC（協定世界時）で記録しているのに対し、WindowsのVFATではローカルタイム（地方時）のまま記録するという違いもある。

タイムスタンプを変えずにファイルをコピーする

通常、cpを使ってファイルをコピーすると、コピー先のファイルのタイムスタンプは、その時点の日時になる。また、時間情報以外のファイル情報（所有者やグループ、パーミッションなど）も変更される。

内容は同じでも別のファイルなのだから、考えてみれば当然のことだ。しかし、時にはコピー元のファイルと同じファイル情報が必要なこともある。たとえば、ハードディスクが手狭になってきたので、/homeディレクトリ以下をまるごと別のパーティションにコピーして引越しするといった場合だ。こうしたケースでは、元のファイルの時間情報やパーミッションなどをコピー先でもそのまま再現しなければならない。

以下では、タイムスタンプやその他のファイル情報を変えないままファイルをコピーする方法を2種類紹介しよう。ひとつは、cpの-aオプションを利用する方法、もうひとつはtarを利用する方法だ。

(1) cpの-aオプションを利用する方法

GNU版のcpには、元のファイルのタイムスタンプ、所有者、グループ、パーミッションを保持したままコピーする-pオプションが用意されている。これを使って、

```
$ cp -p hoge.txt /mnt/tmp
```

とすると、コピー元のhoge.txtのタイムスタンプがそのまま、コピー先のhoge.txtに設定される。なお、タイムスタンプ以外の時間情報については、コピー元のファイルと同じにはならない。

サブディレクトリ以下のファイルも含めてコピーするには、ディレクトリを再帰的にコピーする-Rオプション（大文字）を追加する。また、コピー元のファイルがシンボリックリンクの場合、通常はリンク先のファイルの内容がコピーされてしまうので、シンボリックリンク自体をコピーする-dオプションも付けておこう。

たとえば、/homeディレクトリ以下のファイルをすべて/mnt/tmpにコピーするには、スーパーユーザー（root）になった状態で、

```
# cp -dpR /home/. /mnt/tmp
```

とする。「-dpR」と同じ意味のオプションとして-aオプションが用意されているので、

```
# cp -a /home/. /mnt/tmp
```

としても同じだ。なお、コピー元を「/home/.」ではなく「/home」としてしまうと、/mnt/tmpにhomeディレクトリが作成されてしまうので注意されたい。

(2) tarを利用する方法

アーカイバーのtarを利用しても、元のファイルのタイムスタンプ、所有者、グループ、パーミッションを保持したままファイルのコピーを行える。

たとえば、cpの説明にも出てきた/home以下のファイルの/mnt/tmpへのコピーは、スーパーユーザーで、

```
# cd /home
# tar cf - . | (cd /mnt/tmp; tar xpf -)
```

とすればいい。2つのtarをパイプで接続することで、一時ファイルを介することなくメモリ上でアーカイブの作成と展開を行える。

まず、「cd /home」でカレントディレクトリをコピー元（/home）に切り替えた後、パイプ左側の「tar cf - .」でカレントディレクトリ以下の全ファイルをアーカイブして、標準出力に出力する。tarのcオプションはアーカイブの作成、fオプションはファイルを指定するものだ。また、出力ファイルの「-」は標準出力、アーカイブ対象の「.」はカレントディレクトリを意味する。

こうして標準出力に出力されたアーカイブデータは、パイプ右側のコマンドに渡される。これら2つのコマンドはカッコで囲まれているため、端末画面のシェルとは別のプロセス（サブシェル）で実行される。

まず、「cd /mnt/tmp」で、サブシェルのカレントディレクトリをコピー先（/mnt/tmp）に切り替える。なお、サブシェルのカレントディレクトリは親シェルには影響を与えないので、実行後のカレントディレクトリは元のままだ。続いて、「tar xpf -」で標準入力から読み込んだアーカイブデータを展開する。tarのxオプションはアーカイブの展開、pオプションはパーミッションの復元を指示するものだ。また、fオプションで指定された入力ファイルの「-」は標準入力を意味している。

タイムスタンプだけを変える

タイムスタンプは、さまざまなコマンドでファイルの新旧を判別する方法として利用されている。たとえば、cpに-uオプションを付けると、コピー先がコピー元より古い場合のみ上書きする。また、プログラム作成時に利用するmakeは、ソースファイルのタイムスタンプによってコンパイルの必要なファイルを判断している。こうしたコマンドを利用する際、タイムスタンプを変更する方法を知っていれば、いちいちエディタを起動したりする手間を省けるわけだ。なお、マシンの内蔵時計は正確な値に設定しておこう（**コラム**参照）。

ファイルのタイムスタンプを変更するには、touchを利用する。使い方は簡単で、引数としてファイル名（複数可）を指定すればいい。たとえば、

```
$ touch hoge.txt
```

とすると、hoge.txtのタイムスタンプ、すなわち修正時間（mtime）が現在の日時で更新される。

通常は、アクセス時間（atime）とステータス変更時間（ctime）も同じ内容で更新される。修正時間とアクセス時間を個別に変更することも可能だ。-mオプションで修正時間、-aオプションでアクセス時間が対象となる（ステータス変更時間はいずれの場合も更新される）。

現在日時以外の値を設定するには、直接日時をコマンドラインで指定する方法と、ほかのファイルの時間情報を利用する方法がある。

(1) コマンドラインで日時を指定する

-tオプションに続けて10進数で日時を指定する。通常は、「月・日・時・分」の順に2桁ずつ指定する。たとえば、

```
$ touch -t 09081430 hoge.txt
```

とすると、タイムスタンプに今年の9月8日午後2時30分が設定される。先頭に年（2桁または4桁）、末尾に「.」に続けて秒（2桁）を付けることも可能だ。

通常は、修正時間とアクセス時間が指定した日時で更新される（-m、-aオプションで個別変更も可能）。なお、ステータス変更時間は、指定した日時ではなく更新時の日時になるので注意されたい。

GNU版のtouchには、lsの-lオプションで使われている日時と同じ形式で指定できる-dオプションも用意されている。たとえば、上の例は、

```
$ touch -d 'Sep 8 2:30 pm' hoge.txt
```

としても同じ結果になる。

(2) ほかのファイルの時間情報を利用する

-rオプションに続けてタイムスタンプを参照するためのファイルを指定する。たとえば、

```
$ touch -r README hoge.txt
```

とすると、READMEのタイムスタンプの日時がそのままhoge.txtのタイムスタンプに設定される。

通常は、修正時間とアクセス時間が、それぞれ参照用ファイルの修正時間とアクセス時間で更新される。-m、-aオプションで個別に変更することも可能だ。ステータス変更時間は、参照用ファイルのステータス変更時間とは関係なく、更新時の日時になる。

Column

NTPサーバに接続して日時を修正

タイムスタンプを利用するコマンドが正しく処理を行うには、現在時刻を示す内蔵時計（RTC）の値が正確でなければならない。しかし、パソコンの内蔵時計は意外に精度が低く、1日に秒単位で狂いが生ずることもある。これをほうっておくと、メールの日時が狂ったり、複数マシンでファイルを共有する際にmakeの動作がおかしくなったりする。。

この問題を解決するには、現在の正確な日時を提供するNTP（Network Time Protocol）サーバに定期的に接続し、各マシンの内蔵時計を同期させればいい。接続用ソフトとしてはNTP（http://www.ntp.org/）が定番だ。NTPは、外部のNTPサーバから取得した日時を、NTPサーバとして他のマシンに提供できる。複数のマシンを使っている場合は、そのうちの1台をNTPサーバとし、ほかのマシンはそのサーバから日時を取得するとよいだろう。「桜時計」などWindows用のNTPクライアントもある。

- (1) 指定したファイルの3種類の時間情報をまとめて出力するスクリプトlstimeを作成する
- (2) 指定したディレクトリ以下のファイルの中から、最近n日以内に作成・更新したファイルを検索し、新しい順に並べて出力するスクリプトnewfilesを作成する

後半は毎回テーマを絞り、それを実現する方法を説明する。今月のお題は、

(1) 指定したファイルの3種類の時間情報をまとめて出力するスクリプトlstimeを作成する
(2) 指定したディレクトリ以下のファイルの中から、最近n日以内に作成・更新したファイルを検索し、新しい順に並べて出力するスクリプトnewfilesを作成する

というもの。いずれも、ディレクトリ階層をたどってファイル検索を行うfindを利用する。

findは、検索開始ディレクトリ、検索条件（判別式）、検索ファイルの処理（アクション）を「find ディレクトリ 判別式 アクション」という形で指定する。

これまでの連載では取り上げなかったが、findにはファイルの時間情報を表示するアクションや、時間情報を利用した判別式が用意されている。今回はこれらを利用して、時間情報の表示を行うスクリプトと、時間情報を利用した検索を行うスクリプトを作成しよう。

●findで時間情報を表示する

3種類の時間情報をまとめて表示するには、findのアクションとして-printfを使用する必要がある。-printfは、「-printf フォーマット」という書式で使用する。フォーマット部分には、ファイル名に置換される「%p」などの書式指定子や、改行を表わす「\n」などのエスケープ文字を利用できる。

たとえば、

▼リスト1 スクリプトlstime

```
1: #!/bin/sh
2: if [ -z "$1" ]; then
3:   echo "usage: lstime FILES..." > /dev/stderr
4:   exit 1
5: fi
6: find "$@" -maxdepth 0 -printf 'mtime %Tc\natime %Ac
%p\nctime %Cc\n\n'
```

```
$ find . -printf '%s %p\n'
```

とすると、カレントディレクトリ以下のすべてのファイルとディレクトリを対象として、「5814 ./hoge.txt」のようにファイルサイズとファイル名が出力される。フォーマットの末尾に改行（\n）を指定していることに注意されたい。これを忘れると、すべての出力が1行に繋がって表示されてしまうのだ。

今回のスクリプトを作成するのに利用する時間情報に関する書式指定子は、「%T」（修正時間）、「%A」（アクセス時間）、「%C」（ステータス変更時間）の3つ。いずれも、時間の出力形式を表わす1文字の英字・記号（strftime）をその直後に指定する。

たとえば、修正時間を表示する「%T」に、現在のロケールによる日付・時刻表記を意味する「c」を付けて、

```
$ find . -printf '%Tc %p\n'
```

とすると、「Sat Sep 9 14:30:28 2000 ./hoge.txt」のように、修正時間とファイル名が表示される。

日付と時刻をすべて数字で表示するには、日付の数字表記を意味する「D」と、時刻の数字表記を意味する「X」を使えばいい。ただし、「%TDX」とまとめて書くことはできず、「%TD %TX」のように1つずつ「%T」と組み合わせる必要がある。

たとえば、

```
$ find . -printf '%TD %TX %p\n'
```

とすると、「09/09/00 14:30:58 ./hoge.txt」という形で修正時間とファイル名が表示される。

●スクリプトlstimeを作成する

findの-printfを使って3種類の時間情報を表示するスクリプトlstimeを**リスト1**に示す。

このスクリプトに、

```
$ chmod +x lstime
```

として実行属性を付加すると、「./lstime ファイル名」という書式で実行できるようになる。

たとえば、引数に「hoge.txt」を指定すると、

```
$ ./lstime hoge.txt
mtime Thu Jul 20 00:44:11 2000
atime Wed Jul 26 03:01:54 2000  hoge.txt
ctime Thu Jul 20 00:44:11 2000
```

のように、1つのファイルにつき4行の情報が出力される（4行目は空行）。これは、3種類の時間情報を縦に並べたほうがお互いを比較しやすいからだ。もちろん、複数のファイル名を指定すると、それらの情報を順番に表示する。

短いスクリプトなので、内容を詳しく見ていこう。2〜5行目は、引数を1つも指定しないで実行した場合に、使い方を標準エラー出力（通常は端末画面）に表示して、スクリプトから抜ける処理をしている。

スクリプトの実体は6行目のfindだけだ。まず、検索開始ディレクトリを記述する部分に、「"$@"」が書かれている。これは、「スクリプト実行時のすべてのコマンドライン引数を、1つずつダブルクォート（"）で囲んで、スペースを区切って並べた」ものと同じだ。

それに続く-maxdepthは、find全体の動作を変更するオプションで、ディレクトリをたどる深さの最大値を指定する。ここでは「-maxdepth 0」なので、検索開始ディレクトリで指定された内容（ディレクトリではなく通常のファイルでも構わない）のみを、この後で指定するアクションの対象にすることを意味する。

判別式は含まれていないので、引数で指定したすべてのファイルについて、アクションである-printfが実行され、さきほどの例のような4行分の情報を表示する。

フォーマットの内容のうち、出力の1行目にあたるのが「mtime %Tc\n」で、文字列「mtime 」に続けて、修正時間（%T）がロケールによる日付・時刻表記（c）で表示される。以下、日時の表記はすべて同じ形式だ。

出力の2行目に相当する部分は「atime %Ac %p\n」で、文字列「atime 」に続けて、アクセス時間（%A）が表示され、2つスペースを挟んでファイル名（%p）が表示される。同様に、3行目にあたるのは「ctime %Cc\n」で、ステータス変更時間（%C）が表示される。4行目は「\n」のみで、次のファイルの情報との区切りとなる空行が表示される。

●findで時間情報を利用した検索を行う

findでは、3種類の時間情報をファイルの検索自体に利用することも可能だ。たとえば、最近1週間以内に作成・修正されたファイルや、1カ月以上アクセスされていないファイルを検索できる。

こうした検索を行うには、判別式として-mtime、-atime、-ctimeを使う。時間情報の名称と同じなので覚えやすいだろう。判別式の直後には、判断基準となる日数を±の符号付き整数で指定する。

たとえば、ちょうど1日（24時間）前から現時点までに作成・修正されたファイルをカレントディレクトリ以下から検索するには、-mtimeを利用して、

```
$ find . -mtime -1 -print
```

とすればいい。1週間以内ならば「-7」だ。

また、最後にアクセスされた日時が、現在より1カ月（ここでは30日としよう）以上前のファイルを/usr/binから探すには、-atimeを利用して、

```
$ find . -atime +30 -print
```

とする。

厳密にいうと、どちらの場合も指定した日数×24時間ちょうど前の日時のファイルは対象にならないので、「以前」や「以下」とは言えない。もっとも、実際には秒数まで完全に一致するファイルが存在することはほとんどないと思われるので、「以上」「以下」と考えても差し支えないだろう。

また、比較する時間の単位をもっと細かくしたい場合は、判別式として-mmin、-amin、-cminを利用する。これらは、名前の一部に「min」が含まれることからもわかるように、判断基準となる分数を±の符号付き整数で指定する。符号の意味は-mtimeなどと同じだ。

たとえば、今から10分以内に作成・修正されたファイルをカレントディレクトリ以下から探すには、

```
$ find . -mmin -10 -print
```

とすればいい。

●スクリプトnewfilesを作成する。

指定したディレクトリ以下のファイルの中から、1週間以内に作成・更新したファイルを検索し、新しい順に並べて

出力するスクリプトnewfilesをリスト2に示す。このスクリプトに、

```
$ chmod +x newfiles
```

として実行属性を付加すると、「./newfiles -日数 ディレクトリ」という書式で実行できるようになる。ディレクトリは複数指定したり、省略することも可能だ（省略時はカレントディレクトリが対象となる）。

たとえば、1週間以内に作成・更新されたファイルをカレントディレクトリ以下から検索するには、

```
$ ./newfiles -7
```

とすればいい。新しいファイルから順に、lsの-lオプションによるファイル情報が出力される（**画面1**）。

また、/home/samba以下のファイルを対象として、1カ月以内に作成・更新されたファイルを検索するには、

```
$ ./newfiles -30 /home/samba
```

とする。最新のファイル5個だけ表示したいなら、ファイルの先頭からオプションで指定した行数だけ出力するheadをパイプで接続すればいい。

```
$ ./newfiles -30 /home/samba | head -5
```

それでは、スクリプトの内容を見ていこう。2〜5行目は、引数を1つも指定しないで実行した場合に、使い方を標準エラー出力（通常は端末画面）に表示して、スクリプトから抜ける処理をしている。

6行目では、日数として利用する最初の引数をシェル変数dayに格納し、7行目のshiftで2番目以降の引数の内容を1つ前に詰めている。

スクリプトの実体は8行目のfindと、xargsにより実行されるlsだけだ。まず、検索開始ディレクトリの位置には「"$@"」が書かれており、実行時の引数（shiftされているため最初の引数を除く）の並びで置換される。

続いて、判別式-typeにより、検索されるファイルの種類が限定される。ここでは「-type f」なので通常のファイルだけが対象となる。また、次の判別式-mtimeでは、引数としてシェル変数dayに格納された「-日数」が使われるので、指定した日数だけ前から今までに作成・修正されたファイルのみが対象となる。

さらに、アクションとして-printではなく-print0（ゼロ）が使われていることに注意されたい。-printとの違いは、出力するファイル名の区切り文字として、改行ではなくヌル文字を使用することだ。-printでは、ファイル名の一部に空白が含まれている場合、ファイル名の区切りと誤認され、lsで表示する際にエラーメッセージが表示される。一方、-print0の場合は、ファイル名に絶対含まれないヌル文字が区切りなので、空白などを含んでいるファイル名も問題なく扱えるのだ。

findにより出力されたファイル名のデータは、パイプで接続されたxargsに渡され、lsのコマンドラインとして並べられる。xargsには、入力がなかった場合にコマンド（ここではls）を起動しない-rオプションと、ヌル文字を入力データの区切りとして扱う-0（ゼロ）オプションが指定されている。

最後に、xargsによりlsが起動され、-l、-tオプションにより、タイムスタンプが新しいものから順に、ファイル情報が表示される。なお、末尾の「2>/dev/null」は、エラーメッセージが画面に表示されるのを抑制する仕掛けで、書き込んだ内容が即座に捨てられる/dev/nullに標準エラー出力をリダイレクトする。

▼リスト2 スクリプトnewfiles

```
1: #!/bin/bash
2: if [ -z "$1" ]; then
3:   echo "usage: newfiles -DAY [DIR...]" >
/dev/stderr
4:   exit 1
5: fi
6: day=$1
7: shift
8: find "$@" -type f -mtime $day -print0 | xargs -r0
ls -lt 2>/dev/null
```

画面1 タイムスタンプの新しいファイルから順に表示

Linux日記

第13回　名前解決（6）

6回も続いた名前解決のお話も今回が最後です。メールやWebなど、ふだん何気なく使っているインターネットアプリケーションがどのように名前解決をしていたか、イメージがつかめたでしょうか。

文：榊 正憲
Text : Masanori Sakaki

illustration ; Aki

ここしばらく、紙テープリワインダだとかタイガー計算器だとか、古い物をいろいろいじっていた。紙テープのリワインダは部屋で発掘し、タイガー計算器はインターネットのオークションで入手したのだが、こんどは親戚の叔母さんから古い物をもらった。英文タイプライタである。これもPCの普及により見ることのなくなった機械である。PCの普及によって姿を消した事務機器は数多くあるが、英文タイプライタはかなり早い段階でその座を奪われたものだろう。なにせ、8ビットコンピュータとプリンタの組み合わせで置き換えることができたのだから。

1980年頃までは、何かと英文を扱う仕事をしている人、あるいは大学生などは、結構タイプライタを持っていたものである。筆者は持っていなかったが、それでもいじったことはあるし、大学生協にも商品として置いてあったような記憶がある。

20年ぶりに手動式のタイプライタをいじってみたのだが、もはやまともに使うことはできなかった。手動式タイプライタは、キーからつながったリンク機構で活字を紙に打ちつけるという単純な仕組みで動作する。排他制御などしていないので、複数のキーを同時に押すと、活字が絡まってしまう。また、機械仕掛けによって活字を打っているため、キーストロークが異様に長い。2cm近くキーを押さなければならない。押す力にむらがあると、それが文字の濃淡になってしまう。このキーがまた、段差1cm程度の階段状に並んでいる。

それでふと思い出した。昔はタイプライタの教則本というのがあったっけ。キーボードに向かう姿勢は、肘をおおよそ直角にして、てのひらは机に付けずに宙に浮かすといったことから始まっていたと思う。現在のコンピュータ用キーボードをおよそこんな姿勢で打つことはないが（今でも正しい姿勢はこれらしいが）、手動タイプライタは、まさにこの姿勢でないとまともに使えない。ついで、指をホームポジションに置いて各文字を正しくタイプするという練習だったと思う。小指や薬指は力が入らないので、どうしても薄くなりがちである。どの文字も一定の濃度で、一定のリズムで、そしてもちろん活字を絡ませないように打つというのがタイプライタの練習であった。手動式タイプライタは、人間側の技能をかなり要求する機械だったのである。

筆者自身は手動式英文タイプライタはほとんど使わなかったが、電動タイプライタはかなり使った。これは文字の印字をモーターや電磁石で行うものだ。電動タイプライタは高級な事務機器で、操作感も抜群であった。印字タイミングは電気仕掛けや機械仕掛けできちんと制御されているので、むらもなければ絡むこともなかった。そして何より、キータッチがよかった。適度なストローク（5mmくらいだっただろうか）があり、押すとだんだん重くなり、タイプの瞬間にそれが軽くなってから底に当たる。今でもこのタイプのキータッチが一番好きで、IBMのキー

写真1　ASR-33テレタイプの印字部

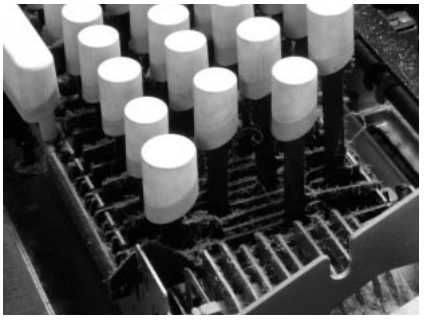


写真2　ASR-33テレタイプのキーボード

写真3　手動式タイプライタOlivetti Lettera 32

ボードを愛用している（もちろん、日本アイ・ビー・エムのではなく、US版である）。

電動タイプライタは電気的に制御できたので、コンピュータの端末としても使われていた（筆者が電動タイプライタに触れたのはもっぱらこっち方面だった）。通常のタイプバー式タイプライタは毎秒10文字が速度の上限だったのだが、IBMは14文字打てる製品を出していた。これはボール状のタイプヘッドに活字を並べ、このボールがくるくる回りながら文字をタイプするというタイプライタで、確かセレクトリックタイプという名称だったと思う（一般にはゴルフボールタイプライタと呼ばれていた）。「謎の円盤UFO」というドラマで、オープニングで文字をタイプしているシーンに使われているタイプライタといえば、覚えている人もいるかもしれない。これはタイプバー式より高速ということもあり、コンピュータの端末としても数多く使われていた。筆者もジャンク屋で仕入れて家で遊んでいた（もちろん、「ＵＦＯ… Unidentified Flying Object」などとタイプして遊んでいたのである）。

セレクトリックタイプのキータッチもすばらしかったが、ASR-33テレタイプのキータッチも結構好きだった。1970年代の多くのミニコンは、ASR-33を標準コンソールとして使っていた。ASR-33は110bps全二重端末で、キーボード／プリンタ、紙テープリーダ／パンチャを備えた便利な機器である（今でもtermcapファイルなどにその名前が残っていることがある）。この機械の驚くべき点は、シリアル－パラレル変換、文字のデコード／エンコードなどをすべてモーターを動力とする機械仕掛けで行っていた点である。

ASR-33はさほどいいキータッチというわけではなかったが、連続打鍵したときに、指が吸い込まれる感覚があり、これが結構好きだった。慣れると、この感覚で、秒10文字のペースでタイプしていることがわかる。実際、連続打鍵すると、モーターの回転により、それまで押せなかったキーが押せるようになるのである。そして、タイプしていると、キーを押すのではなく、指を載せたキーが沈みこんでいくように感じるのである。

パソコンが普及したことによる悲劇のひとつは、キーボードがコストダウンの対象となってしまったことだ。かつて（メインフレームやミニコンの時代）、キーボードは高級なユニットであった。キータッチ、寿命などを考慮し（そして価格をあまり考えず）、最高級のスイッチを使っていた（秋葉原でパーツとして買うと、キースイッチ1個が数千円、ユニットで数万円以上した）。筆者のささやかな願いは、値段が高くなってもいいから、キータッチが良くて長持ちするキーボードを市場に提供して欲しいということである。今使っているUS IBMのキーボードは、1万3000円くらいで買った高級品であるが、かつてのキーボードに比べればまだまだである。寿命という点ではとりあえず満足しているが（というか、まだ壊れていないというだけだが）。

さて、キーボードに関する思い出話と愚痴はこれぐらいにして、ネームサーバの話の続きである。

Column

和文タイプライタ

英文タイプライタに対して、和文タイプライタという機器もかつてはよく使われていた。2000字ないし3000字の活字盤から活字を選んで和文をタイプできるというタイプライタである。これも日本語ワープロと高解像度の日本語プリンタの普及によって姿を消した（かつて、公文書などには16ドット漢字フォントは使えないなどという時代もあった。文字が略字だからそうだ。24ドットフォントならよかったらしい）。和文タイプライタのミソは、縦書き文書も横書き文書も作成できたということである。

筆者がかつて某A社に勤めていた頃、辞令だけは最後まで和文タイプライタで打ったものをもらっていた。そう考えると、1990年代初頭まで、和文タイプライタは現役だったことになる。

写真5　IBM モデル82のキーボード

写真4　電動タイプライタ IBM モデル82

■ダイナミックDNS

ゾーンデータベースは、基本的にマスタネームサーバ上のデータベースファイルという形で定義され、これを更新するのは人間であるシステム管理者である。このデータベースでドメイン名とIPアドレスを対応付けるAレコード、PTRレコードは、静的なデータを登録するものであり、ドメイン名とIPアドレスは、システム構成を変更したりしない限り、常に一定の対応付けであることを前提としている。UNIXマシンが高価なホストコンピュータやワークステーションであり、何人ものユーザーが共同利用していた時期はこれでまったく問題はなかったのだが、今はそうもいっていられない。

多数のコンピュータのIPアドレスをきちんと管理するために、今はDHCPが広く使われている。DHCPを使えば、割り当てるIPアドレスをサーバ上で一元管理することが可能になり、IPアドレスの衝突などによるトラブルを回避できる。しかし、DHCPを使ってホストにIPアドレスを動的に割り当てた場合、DNSによる名前解決ができなくなるという問題がある。

DHCPは、多数のコンピュータでいちいちTCP/IPの設定をするという手間を省いてくれるが、その代わり、各コンピュータのIPアドレスが固定的ではなくなってしまう。DHCPサーバは動的割り当て用のIPアドレスプールを管理しており、アドレス割り当てのリクエストを受けると、アドレスプールから未使用のアドレスを選んで割り当てる。つまり、DHCPクライアントのIPアドレスは、アドレスプールで定義された範囲内にあることは確かだが、ブートするたびに変わってしまう可能性があるわけだ。このように動的にアドレスが割り当てられたコンピュータのドメイン名は、DNSに登録することができなかった。前に書いたように、DNSのデータベースは人間が更新する静的なものだからだ。たとえばアドレスプールの一群のIPアドレスにdhcp1、dhcp2といった名前を定義しておき、ネームサーバの参照時にこういった名前を返すことはできる。しかし、こういった名前は、そのホストがDHCPクライアントであるということを意味するだけで、特定のホストを識別するという意味合いはまったくなくなっているのだ。

実際問題として、DHCPでアドレス割り当てを行うのは、一般にクライアント（ワークステーション）コンピュータなので、別のマシンからそのホスト名を指定してサービスのリクエストを受けることはほとんどないし、外部から特定のワークステーションに対する名前解決リクエストがくることもあまりないのだが（当然のことながら、外部からアクセスされるサーバのアドレスは固定にしておく）、ちょっとtelnetでつないで作業をしたいといったことがある。また、各種ネットワークコマンドなどで、ドメイン名が参照されることもある。やはり、このようなアドレスが動的に変化するホストについても、DNSで名前解決が行えるに越したことはない。

これを解決するのがダイナミックDNSである（BIND 8.XはダイナミックDNSをサポートしている）。DNSのデータベースは、マスタサーバのデータベースファイルに基づくものだが、それに加えて、クライアントからのリクエストに応じて、データを動的に登録できるようにしたのである。DHCPでアドレス割り当てを受けたクライアントのドメイン名とIPアドレスを、ネームサーバに登録するのである。以後ネームサーバは、静的に登録されたゾーン情報に加えて、動的に登録された情報についても名前解決を行えるようになる。動的に登録された名前を解決

Column

CTRL－R

UNIXでシェルなどを使っている時の特殊なキー割り当てに、行を再表示するreprintという機能がある。これには通常CTRL－Rが割り当てられている。今ではあまりお世話になることはないが、印字式端末を使っていた頃には非常に重要なキーであった。

印字式端末では、BackSpaceやDeleteをタイプしても、文字は消えない（高級なタイプライタでは、ホワイトで文字を消せるものもあったが）。そのため、ミスタイプした文字がそのまま残ってしまう。そのため、何度も修正した行は、いったい何が入力されたのかわからなくなってしまう。そこでCTRL－Rの出番である。CTRL－Rをタイプすると、入力した文字が再表示されるのである。

するように求められれば、ネームサーバは、動的に登録されたIPアドレスを返す。この仕組みにより、動的に割り当てられるIPアドレスについても、DNSだけで名前解決を行えるようになるわけだ。

ところで、ネームサーバが複数ある場合（スレーブサーバも存在している場合）、動的な登録はどのように処理されるのだろうか？　スレーブサーバは、マスタサーバのゾーンデータのコピーを持っている。動的な登録に対する名前解決をすべてのネームサーバで行うためには、動的な登録はマスタサーバに対して行われる必要がある。しかしクライアント側は、どのネームサーバがマスタであるかという情報は持っていない。従って、スレーブサーバが動的な登録のリクエストを受けることもあるわけだ。この場合、スレーブサーバは、登録された情報をマスタサーバに転送する。マスタサーバはこのデータを登録し、ほかのスレーブサーバにも通知する。これにより、どれかひとつのネームサーバに登録すれば、ほかのネームサーバにも情報が伝搬される（**図1**）。

BIND 8.Xでは、デフォルトでは動的更新は禁止されている。これを可能にするには、named.conf中のzone文ブロック中で、allow-updateを指定し、そして更新リクエストを受理するIPアドレスを記述する。

動的な登録機能を確かめてみるには、nsupdateコマンドを使えばよい。

■プライベートIPアドレスセグメント

ある程度の規模のネットワークを実際にインターネットに接続した場合、すべてのホストにグローバルIPアドレスを割り当てることはまず不可能だ。そんなにグローバルアドレスを割り当ててもらえないからだ。普通は、インターネット接続用のルータ、外部からアクセスされるネームサーバ、メールサーバ、WebサーバなどにだけグローバルIPアドレスを割り当て、残りのホスト（組織内向けのサーバやクライアント）にはプライベートIPアドレスを割り当てることになる。そして、グローバルIPアドレスとプライベートIPアドレスの変換を行うためにIPマスカレード（NATデーモン）を動かしたり、アプリケーションゲートウェイ（プロキシサーバ）を置くことになるだろう。このようなゲートウェイホストは、グローバルIPアドレスとプライベートIPアドレスを持っている。また、世の中物騒になっているので、何らかのファイアウォールも用意することになるだろう（というか、NATやプロキシサーバはファイアウォールの一部として組み込まれる形になる）。

さて、プライベートIPアドレスを使ったセグメントがある場合、ネームサーバをどのように設定すればいいのか？　メールサーバやWebサーバと共に、サイト内のドメイン名をすべて解決できるように設定するのは問題がある。サイト内のホストの多くはプライベートIPアドレスを使っているので、インターネットにおいて無効なアドレスが返されてしまう。もっとも、サイト内のプライベートIPアドレスのホストは外部に対してサービスを提供していないはずなので、名前解決リクエストが正しい結果を返さなくても実害はない。だが、それでも美しくないものは美しくない。すっきり解決するために、ネームサーバに求められる条件は次のようになるだろう。

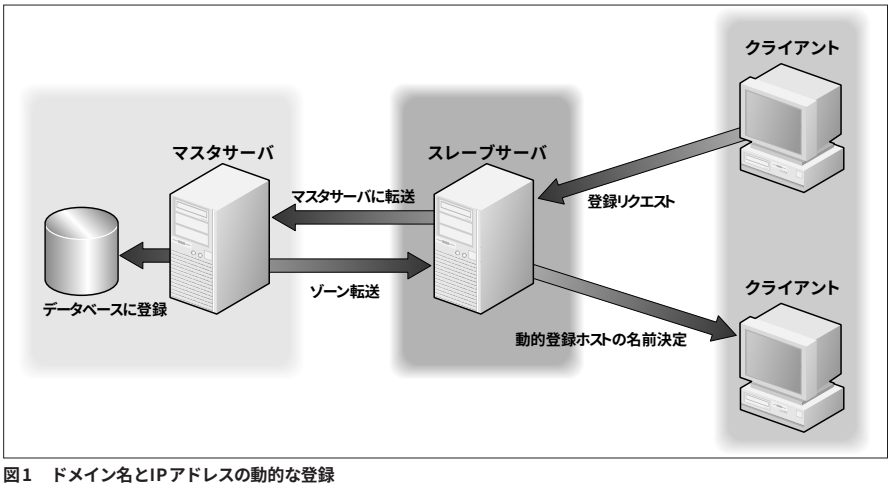


図1　ドメイン名とIPアドレスの動的な登録

Column

Windows 2000とダイナミックDNS

ネットワーク上にWindows 2000（特にServer）があると、起動時に動的な登録リクエストがやたらと送られてくる。Windows 2000のネットワークは、各種サービスにDNSを多用しているので、動的登録しないようにすればいいというわけにもいかないことがある。かといって、組織の玄関にあたるネームサーバに組織内でプライベートに使用するレコードを多数登録するわけにもいかない。ネットワーク上にWindows 2000 Serverがある場合は、いろいろ考えなければならないのだ。

- **サイト外からのメールサーバ、Webサーバなどに対する名前解決リクエストを処理できる。**
- **サイト内のホストにも、（db-svr.linux-mag.ascii.co.jpというような）一貫性のあるドメイン名を使用したい。**
- **サイト外からは、サイト内のプライベートなドメイン名の解決を行えない（未知のドメイン名というエラーを返す）。**
- **サイト内では、ネームサーバを使ってプライベートなドメイン名の解決を行える。**
- **サイト内から、自由にインターネット上のグローバルドメイン名の解決を行える。**

どのようにすればこのような条件を実現できるだろうか？　セキュリティ対策も含めて、さまざまな条件に対していろいろな実現方法（そして欠点）があるのだが、詳細についてはバッタの本やファイアウォールの解説書を参照してほしい。ここではひとつの方法を紹介するにとどめる。

まず、前提条件を示そう。外部からアクセス可能な各種サーバはグローバルIPアドレスを持ち、XXX.ascii.co.jpというドメイン名を持つとする。それに対して、サイト内でのみアクセスされるプライベートIPアドレスのホストは、XXX.SUBDOMAIN.ascii.co.jpという形式にする（**図2**）。

まず、グローバルIPアドレスを持つネームサーバ、ns1.ascii.co.jpを用意する。これは上位ドメインからascii.co.jpドメインの名前解決を委任された権威のあるネームサーバで、XXX.ascii.co.jpというドメイン名を解決する。このネームサーバには外部から名前解決リクエストが届き、そしてメールサーバ、Webサーバなどの名前解決を行う。これらのサーバもグローバルIPアドレスを持っているので、外部からアクセスできる。

ascii.co.jpにサブドメインがあり、それが別のゾーンである場合、普通ならns1が別のネームサーバ（たとえば、ns2.SUBDOMAIN.ascii.co.jp）にそのゾーンの管理を委任する。しかしこのような委任を行うと、外部からサブドメインに対する名前解決リクエストが来たときにns2を紹介してしまい、その結果、ns2からプライベートIPアドレスを返す応答が送られてしまう。また、ns2がプライベートアドレスを使っている場合、ns2そのものにアクセスできないというエラーが発生してしまう。名前解決リクエストに対してプライベートIPアドレスを返したり、ネームサーバにアクセスできないという事態を避けるために、ns1はこのような委任を行わない。これにより、外部からはサブドメインに関する名前解決を行えなくなる。

グローバルIPアドレスセグメントとプライベートIPアドレスセグメントは、IPマスカレードが動いているゲートウェイで接続されているとしよう。そして、プライベートセグメント内に、ネームサーバns2.SUBDOMAIN.ascii.co.jpがある。ns2は、SUBDOMAIN.ascii.co.jpゾーンを管轄するネームサーバで、サイト内からの名前解決リクエストを処理する。これにより、サイト内のサブドメインのホストは、ns2を使ってサブドメイン中の名前解決を行える。ns2はns1から委任されていないので、外部から名前解決リクエストが来ることはない。

最後の条件は、サイト内のホストはインターネット上の名前解決をどのように行うかということだ。ひとつの答

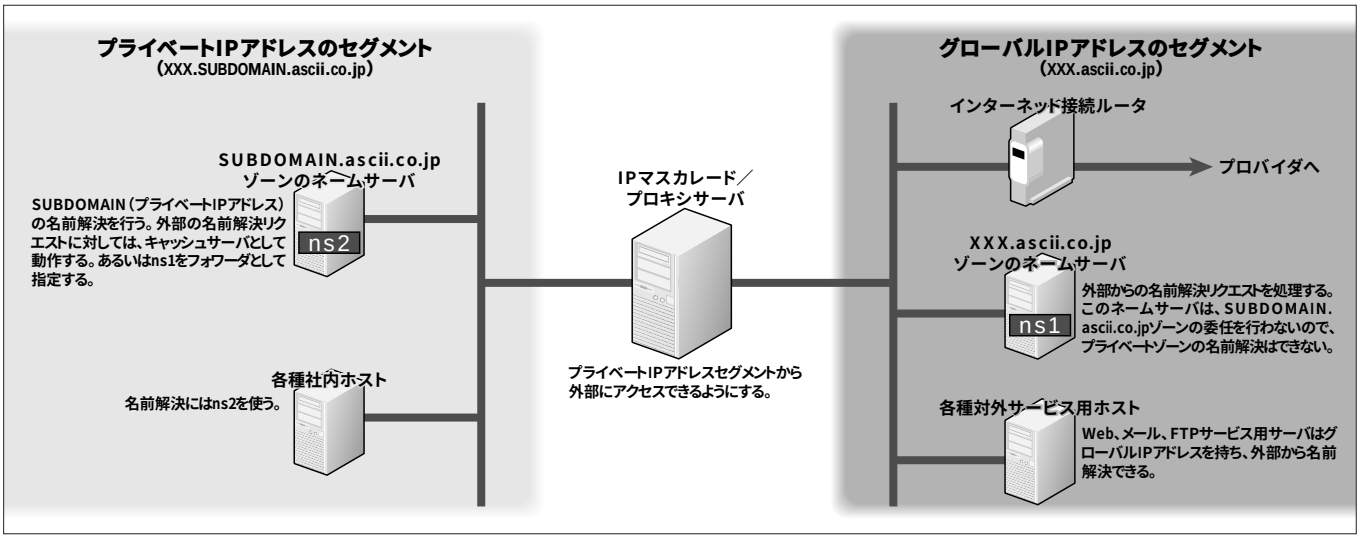


図2　プライベートなサブドメインを使う

えは、特に何もしないというものだ。ネームサーバは、自身が管轄するゾーン以外のドメイン名については、キャッシュサーバとして動作する。つまり、ルートネームサーバのヒント情報さえ持っていれば、ルートネームサーバから始めて、インターネット上のいくつかのネームサーバに問い合わせ、任意のインターネットドメイン名を解決できるのである。

もうひとつの方法は、ns1をフォワーダに指定するというものだ。一般にフォワーダ指定は負荷分散に使われるが、マスタサーバやスレーブサーバでもフォワーダを指定することができる。この場合ネームサーバは、自身が管轄するゾーンについては自身のデータベースから応答を返し、それ以外のドメイン名については、フォワーダとして指定した別のネームサーバにリクエストを中継する。そしてフォワーダから返された応答をクライアントに返すのである。この場合、ns1をフォワーダに指定すればそれで済む。ns2は、サブドメインについては自身のデータベースで名前解決を行い、それ以外のドメイン名については、ns1に解決してもらうのである。

■ネームサーバのセキュリティ

専用回線などでインターネットに常時接続しているサイトで、自前のネームサーバを組織内に置いた場合、つまり、組織が独自のドメイン名を持ち、そのドメインのネームサーバとして組織内のサーバがJPNICやInterNICに登録されている場合、当然のことであるが、そのネームサーバには外部からのアクセスがある。そのサイト宛のメールを送るためには、MXレコードと実際のメールサーバを参照するためのリクエストが来るだろうし、Webサーバを運用していれば、たとえWebサーバ自体はプロバイダ側にあったとしても、名前解決リクエストはサイト内のネームサーバにやってくる。

こういった、本来のネームサーバの用途であるリクエストが来ることは問題ない。しかし、世間にはよからぬ輩が多数いる。クラッカーという連中だ。ネームサーバに限らず、外部からのアクセスがある、あるいは外部からアクセス可能なサーバがある場合、セキュリティ対策を講じておく必要がある（いやな世の中だ）。ネームサーバの場合、ゾーン転送機能が悪用される可能性がある。プロバイダのネームサーバなど、多数の組織のドメイン名情報などを収めているネームサーバからゾーン転送を行えば、どのようなドメイン名やホストがあるのかといったことがすべてわかってしまう。そのため現在では、指定した相手（すなわち正規のスレーブサーバ）以外はゾーン転送できないように設定するのが普通だ。たとえば以下のようにゾーン定義中に記述することにより、192.1.2.1以外の相手に対してゾーン転送を禁止することができる。

```
zone "linux-mag.ascii.co.jp" {
        type master;
        file "linux-mag.db";
        allow-transfer {
                192.1.2.1;
        };
};
```

このほかにも、問い合わせを受理する相手を制限したりすることも可能だ。これを利用して、前に説明したプライベートセグメントの問題を解決することができる。たとえば192.1.1.0／24というグローバルセグメントにあるns.ascii.co.jpが、ascii.co.jpゾーンとSUBDOMAIN.ascii.co.jpゾーン（アドレスは10.1.1.0／24）のマスタサーバだとしよう。ascii.co.jpゾーンは対外的に公開されており、アクセス制限はない。それに対してSUBDOMAIN.ascii.co.jpはプライベートセグメントであり、サイト内からのアクセスのみを許可するといった場合、ns上で次のようにSUBDOMAIN.ascii.co.jpゾーンを設定すればよい。

```
zone "SUBDOMAIN.ascii.co.jp" {
        type master;
        file "SUBDOMAIN.db";
        allow-query {
                192.1.1/24;
                10.1.1/24;
        };
};
```

これにより、SUBDOMAIN.ascii.co.

Column

やはりハッカーではなくクラッカーと呼ぼう

世間（特にマスコミ）は、システムに違法侵入して悪事を働いたり、ウィルスを撒き散らす連中のことを相変わらず「ハッカー」と呼んでいるが、やはりこれは「クラッカー」と呼ぶべきだろう。ハッカーというのは、伝統的にやはりハッカーなのである。今の典型的なハッカーの生態は知らないが、個人的には、ファーストフードを主食とし、夜行性で、自分の作っているソフトウェアをより良いものにすること以外にほとんど関心がないというのがハッカーのイメージである。

良くも悪くも、今のコンピュータ／ネットワークの世界は、こうした連中抜きにはあり得なかったのだから。

jpゾーンについては、サブドメイン内とascii.co.jpゾーンのホストからの問い合わせのみを受理するようになる。外部から問い合わせが来ても受理しない。

■自前のサーバを用意しなくても

サイト内に自前でネームサーバを用意するということ、つまり、専用線でプロバイダと結ばれたこちら側に、適当なオペレーティングシステムを載せた常時稼働のマシンを用意し、その上でネームサーバを運用し、外部からアクセスできるようにするというのは、はたして賢明なことなのだろうか？

単に「インターネットを使う」という面では、答えはノーだろう。

かつては、組織のLANをインターネットに常時接続するということは、ドメイン名の登録、グローバルIPアドレスの取得（現在では、IPアドレスはプロバイダから借りるという形式が普通だ）、DNSのマスタサーバとメールサーバの立ち上げという作業を行わなければならなかった。コンピュータやネットワークを業務とする組織や研究機関でもない限り、これはかなり敷居の高い作業である。大きな組織であれば、専任の要員を用意することもできるだろうが、中小の組織ではそうもいかないだろう。

現在、多くのプロバイダがネームサーバ、メールサーバ、Webサーバ、FTPサーバなどを提供するサービスを行っている。これを使えば、契約者はほとんどサーバの管理の手間をかけることなく、各種のインターネットサービスを便利に使うことができる。以前はドメイン名などにいろいろ制限があったりしたものの、現在はすべてプロバイダまかせでuser@会社名.co.jpというメールアドレスを使い、www.会社名.co.jpというWebサーバを運用することができる。ユーザーが行わなければならないことは、メールクライアントの設定、Webコンテンツの作成程度だ。日々の管理やセキュリティ対策はプロバイダまかせで済む。

サーバ類をプロバイダまかせにすることは、もうひとつ大きなメリットがある。サーバはプロバイダ側にあるので、インターネットと大容量の回線でつながっているということだ。たとえば128kbpsの回線でプロバイダに接続しているサイトにWebサーバがある場合、外部からのWebアクセスはすべてこの128kbps回線を通ることになる。アクセスが集中するサーバに対して、128kbps回線はあまりにも細い（今や128kbpsは、個人用の回線速度だ）。サーバがプロバイダ側にあれば、少なくともMbpsのオーダでインターネットに接続されているだろう。ネームサーバは、Webサーバに比べればわずかなトラフィックに過ぎないが、それでもプロバイダにまかせない理由はない。

このように、サーバ類をすべてプロバイダ側が用意し、契約者側サイトにはクライアントしかないという構成にする場合、プロバイダのネームサーバの設定はどのようになるのだろうか。ここでは、契約者のドメイン名はascii.co.jp、プロバイダのネームサーバはns.provider.ne.jp、ascii.co.jpが使う対外的なWebサーバの名前はwww.ascii.co.jp、実際にこのサービスを行うホストはwww0.provider.ne.jp、メールエクスチェンジャはmail0.provider.ne.jpとする。

リスト1～**3**を見ると、ascii.co.jpというゾーンがプロバイダのネームサー

▼リスト1　ns.provider.ne.jpのnamed.conf

```
zone "provider.ne.jp" {
        type master;
        file "provider.ne.jp.db";
};
zone "ascii.co.jp" {
        type master;
        file "ascii.co.jp.db";
};
```

▼リスト2　provider.ne.jpゾーンのデータベースファイル

```
@               IN SOA  ns.provider.ne.jp. postmaster.provider.ne.jp.  (
                        :
                        :
;       ネームサーバレコード
                IN NS   ns.provider.ne.jp.
;       ホストレコード
ns              IN A    192.168.1.1      ; ネームサーバ
mail0           IN A    192.168.1.2      ; メールサーバ
www0            IN A    192.168.1.3      ; WWWサーバ
```

▼リスト3　ascii.co.jpゾーンのデータベースファイル

```
@               IN SOA   ns.provider.ne.jp. postmaster.ascii.co.jp.  (
                         :
                         :
                IN NS          ns.provider.ne.jp.
                IN MX    10    mail0.provider.ne.jp.
www             IN A           192.168.1.3  ; WWWサーバ
```

バ上で定義されている。このゾーンは、メールサーバとWebサーバでascii.co.jpというドメイン名を使うために定義されたものだ。www.ascii.co.jpは実際にIPアドレスが割り当てられているが、これは実際にはプロバイダのWebサーバであり、そしてこのゾーンのメールサーバもプロバイダのメールサーバである。つまり、ascii.co.jpゾーンは存在しているものの、そのゾーンはまさにドメイン名を定義するためだけに存在するものであり、実際のホストはすべてプロバイダが提供しているのである。このような仕組みを実現できるのは、DNSの自由度の高さのおかげである。

■リゾルバの設定

だいぶ長くなったが、ネームサーバの設定については、このへんでおしまいにしておこう。さらに知りたければ、バッタの本を参照のこと。さて、最後に名前解決サービスのクライアント側の設定について簡単に解説しておこう。

ネームサーバに問い合わせ、各種情報を受け取る側のモジュールをリゾルバという。インターネットドメイン名を使うシステムはすべてリゾルバである。ネームサーバを使った名前解決を行うクライアントはリゾルバとしての設定を行わなければならない。少なくとも、ネームサーバのIPアドレスだけは登録しておく必要がある。

WindowsやMacintoshでTCP/IP設定を行うときに、DNSネームサーバのIPアドレスとデフォルトドメイン名を設定するが、これはまさにリゾルバ設定なのである。UNIXの場合も同じような設定項目がある。インストーラや設定ユーティリティで同じような項目を設定するが、こういったプログラムで指定した情報は、/etc/resolv.confというファイルに記録される（**リスト4**）。

resolv.confには、デフォルトドメイン名やネームサーバのIPアドレスを登録する。このファイルに設定できる項目を以下に示す。

・nameserver

そのホスト上でネームサーバプロセスが動作していない場合に、ネームサーバのIPアドレスを指定する。このホストは、名前解決の際に、リクエストをこのIPアドレスに送る。nameserverを指定しなかった場合はローカルホスト上でネームサーバが動作しているものとみなす（名前解決リクエストをローカルホストに送る）。

・domain

省略されたドメイン名を指定したときに、後ろに付加するドメイン名を指定する。たとえばascii.co.jpゾーンで「domain ascii.co.jp」という指定を行っている場合、単に「ns」と指定すると、リゾルバは「ns.ascii.co.jp」というドメイン名の解決を求める。指定されたドメイン名が省略形かフルドメイン名かどうかは、名前に含まれているドットの数で判定する。デフォルトでは、ドットが1つも含まれていないドメイン名を省略形とみなす。

・search

domainと同様にデフォルトドメイン名を指定するが、domainが1つしか名前を指定できないのに対して、searchは複数の名前を指定できる。これにより、いくつかのデフォルトドメインを付加した名前を順番に検索できる。

・sortlist

問い合わせに対して複数のアドレスが返された場合に、どのネットワークアドレスを使用するかを指定する。相手までの到達経路が複数あり、特定のネットワークインターフェイスを使いたい場合などに指定する。

・options

ネームサーバのデバッグオプション（ここでは解説しない）と、フルドメイン名とみなすためのドットの数を指定する。デフォルトではドメイン名にドットが1つ以上含まれている場合はフルドメイン名とみなすが、「options ndots:2」というように指定すると、ドットが2つ以上含まれているドメイン名をフルドメイン名とみなす。この場合、たとえば「ns.linux-mag」と指定した場合は省略形と判断され、「ns.linux-mag.ascii.co.jp」という名前を解決することになる。

▼リスト4　resolv.confの例

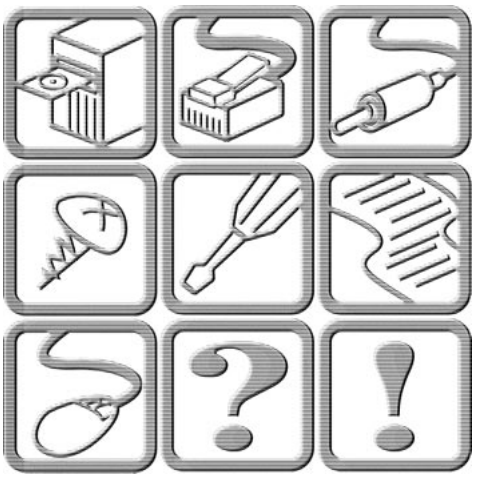
```
nameserver 192.168.1.1
search linux-mag.ascii.co.jp ascii.co.jp
```

Column

host.conf

resolv.confはDNSを使って名前解決を行う際の設定ファイルだが、それ以前に、名前解決にDNSを使うという設定が必要だ。UNIXには/etc/hostsという静的な名前とアドレスの対応付けを記述するファイルがあり、名前解決にDNSとhostsファイルのどちらを使うかを指定しなければならない。これは/etc/host.confというファイルで指定する。

Try & Try [trái] [trái]



ネットワークパケットフィルタリング機能 Netfilterフレームワーク　その1

文：政久忠由
Text : Tadayoshi Masahisa

カーネル2.4で実装されたNetfilterフレームワークは、従来のカーネルに実装されていたIPパケットフィルタリングやNAT、IPマスカレード、ポートフォワード、透過プロキシなどの機能を統合し、さらに機能を拡張しつつシンプルに実装し直した機構である。このNetfilterフレームワークにより、ソースコード的にも複雑さ、煩雑さが解消され、またユーザーが行う設定の面でもわかりやすいものとなっている。

主観的にはLinuxカーネル2.4は、特にこれといった目新しい機能が追加されているわけではないが、このNetfilterフレームワークをはじめ、ディスクI/O、ネットワークI/O、さらに内部構造にわたり見直され、全面的にオーバーホールしたものといえる。これは従来、機能を追加する過程で場当たり的にパッチを当てるなどの手法で拡張してきたために、さまざまな部分が複雑に入り組んだ状態、ある意味、不細工で汚い状態に陥ってしまったカーネルソースコードを再構築し、コンセプトはそのままに本来のLinuxカーネルの真骨頂であるシンプルな構成として再実装したとも言い換えられる。その中でもカーネル2.4では、ネットワーク部がもっともブラッシュアップされた部分なのである。

フィルタリングポイント

まずはNetfilterフレームワークでフィルタを仕掛けることができるポイントを説明しておこう。Netfilterフレームワークでは、フィルタを次の5カ所に設定できるようになっている。

- NF_IP_PRE_ROUTING
- NF_IP_LOCAL_IN
- NF_IP_FORWARD
- NF_IP_POST_ROUTING
- NF_IP_LOCAL_OUT

各インターフェイスに入ってきたパケットをルーティングする前に処理するNF_IP_PRE_ROUTING、次にルーティング処理後、ローカルに入ってくるパケットに対して行うNF_IP_LOCAL_IN、別のインターフェイスに送るパケットに対して行うNF_IP_FORWARD、そして最終的に各インターフェイスから出て行く際に行うNF_IP_POST_ROUTINGである。加えてローカルで生成されたパケットに対してフィルタリングを行うNF_IP_LOCAL_OUTの5つのフィルタリングポイントが実装されている。もちろんこれらはパケットの流れを表すもので、内部とか外部というネットワークのレイアウトには関係していない。つまり、特別にインターフェイスを指定しない限り、すべてのインターフェイスに影響するようになっている。

各フィルタリングポイントには、フック機構が実装され

ていて、それを利用して通過するパケットをフィルタリング処理し、設定に応じてパケット、またその内容を操作できるように設計されている。特にPRE_ROUTINGやPOST_ROUTINGポイントでは、IPパケットレベル以外にMACアドレスなどに対してチェックを行うことも可能だ。つまりEthernetフレームなどの段階のヘッダ情報もチェックできるというわけだ。

フィルタリングルールを設計する際には、どの位置にフィルタを設置するかは極めて重要になる。というのも、その位置次第でセキュリティ面にもパフォーマンス面にも大きな影響を及ぼすからだ。

たとえば、通常、ルーティング処理後のパケットはどこから来たものかを知るすべがない。そのため出口やフォワードの位置にフィルタを設置していたとしても、インターネットのような外部ネットワークから入ってきたパケットが送信元アドレスを偽っていた場合（アドレススプーフィング）などに対処することができない。だから外部からの問題に対するフィルタは、できる限りインターフェイスに入ってきた段階、ルーティング処理の手前に設置し、水際で対処するようにしなければならない。しかしこの場合、通過パケットがほかのポイントに比べて多いので、負荷が増しパフォーマンスに影響を及ぼすという別の問題が生じてくる。ただし、フィルタリングルールの内容や数にもよるが、よほどの高速リンクでもない限り、Pentium CPUクラスであれば処理が飽和することはないので、あまり心配する必要はないと思う。

なおNetfilterフレームワークでは、パケットが来た、または出ていくインターフェイスも監視することができるので、実はルーティング処理後の段階であっても入ってきたインターフェイスを指定してチェックを行うことができるようになっている（NATではこの機能が不可欠だからね）。だとすると上記のような水際での対処は必要ないのだろうか？　いや、そういうものではない。フィルタリングは、いかなる場合でも処理能力が許す限り、できるだけ早い段階でのチェックが望まれる。その意味で先のPRE_ROUTINGというフィルタリングポイントは重要になってくる。一応、PRE_ROUTINGでは、入ってくるすべてのパケットがフィルタリング対象となってしまうのに対し、INPUT、FORWARDの段階ではルーティング処理による一種のフィルタにかけられたあとのパケットのみが対象ということになる（ルーティングできないものは捨てられるから）。とにかく、フィルタの内容だけでなく、その位置も目的に応じてよく考えて設計する必要があるということだ。

Netfilterフレームワークの構成

Netfilterフレームワークは、IPv4、IPv6、DECNETのそれぞれのプロトコルスタックごとに実装されている（パケットのヘッダの構成内容が違うのだから当然なのだけれど）。ここではIPv4にだけ注目することにしたい。

前回Netfilterフレームワークを有効にするためのカーネル構成方法を紹介したが、その設定でもわかるようにNetfilterフレームワークは機能別に細分化され、20数個のカーネルモジュールとして実装されている。これらの中でコアとなるモジュールはip_tablesとip_conntrackの2つだ。ip_tablesはNetfilterフレームワークのもっともジェネリックな部分で基本構成を提供するモジュールだ。ユーザースペースのユーティリティiptablesとの対話も行う。ip_conntrackは通過するパケットの状態を記録しておくためのモジュールで、NATやIPマスカレードの際のコネクション対応を管理する。またこのコネクション状態情報はフィルタリング対象としても利用できるようになっている。紙面の都合で個々のモジュールの説明はしないが、名前と機能が基本的には一致しているので、だいたいどの機能を提供するものであるかはわかると思う。

Netfilterフレームワークは、このようなモジュール構成になっていることからわかるように、非常に汎用的で拡張性に富み、容易にフィルタリングモジュールを追加できるようになっている。とはいえ、標準で必要なフィルタはほぼ実装されているため、あえて追加しなければならない機能はすぐには思いつかないのだが、特定のネットワーク攻撃を排除／監視するためのモジュールなどを作成すると設定の手間が省けて便利なのではないかと思う。

フィルタリング対象

フィルタリングの対象となるのは、一般にプロトコルヘッダと呼ばれる部分だ。前述のようにEthernetフレームのMACアドレスを対象とすることもできるが、基本的にはIPv4、TCP、UDP、ICMPなどのヘッダのフィールド情報が使用される。これらの各フィールドはすべてフィルタリングのパターンマッチ対象として利用可能である。しかし一般には、IPv4の送信元／先アドレス、プロトコルタイプ、TCPとUDPの送信元／先ポート番号、そしてTCPフラグオプションのフィールドを用いることでほとんどの要件を満たすことができる。

パケットの内容（データ部）を対象とすることもできなくはないのだけれど、Netfilterフレームワークはカーネルモードで動作するルーチンなので、常識的にあまり時間のかかる処理を実行することができない（させるわけにはいかない）。そのため通常、Virusスキャンや不正アクセスの検出などのパケット内容を処理する作業は、ユーザースペースのプロセスにて行う。Netfilterフレームワークではこれをサポートするために QUEUE というパケットをキューイングするアクションを設定することができるようになっている。

フィルタリングアクション（ターゲット）

設置したフィルタにマッチした場合のアクション（Netfilterフレームワークではターゲットと呼ぶ）には、現時点で内部的にNF_DROP、NF_ACCEPT、NF_STOLEN、NF_QUEUE、NF_REPEATの5つの処理が用意されている。だいたい名前から内容は想像できると思うが、一応簡単に説明しておくと、DROPは通過をブロックしパケットを廃棄する、ACCEPTは通過を許可する、STOLENはパケットを通過させず横取りする、QUEUEはユーザースペースプロセスのためにキューイングする、REPEATはフィルタリング処理を繰り返すためのものだ。

実際にユーザーがフィルタリングルールとして指定するのは、ACCEPT、DROP、QUEUEとRETURNである。前3つは上記の説明のとおりである。RETURNは、現在処理中のchainとして定義されているフィルタリングルールのセットから抜け、そのchainを呼び出したルールに戻るための指定だ。ビルトイン定義chainからユーザー定義chainを呼び出した場合などに利用する。

特別な場合の除き、パケットを通過させるかブロックするかのACCEPTとDROPしか使わないので、この2つを理解しておけば支障はないと思う。

拡張機能を利用した場合は、REJECT（パケットをブロックし、その際エラーメッセージを返信する。DROPではエラーを返さない）、LOG（マッチしたパケットのログを残す）、MARK（特定パケットにマーキングを行う：mangleで使用）、TOS（IPのサービスタイプを操作する：mangleで使用）、MIRROR（送信元／先アドレスを逆にして処理を続行する）といったアクションを指定することもできる。ちなみにREJECTとMIRRORはINPUT、OUTPUT、FORWARDのchainでしか使用できない。

またNAT機能を利用する場合（後述のnatテーブルでのみ指定可能）は、SNAT（送信元アドレス変換）、MASQUERADE（IPマスカレード：送信元アドレス／ポート番号変換）、REDIRECT（送信先アドレスをローカルで受け取るために変換）、DNAT（送信先アドレス変換）を指定することができる。

フィルタリングテーブル

Netfilterフレームワークでは、処理内容に応じてfilter、nat、mangleという3種類のフィルタリングテーブルを用意している。この各テーブルで設定されたフィルタリングルールがそれぞれ内部でパケットマッチングテーブルとして使用される。これらはそれぞれモジュールが異なり、iptable_＊という名前で管理されている。基幹となるip_tablesモジュールさえロードしていれば、それ以外のモジュールは必要に応じてロードされるようになっているので、個々のロードを明示的に行う必要はない。

各テーブルの説明をしておくと、filterは普通のパケットフィルタリング、natはアドレス変換やコネクションの追跡を行う場合、mangleはnetfilter markやサービスタイプの変更操作を行う処理を設定する場合に使用する。

また各テーブルではビルトインchainが若干異なる。filterには、INPUT、FORWARD、OUTPUT、natには、PREROUTING、POSTROUTING、OUTPUT、mangleには、PREROUTING、OUTPUTがそれぞれ定義されている。これらは次のコマンドで確認することができる。

```
# iptables -L（-t filterは省略可能）
# iptables -t nat -L
# iptables -t mangle -L
```

ここで注意して欲しいのは、上記のINPUT、OUTPUTはインターフェイスの出入り口ではなく、先のフィルタリングポイントの説明で述べたLOCAL_IN、LOCAL_OUTであるということだ。すなわちルーティング処理の後であり、ローカルのネットワーク上位層とのやり取りが行われるパケットにしか、フィルタは有効に機能しない。つまりfilterテーブルのINPUTではアドレススプーフィングを回避することはできないということだ（アドレススプーフィングはnatテーブルのPREROUTINGで対処しなければならない）。なおこれらのことはLinuxのNetfilterフレーム

ワークの実装によるものであって、汎用的な定義ではないということも付け加えておく。

フィルタリングルールの作成と管理

ざっとNetfilterフレームワークの全体像を説明したので、実際にiptablesコマンドを使用してパケットフィルタリングルールを作成してみることにしよう。iptablesユーティリティのコマンドやオプションを個別に説明していると日が暮れてしまうので、これらに関してはヘルプやmanで確認していただきたい。

パケットフィルタリングルールは、各テーブルの各chainに対して設定していく。ユーザー定義のchainを作成することもでき、これは複数のchainに同様のルールを設定するときなどに便利だ。

また各テーブルの各chainにはそれぞれポリシーが指定されている（デフォルトはACCEPT）。これはパケットがどのルールにもマッチしなかった場合のアクションになる。基本的にはすべてのパケットは通過させ、特定のものだけブロックする場合はデフォルトのACCEPTポリシー、逆に特定の通路だけを空け、あとはブロックしてしまう場合はDROPポリシーを指定する。ポリシーはiptables -P chain名 ターゲット（アクション）名で変更可能だ。なおこのchainのポリシーに頼らず、すべてのパケットにマッチするアクション（アドレスとして0.0.0.0を指定）をルールの最後に指定しておくことで全体のセキュリティポリシーを決定することもできる。どちらかというと後者のほうが、設定が明確になるというメリットがあるのだけれど、処理内容が変わるわけではないのでどちらでも構わない。

あと言うまでもないことだけど、各chainに作成していくルールの順番には大きな意味がある。各フィルタリングポイントでは、パケットが通過する際にその内容と設定されているテーブルのフィルタリングルールとのパターンマッチングが行われるわけだが、このマッチング処理は基本的にリスト順にシーケンシャルに実行されていく。マッチング処理はマッチしてパケットの処理方法が決定されるまで続けられ、最終的にどのルールにもマッチしない場合は、先ほどchainのポリシーで説明したアクションが適用されるようになっている。

ルールの設計ポイントとしてはまず、ひとつのルールとパケットとのマッチング処理時間は極めて短いものだが、ルールの数が多かったり、処理するパケットの数が多かったりすると、塵も積もればというわけで、大きな時間のロスとしてパフォーマンスにも影響してくるようになるということだ。そこでルールを設計する際には、できるだけルールの数は少なく、かつパターンマッチの確立が高いものから登録していくように心がけたい。ルールの数を少なくするには、アドレスの指定ではマスクの活用、ポート番号の指定では“137-139”のようなレンジや“21,80”のような複数設定（7ポートまで指定可能）、さらに指定以外を表す“!”を適宜使用するとよいだろう。

また最初にマッチしたルールが適用されることになるので、あるポートをブロックするルールのあとに特定のアドレスだけ通過させるルールを登録しても意味をなさない。局所的な例外ルールは先に、広範囲に影響するルールは後に設定する必要がある。

とにかくルールを設計するには、場当たり的に登録していくのではなく、個別の用件を洗い出したのち、その影響範囲を考えて設定を組み立て直すことが大切だ。

■フィルタリングルールの作成

まずは試しに192.168.0.11との接続以外は通過させない設定を行ってみる。設定後は、きちんと機能しているか適当なテストを行い、その統計情報を確認してみるとよい。

```
# modprobe ip_tables
# iptables -A FORWARD -d 192.168.0.11 -j ACCEPT
# iptables -A FORWARD -s 192.168.0.11 -j ACCEPT
# iptables -P FORWARD DROP
# echo 1 > /proc/sys/net/ipv4/ip_forward
# iptables -L（ルールの確認）
Chain FORWARD (policy DROP)
target   prot opt source               destination
ACCEPT   all  --  192.168.0.11        anywhere
ACCEPT   all  --  anywhere            192.168.0.11
# iptables -L -v（統計情報の表示）
Chain FORWARD(policy DROP 7748packets, 650832bytes)
pkts bytes target prot opt in out source destination
3807 320K ACCEPT all -- any any 192.168.0.11 anywhere
3808 320K ACCEPT all -- any any anywhere 192.168.0.11
```

■IPマスカレードの設定

次にNATの設定を行ってみる。NATといっても用途に応じたいくつかの手法があることは先に述べたとおりだ。一般に多くのユーザーにとって有益でよく利用されるのが、内部ネットワークのアドレスをグローバルな1つのア

ドレスに変換しポート番号も適宜変更して通信を行うMASQUERADEである。特に変換するポート番号のレンジを制限したり、静的な設定をしなかったりする場合は次のコマンドだけで設定は完了する（インターフェイスは便宜上ppp0としている）。

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

見ればわかると思うが、上記のコマンドでは、natテーブルを選択し、chainにはPOSTROUTING、出口インターフェイスにはppp0、そしてアクションにMASQUERADEを指定している。難しいところは何ひとつないと思う。この設定によりppp0インターフェイスから出ていくすべてのパケットは、ppp0に設定されたアドレスに変換されるようになる。この指定はppp0インターフェイス自体がないと設定できないが、そのインターフェイスにアドレスが付けられていなくても問題はない。その場合、すべてのパケットは単にブロックされてしまうだけである。

よくNATを利用するとセキュリティ的にも安全といわれるが、これには少し間違いがある。実は内部ネットワークにプライベートIPアドレスを使用しているからこそ、比較的安全なのだ。基本的にプライベートIPアドレス宛のパケットがインターネットを経由してルーティングされることはない。だから問題ないだけで、もし外部ネットワークから直接内部ネットワーク宛のパケットが送られてくるような場合は、上記の設定だけだと、セキュリティ対策としての価値はまったくない（普通通り内部にルーティングされてしまう）。まぁいい加減な管理者が管理しているプロバイダでない限り大丈夫だと思うが、気になるようならPREROUTINGで外部からの内部ネットワーク（プライベートIPアドレス）宛のパケットを明示的にDROPしてしまうのがよいだろう（設定するアドレスは適宜変更すること）。これでやっとNATの部分のセキュリティ対策だけに注力することができる。

```
# iptables -t nat -A PREROUTING -i ppp0 -d
192.168.0.0/24 -j DROP
```

■NATとフィルタリングを組み合わせる

MASQUERADEを使用して動的なポート番号マッピングを行っている場合、外部ネットワークから内部ネットワークに対してパケットを送りつけることは極めて難しい。それはほぼすべてのマッピングされたポートは内部から外部へ能動的に開かれたもので、受動的なポートではないからだ。能動的なポートであってもシーケンシャル番号などのいくつかのハードルを越えれば、偽パケットを送りつけてシステムを混乱させたり、乗っ取りを行ったりもできるが、あまりに面倒なのでそれをやろうと思う人間は希である。ただし、ポートの静的マッピングなど受動的なポートを設置する場合は、公開サーバと同等のセキュリティ対策を講じる必要がある。また内部ネットワークだけでなく、NAT機能を提供しているLinuxマシンのローカルのセキュリティも忘れずに設定しておきたい。

まずLinuxマシンのローカルの設定としては、外部ネットワークからのパケットをINPUTでDROPしてしまうということが考えられる。その際、Webなどを公開している場合はそのルールの前に明示的にACCEPTしておくとよいだろう。また名前解決のことも考慮しておこう。とりあえず次のようなルールが考えられる。

```
# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
# iptables -A INPUT -s DNSサーバアドレス -p udp
--sport 53 -j ACCEPT
# iptables -A INPUT -i ppp0 -j DROP
```

この設定により、外部ネットワーク（ppp0インターフェイス）からNATを提供するマシンのローカルに対しては、Webサービスへの接続要求とDNSの返答パケット以外はブロックされる。INPUT chainはローカルに入ってくるパケットに対してのフィルタなので、内部ネットワークのクライアントと外部ネットワークのサーバとの接続や、内部でのこのマシンへの接続には特に制限は生じない。

■DNAT

NATには先に述べたように目的に応じたいくつかの種類がある。SNATとMASQUERADEは内部ネットワークから外部にアクセスするため、DNATとREDIRECTは外部からの要求を操作するために使用される機能だ。後者は、マルチプルサーバ、ロードバランシング、透過プロキシ、ポートフォワーディングなどに利用される。もちろん内部ネットワーク上のサーバを公開する際にも利用できる。この例を紹介しておこう。

```
# iptables -t nat -A PREROUTING -i ppp0 -p tcp --
dport 80 -j DNAT --to 192.168.0.3:80
```

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6回
【ディレクトリ配置】（ディレクトリはいち）
【/bin】（すら・びん）
【/usr】（すら・ゆーざー）
【/etc】（すら・えとせ）
【/var】（すら・ばー）
【/dev】（すら・でぶ）
【/root】（すら・るーと）
【lost+found】（ろすと・あんど・ふぁうんど）

ディレクトリ配置
【ディレクトリはいち】

Linuxのファイルシステムは、ディレクトリを利用した階層構造を採用しているため、あらゆるファイルをディレクトリごとに整理して格納する。その整理のしかたはユーザーに一任されているが、長い歴史のなかである程度の標準的配置が慣習として定められてきた。「/usr」「/var」などはその例であり、それぞれのディレクトリごとに用途が決まっている。しかし、なにごとも素直に受け入れるばかりでは、進歩はありえない。「Program Files」や「ﾃﾞｽｸﾄｯﾌﾟ」（必ず半角カナで）などのディレクトリを作成し、あなただけのLinux道をきわめてみるのも一興であろう。

/bin
【すら・びん】

binaryの略から。誰もが利用する、重要な基本コマンド群だけを収容するディレクトリ。Linuxのコマンドにとって、/binに入ることは殿堂入りにも等しい最大の栄誉である。会社でいえば“取締役”にあたる。/binよりワンランク下には「/usr/bin」（ゆーざー・びん）があり、「重要だが、いざというときには責任をおしつけて辞任させてもいい」コマンドを収容している。会社では“中間管理職”に相当する。そのさらに下が「/usr/local/bin」（ゆーざー・ろーかる・びん）であり、「ハードディスクが足りなくなったらいつでもリストラの対象」にされかねないような立場のコマンドが置かれている。いわば“平社員”である。

すら・しゃちょー

Linux界では、多くのコマンドがこのような出世コースをのぼりつめるべく、熾烈な戦いを繰り広げている。最近では、perlがその実績を認められてユーザーのホームディレクトリから/usr/local/binへ、そして/usr/binへと昇進し、ノンキャリア組の出世頭と注目された。いっぽうで、このような年功序列の評価システムは組織の硬直化という弊害も生んでおり、edやsetserialのような誰も使わないコマンドが/binに居座っている現状を憂う識者は多い。グローバリゼーション、IT化の波に乗り遅れないために、Linux界にも成果主義を取り入れ、人材を的確に評価するしくみを導入することが不可欠であろう。

/usr
【すら・ゆーざー】

user services and routinesの略。どのユーザーも共通して利用するような、一般的なデータ／コマンドを配置するディレクトリ。/usr以下は、さらに下記のようなディレクトリで細分類される。

・/usr/src（ゆーざー・そーす）

ウスター、おたふく、デミグラスなど、ひととおり備えておくとイザというときに役に立つディレクトリ。Linux主婦としての腕前がここに表れるといっても過言ではない。なお、使い終ったソースは瓶のキャップをきちんと閉めておく

よう心がけたい。ソースをオープンにしておくなど、もってのほかである。

・/usr/share（ゆーざー・しゃれ）

小話をストックしておくディレクトリ。つまらないものは、特に/usr/share/daに分類、収容される。利用者の便宜をはかるため、可能なら/usr/jare/daにシンボリックリンクを張っておきたい。

・/usr/man（ゆーざー・まん）

見つかると最後なディレクトリ

漢（おとこ）の、漢による、漢のためのディレクトリ。女人禁制で実用性満点なデータが大量におさめられている。古来、男性Linuxユーザーは、夜を徹しての作業に行き詰まると、もやもやを解消すべく、ひとりmanに耽ったとされている。最近ではLinux界における男性上位、女性蔑視の象徴と目されており、/usr/womanの設置をめざすフェミニズム運動が盛んになりつつある。今後は/usr/gay、/usr/lesも標準化するなど、なおいっそうの性の多様化、性の解放が課題となりそうだ。

/etc

【すら・えとせ】

有料道路の料金所に設置したアンテナと通行車両に装着した機器との間で無線通信を行い、停車することなく自動的に通行料を徴収するしくみ。ノンストップ自動料金収受システムの略。その多くが料金支払い時に発生している高速道路での交通渋滞緩和につながるだけでなく、石原都知事が推進する東京都ロードプライシング政策実現のカギともいわれ、期待が集まっている。

首都高速道路公団では、2002年度末までに全料金所にETCを設置する予定をたてているが、目下の課題は一般人への周知徹底。編者の独自調査によれば、東京・秋葉原の世情にうといLinuxユーザー100人に聞いたところ「ETC？ ああ、システム設定ファイルを保管しておくディレクトリでしょ？ ごめん。これからAthlon 1.1GHz買いに行くから失礼するね」というワケのわからない勘違い回答がもっとも多かった（2番目に多かったのが「宗教は間に合っているから」などの無効回答）。最近の若者の時事問題への無関心さには、ほとほと困ったものである。

/var

【すら・ばー】

variable but not valuable data filesの略。放っておくとどんどん増えるが、どうせ本当に大切なものなどなにひとつありはしないログや一時データの収容場所として使われる。/var以下は、さらに下記のようなディレクトリで細分類される。

・/var/lock（ばー・ろっく）

ミック・ジャガーや忌野清志郎、ジミー・ヘンドリックスなどのMP3ファイルを収容する。菊池桃子、ラ・ムーもここに入る。

・/var/log（ばー・ろぐ）

ログファイルの保管場所。syslogたちがせっせとゴミをためているが、ログなどどうせ誰も読まないのでパーミッションは333でよい。

/dev

【すら・でぶ】

一部のふくよかな人々にとっては、声に出して読みたくない名前のディレクトリ。中になにが入っているかは、あえて考えたくない。

/root

【すら・るーと】

管理者様用ホームディレクトリ。管理者特権を濫用して、秘蔵ムービーやMP3ファイルを貯蔵しておくのに最適。ただし、chmod 0 を忘れてはならない。

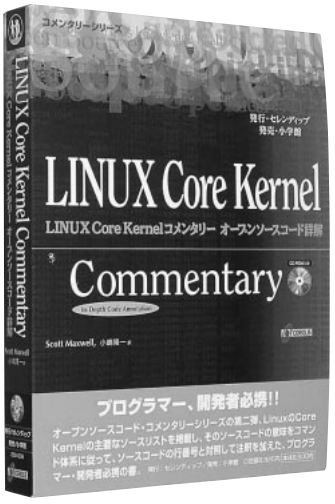
lost+found

【ろすと・あんど・ふぁうんど】

預かり所

遺失物預かり所の意。OSがファイルシステムを探索中に落とし物ファイルを見つけると、ここに保管しておく。なにかなくした、落としたと思ったときは、探してみると吉。母さん、僕のあの帽子どうしたんでしょうね？ ええ、夏、碓氷から霧積へ行くみちで渓谷へ落とした、あの麦わら帽子ですよ。ちなみにこのあいだfindしてみたところ、編者が失った人生の輝きはまだ見つかっていないようだった。

Books



Linux Core Kernelコメンタリ　オープンソースコード詳解

Scott Maxwell 著　小嶋隆一 訳
セレンディップ／小学館
A4変形判／672ページ／CD-ROM1枚付き　　本体価格 8500円

6月号で紹介した『Apache Serverコメンタリー』に続く、オープンソースソフトウェアのコードを解説する「コメンタリー」シリーズの第2弾。今回は、Linuxシステムの中核となるカーネルのそのまたコアになる部分を取り上げている。前半のソースコード部分と後半のそれに対する解説という2部構成で、掲載されているソースコードは2.2.5のもの。いくぶん前のバージョンだが、カーネルの本質的な面を理解していくうえで問題はないはずだ。

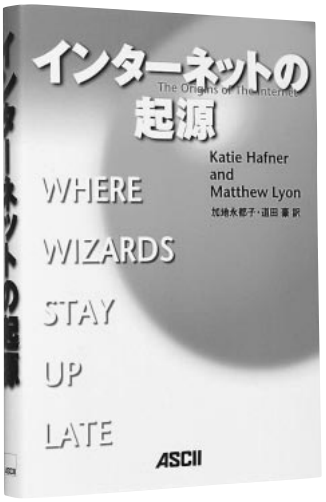
ソースコードの解読を通して、プロセス管理、システムコール、メモリ管理などの機能がどのように実装されているのかを解き明かしていく。Linuxカーネルの構造とコーディングの特徴についても触れられている。Linuxの良質な解説書であると同時に、Linus Torvalds氏をはじめとするカーネル開発者と筆者の共著によるC言語の教科書ともいえる内容だ。CD-ROMには、0.1、2.2.5、2.2.10、2.3.12のソースコードと検索用のelispファイルを収録。

Debian GNU/Linux ―インストールから活用まで

ジョン・ガーゼン／オマッサ・オスマン 著　木田直子 訳
ピアソン・エデュケーション
A5判／210ページ／CD-ROM2枚付き　　本体価格 2600円

8月15日に正式リリースされたDebian GNU/Linux 2.2の解説書が早くも登場！　というワケではなくて、2.2のテストバージョンをベースにしたDebianの入門書だ。本書の執筆時点では、すでにコードフリーズ状態になっており、正式リリースのものと機能面での大きな変更点はないのでご安心を。2枚の付録CD-ROMには、それぞれ2.1r4と2.2-test-cycle-2が収録されている（両者ともオフィシャルi386版のBinary-1 CD）。こちらも、一般に高く評価されているDebianのパッケージ管理機能を利用すれば、インターネットなどを通じてアップグレードできるはず。巻末には、オフィシャルCDの申し込み方法も紹介されている。

内容的には、インストール、コマンドやX Window Systemの使い方、パッケージ管理といったLinux入門書ではおなじみのもの。CD-ROM6枚にも及ぶ巨大ディストリビューションとなったDebianの世界を気軽に体験できる1冊である。



インターネットの起源

Katie Hafner／Matthew Lyon 著　加地永都子／道田豪 訳
アスキー
A5判／336ページ　　本体価格2500円

隆盛を極めるインターネットの原型となった「ARPANET」の歴史を、1958年のARPA（米国国防総省・高等研究計画局）設立から、1989年のARPANETの終焉まで克明に記録したドキュメント。ARPANETの発展過程はそのままコンピュータ／ネットワークの歴史でもある。telnet、FTP、TCP/IP、電子メールなど、現在も使われている技術は、直接あるいは間接的にARPANETから生み出されたといっていい。ARPANETに関わった人々はみな、それぞれに個性的で魅力的な人物ばかりだ。ネットワークの持つ可能性を早くから予見していたJ.C.R.リックライダー、プロジェクトの初期にそのカリスマで強力な推進役を果たしたロバート・テイラー、「ARPANETの父」ラリー・ロバーツ、サイト間を結ぶIMP（Interface Messeage Processer）を作り上げた「IMPガイ」たち……。彼らの苦闘の様子、情熱的な仕事ぶりを丹念に追っていくうちに、当時の息吹までもが伝わってくる。

ハッカーvs.不正アクセス禁止法

園田寿／野村隆昌／山川健 著
日本評論社
四六判／272ページ　　本体価格 1400円

なにゆえ法律の文章はあんなふうなんだろう。どんなふうかというと、たとえば――「アクセス制御機能を有する特定電子計算機に電気通信回線を通じて当該アクセス機能に係る他人の識別記号を入力して当該特定電子計算機を動作させ、当該アクセス制御機能により制限されている特定利用をし得る状態にさせる行為（当該アクセス制御を……」――こんなふうだ。これは今年の2月に施行された本書のメインテーマでもある、いわゆる「不正アクセス禁止法」からの抜粋。これではとても全文を読む気はおきない。

そこで本書の登場である。全体的な内容は、コンピュータと不正アクセス、コンピュータ犯罪と法制度について考察したものだが、特に園田氏による不正アクセス禁止法の解説は必読。とてもわかりやすく、この法律を理解するために非常に役立つ。条文ごとに「わかりやすく書くとこうなる」コーナーが設けられているのがうれしい。

ターボリナックス認定資格対応4講座

大好評受付中

TURBO Certified Administrator　TURBO Certified Engineer

LEC東京リーガルマインド
新宿本校

毎月2回実施
各日PM3:00～
9月24日㊐
10月14日㊏・28日㊏

お問い合せは、e-mailにて。
linux@lec-jp.com

完全事前予約制

TURBOLINUX™
無料体験

LinuxBasicコース　T100-J 6時間
一般的なLinuxの知識を理解し、Linuxについて第三者に説明ができるようになります。初心者向けコース。
●平日コース（2日間）　●土日コース（1日間）

Linuxシステム構築コース　T200-J 12時間
ネットワーク、アプリケーションの基礎知識を習得し、初期設定ができるようになります。中級者向けコース。
●平日コース（4日間）　●土日コース（2日間）

LinuxネットワークA mailコース　T310-J 18時間
Mailサーバに特化した、Linux・サーバの構築、運用ができるレベルの技術が身につきます。上級者向けコース。
●平日コース（6日間）　●土日コース（3日間）

LinuxネットワークB webコース　T320-J 18時間
Webサーバに特化した、Linux・サーバの構築、運用ができるレベルの技術が身につきます。上級者向けコース。
●平日コース（6日間）　●土日コース（3日間）

カリキュラム・日程・料金などの詳細はlinux@lec-jp.comへお問合せください。

れっく LEC 東京リーガルマインド

■お問い合わせ　LEC東京リーガルマインド　開発企画部
〒107-6018 東京都港区赤坂1-12-32アーク森ビル TEL.03-5572-7547
■実　施　校　LEC東京リーガルマインド　新宿本校
〒160-0021 新宿区歌舞伎町2-1-11
http://www.lec-jp.com/

読者の声

俺にもいわせろ！

みなさんこんにちは。編集部の一部ではPalm PDAが流行中です。なにをいまさらなんて言っていた担当も、結局はIBM WorkPad c3を購入しました。さまざまなオンラインソフトが発表されており、これがまた面白い。もうPalm猿と化しています。うっきー。

9月号特集1へのお便り

「夏休みフリーソフト大会」と、フリーソフトのCD-ROM収録はとても嬉しかったです。Linuxのソフトは、ほかのメジャーなOSに比べてあまり情報がなく、困っていたところだったので、とても感謝しています。各ソフトの説明も非常によかったです。

（群馬県　笠井大騎さん）

WindowsユーザーのLinux初心者です（LASER5 Linux 6.2 Develで勉強中）。

フリーソフト大会のソフトの説明とサイトの紹介（ペンギン活用委員会）はありがたかったです。今後も初心者に役立つ誌面づくりをお願いします。

ところでWindowsのデフラグのようなツールはLinuxにあるのでしょうか？

（群馬県　吉田 誠さん）

ありがとうございます。Linuxで動作するフリーソフトは、星の数ほどあります。今後も、役に立つソフトウェア、面白いソフトウェアをどんどん紹介していきますので、ご期待ください。

Linuxで標準的に採用されているext2ファイルシステムのデフラグツールは、次のURLで見つけることができます（ftp://ftp.uk.linux.org/pub/linux/sct/defrag/ext2/）。しかし、ext2ファイルシステムは、Windowsで利用されるFATファイルシステムとは違い、フラグメンテーションが起きにくいように設計されています。このため、フラグメンテーションによって著しくパフォーマンスが低下することはほとんどありません。従って、危険なディスク操作を伴うデフラグを行うメリットはあまりないでしょう。また、このツールは編集部でもテストをしていないので、利用する際は十分注意をしてください。

このほかに、バックアップ＆リストアによってフラグメンテーションを軽減するという手もあります。

9月号特集2へのお便り

「root見栄講座」は、お茶らけた（？）タイトルとは異なり、とても参考になる内容でした。まさに序文にあるとおりで、日ごろ避けてきたことをキッチリ解説してあり、よかったです。

（千葉県　緒方宏昭さん）

「root見栄講座」よかったです。ブ厚い書籍などを読んで、コツコツ学んでいくのが正道なのでしょうが、最初の一歩としては、このような特集はとても役に立ちます。

（沖縄県　奥田 章一郎さん）

「root見栄講座」には多くの反響をいただきました。管理者たるもの、システムのすみずみまで把握して、気持ちよくマシンを使いたいですよね。

もっとも、継続して管理し続けることが大事なのですが、担当はそのあたりがニガテです。ああ、ホームディレクトリにいろんなファイルが増殖していく。

奥田さんは、オーストラリアから読者ハガキを送ってくださいました。切手の図案はコアラです。うーん、行ってみたい〜。

プログラミング工房へのお便り

「プログラミング工房」を楽しく読ませていただきました。エアコンをパソコンで制御するという無謀な題材ですが、これが面白いと感じます。エアコンのリモコンを利用するつもりはありませんが、ほかのリモコンキットで実際にやってみたいと思います。

（福井県　服部昌博さん）

WidnowsではDOSとは違い、外部機器の制御は非常に難しくなりました。今後は、I/Oポートを使用するために、もっぱらPC UNIXが利用されるようになるでしょう。「プログラミング工房」のサンプルプログラムは最も簡単でわかりやすい例であると思います。

プログラミングの記事は毎号楽しみにしています。

（兵庫県　齊藤 暁さん）

ハードウェア制御に関心を持っている方が多いのには驚きました。外部のハードウェアが思い通りに動いた瞬間の嬉しさは格別です。「制御」とか「接続」という言葉を聞くとワクワクするという方は、ぜひハードウェアにもトライしてみてください。

自分専用 My Linux

プレゼント、ほしいですね。それと時間がほしいです。

いい歳をして、最近インストールおたく気味。WinからLinuxへ全面移行したくて、一番良い環境で……、と思いながら始めたことなのですが。雑誌が頼りです。周辺機器がディストリビューションによってうまく認識・設定できたりできなかったり。カーネルをいじっても、よくわからないせいか、うまくいかなかったり。

Red Hat 6.2、LASER5 6.2の調子がいいけど、Kondaraのログイン画面が大好きで、Vineも一部捨てがたい。

いいとこ取りができないか思案中。

（福岡県　Ⓚさん）

いろいろなディストリビューションからソースをもらってきたり、tar.gzから新しいソフトをインストールしたりして、自分に合ったフルカスタマイズドLinux Boxを作る。これぞ王道。もとのディストリビューションがなんだったかもわからなくなるくらいにいじり倒せば、きっと使いやすさも愛着も世界一のマシンに仕上がるでしょう。

とかいいつつ、カスタマイズの途中でわけがわからなくなって、世界一使いづらいLinux Boxにしてしまうのが得意な担当なのでした。

hogeグッズプロジェクト発足？

編集部のホゲ感染者のみなさんこんにちは。ほげ遠藤です。保菌者になって数カ月、いまだ病は快復していません。そこで私は薬を発明しました。それは“ほげ”グッズです。ほげTシャツ、ほげマグカップ、ほげ灰皿、ほげ携帯ストラップ、ほげデスクトップ壁紙、等々。そこいら中に“ほげ”シールを貼っては喜んでいます。資金力が乏しいため、みなさんにお配りすることはできません。そこでぜひ、編集部のホゲ有志の方々で“ほげ”グッズを開発してください。しかし、“ほげ”の特許権がもしかしてあるかもしれません……。ホゲホゲホゲ。

（山形県　旧ホゲホゲ遠藤さん）

ホゲグッズとはまた……。手始めにホゲ壁紙から作ってみましょう。

ところで遠藤さん、タレコミのおハガキをいただいてます。さっそくご紹介しましょう。

旧ホゲホゲ遠藤さん大ピンチ！

“sl”コマンドなるものがあるようですね。画面をSLが横切って行くのを見てみたいものです、しかし、私は修行が足りず、“sl”とタイプミスしたことはありません。それから“dc”コマンドもいいですね。自動的にリボ払いになります。1番いいのは“pc”コマンドです。コピーされてPCが1台増えますから。

あ、ちょっと告げ口。ホゲホゲ遠藤さん、『日経Linux』に掲載されていました。恐るべし！　あっ、他誌を読んでいることがバレ……。

（福島県　香西 薫さん）

日経Linuxの読者でもあったんですね、遠藤さん、香西さん。Linux magazine編集部でも定期購読してますよ。もちろんLinux Japanも購読しています。各誌とも、互いに異なるテイストで興味深いものがあります。

slコマンドは実在します。拡張版では踏切もあったりして、実行により時間をかけるように工夫されているようです。その間は何もできないので、ミスタイプしたことを後悔するわけです。

また、dcコマンドもあります。これは冗談ソフトではなく、逆ポーランド記法のコマンドライン計算器コマンドです。もちろん、リボ払いにはなりません(^^)。いまのところpcコマンドは聞いたことがないですね。

ペンギンのノートがあれば

ペンギンケースのパソコンを作りました。お値段は本体9万円、ディスプレイ8万円。基本スペックは、CPUがDuron 600MHz、メモリ128Mバイト、ハードディスク20Gバイトです。家族の反応は、「17万円も出すのだったら、メーカー製のノートがいいんじゃないの」といまいちよくない。

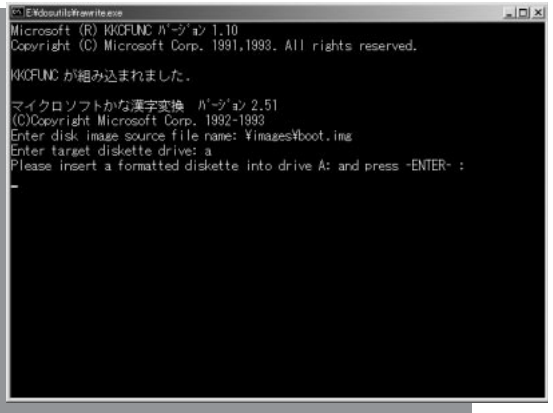
グラフィックスアクセラレータ（Riva TNT2）があるので、XFree86 4.0をインストールしてゲームするぞ。

（広島県　松井敏郎さん）

マシンの新調、おめでとうございます。デスクトップ機は、拡張性の高さや性能もさることながら、ペンギンケースを選べるというアドバンテージがありますよね。ペンギンノートが発売されればさらに楽しいんですけど、なかなか実現しないかなぁ。

付録CD-ROMに収録した
Kondara MNU/Linux 2000 (FTP版) のインストール

本誌付録CD-ROM収録のKondara MNU/Linux 2000はFTP版です。非商用ソフトだけが含まれています。また、製品版を販売しているデジタルファクトリ株式会社からサポートを受けることはできません。なお、製品版に含まれるEmacs及びEmacs関連のパッケージ、WebブラウザのMozillaは収録されていませんので、以下のWebページを参考にして、必要なパッケージをダウンロードしてください。
FTPサイトのリスト：http://www.kondara.org/ftplist.html



ブート用フロッピーディスクの作成

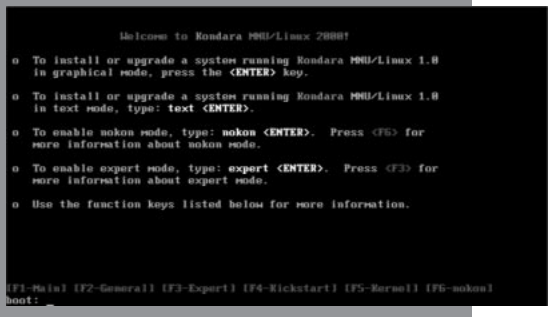
インストールするマシンがCD-ROMから起動できる場合は、CD-ROMからブートしてインストーラを起動します。CD-ROMから起動できない場合は、以下の手順でインストーラ起動用のフロッピーディスクを作成します（ここでは、フロッピーディスクドライブがA:であるとして解説します）。

（1）Windowsのエクスプローラで、CD-ROMの「dosutils」というフォルダを開き、その中にある「rawrite」をダブルクリックします。

（2）DOS窓が開き、ファイル名の入力を促してくるので、

\images\boot.img

と正確にタイプして[Enter]を押します。さらに、フロッピードライブ名を求めてくるので、「A」をタイプして[Enter]を押します。最後にフロッピーがセットされているかを確認して、[Enter]を押すと、フロッピーの作成が始まります。



インストーラの起動

作成したフロッピーディスクや、CD-ROMをドライブにセットして、マシンを再起動します。インストーラが起動し、「boot:」というプロンプトが表示されたら[Enter]を押します。インストール作業をすべてテキスト画面で行いたいときは、「boot:text」として[Enter]を押します。

使用言語とキーボードの設定

Linuxで日本語を使う場合は「Japanese」を選択して「OK」を押します。テキストベースの設定画面では、メニューを「↑」と「↓」で選択し、[Tab]と[Enter]を使って「OK」や「戻る」を押します。

次のキーボード設定では「jp106」を選択することで、日本語106キーボードが使えるようになります。

マウスの設定

4

デフォルトでは「3 Button Mouse(PS/2)」が選択されています。PS/2タイプの2ボタンマウスを使う場合は「2 Button Mouse(PS/2)」をチェックします。「3ボタンマウスエミュレーション」は、2ボタンマウスの左右ボタンを同時に押すことで、3ボタンマウスの中ボタンを代用する機能です。これは便利な機能なので、チェックしたまま「次へ」を押します。

インストールの開始

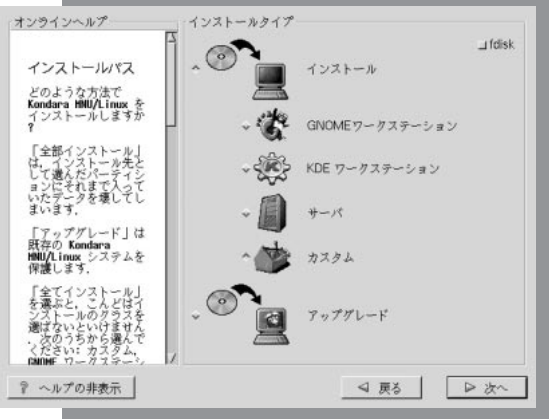
5

さて、ここから本格的なインストール作業の開始です。「次へ」を押します。

インストールタイプの選択

6

「GNOMEワークステーション」と「KDEワークステーション」を選択すると、ブートローダLILOがMBRにインストールされます。Patition Magicなど、他のブートローダでOSを起動するときは、これらのタイプを選択してはいけません。ここでは、より柔軟なオプション選択ができる「カスタム」を選択して「次へ」を押します。「アップグレード」は、すでにKondara MNU/Linuxの以前のバージョンをインストールしているユーザー用です。

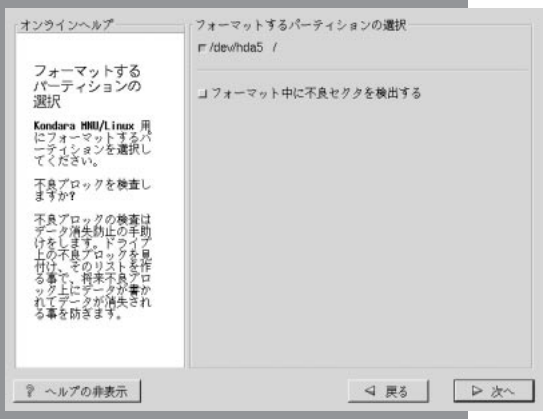


パーティションの作成

7

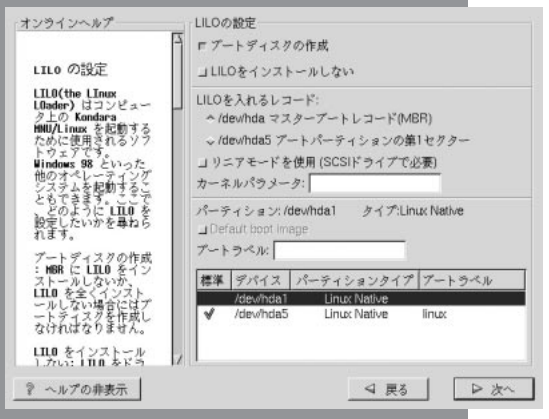
Linuxをインストールするには、最低限Linuxシステム用とSwap用の、2つのパーティションが必要です。「追加」を押すと、画面のようなダイアログが表示されます。Linuxシステム用のパーティションは、「マウントポイント」に「/」を、「パーティションタイプ」は「Linux native」を選択します。Kondara MNU/Linux 2000では、Linuxシステム用のファイルシステムとして、「Linux native（ext2）」のほかにも「Reiser FS」が選べます。新しいファイルシステムに興味のあるユーザーは、「Reiser FS」を選択するのもよいでしょう。Swap用パーティションは「Linux swap」を選択します。各パーティションサイズは、以下を目安にして設定してください。

Linuxシステム用　：フルインストールの場合は2Gバイト程度以上
Swap用　：搭載メモリの1～2倍程度



パーティションのフォーマット

前の場面で作成したパーティションをフォーマットします。チェックされている「/dev/hdax」パーティションがフォーマットされます。「フォーマット中に不良ブロックをチェックする」を選択しておくと、パーティションのフォーマット中に、ディスクの不良箇所を調べてくれます。急いでいなければ、チェックを入れておくと良いでしょう。最後にもう一度フォーマットするパーティションを確認して、「次へ」を押します。



LILOの設定

「LILOを入れるレコード」は環境に応じて選択します。

/dev/hdaマスターブートレコード（MBR）

・LILOを使用してWindowsとLinuxを起動時に選択する
・ディスクにはLinuxのみをインストールする

/dev/hda5ブートパーティションの第1セクター

・System Commanderなど、LILO以外のブートマネージャを使用する

緊急時のために「ブートディスクの作成」をチェックしておくとよいでしょう。

ネットワークとタイムゾーンの設定

イーサネットを使い、ネットワークを利用する場合は、この場面で設定します。「ブート時に有効にする」はチェックしたままで、DHCPでネットワーク情報を取得する場合は「DHCP使用時の設定」を選択し、DHCPを使わない場合は「DHCP使用時の設定」のチェックを外して、マシンのIPアドレス、DNSなど、ネットワーク利用に必要な情報を入力します。ダイヤルアップでインターネットに接続する場合は、インストール後に「netcfg」などを起動して、ネットワークを設定します。

次はタイムゾーンの設定です。マシンを日本で使うときは、世界地図上の日本をマウスでポイントするか、画面下方のメニューで「Asia/Tokyo」を選択して「次へ」を押します。

ユーザーアカウントと認証の設定

Linuxを使用するユーザーを設定します。まず、Linuxシステムに必須の管理者rootのパスワードとして、「管理者のパスワード」と「再度記入」に同じものを入力します。

次に一般ユーザーのアカウントを設定します。「アカウント名」のほかに、パスワードとして「パスワード」と「パスワード（確認）」に同じものを、「フルネーム」にユーザーのフルネームを入力して、「追加」を押します。この要領で複数の一般ユーザーを作成できます。ユーザー設定を終えたら「次へ」を押します。

認証の設定場面は、そのままの状態で「次へ」を押します。

パッケージグループの選択

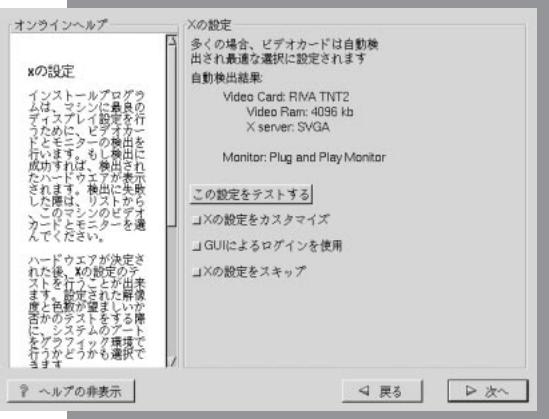
おおざっぱなパッケージグループを選択します。パッケージ群の意味がわかるユーザーは、好みのパッケージ群を選択し、どのパッケージ群を選んでよいかわからないユーザーはすべてのパッケージをインストールするために、最下段にある「Everything」を選択して「次へ」を押します。

Xの設定

Kondara MNU/Linux 2000のインストーラは、ビデオカードとモニタの自動認識を試みます。画面の例ではビデオカードとモニタの両方が自動認識されていますが、ビデオカードやモニタが自動認識されない場合は、ユーザーが手動で設定します。

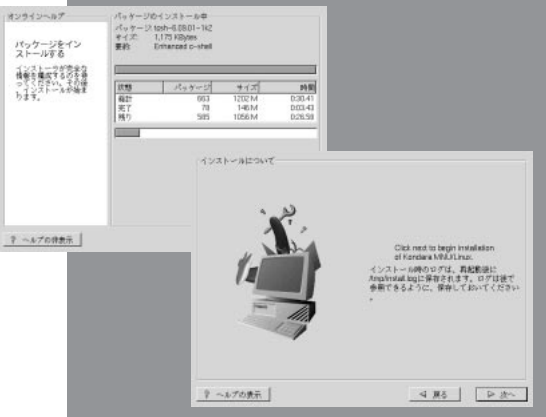
色数や解像度を手動で設定する場合は「X設定のカスタマイズ」をチェックし、グラフィカルなログイン画面を使う場合は「GUIによるログインを使用」をチェックします。

もし、この場面でXがうまく設定できないようならば、インストール後に、コンソールから「Xconfigurator」や「XF86Setup」を起動してXを設定するとよいでしょう。



パッケージインストールの開始

画面（手前）で「次へ」を押すと、パッケージのインストールが始まるので、インストールが終わるまでしばらく待ちます。



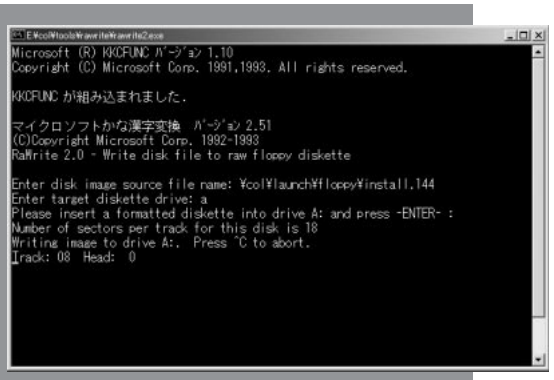
インストールの終了

LILOの設定場面で「ブートディスクの作成」をチェックした場合は、手前の画面が表示されます。「次へ」を押すと、緊急時用のブートディスクの作成が始まります。フロッピーの作成が終わると、奥の画面が表示されます。フロッピーディスクを抜いて「閉じる」を押します。

以上でインストール作業は終わりです。お疲れさまでした。

付録CD-ROMに収録した Caldera Linux Technology Preview(Caldera LTP) のインストール

本誌付録CD-ROM収録のCaldera Linux Technology Previewはフリー版です。非商用ソフトだけが含まれており、開発元のCaldera Systems, Incからサポートを受けることはできません。また、Caldera Linux Technology Previewはおもに開発者向けのディストリビューションなので、安定して動作するLinuxシステムを使いたいユーザーは、ほかの安定版ディストリビューションの使用をお勧めします。



1 ブート用フロッピーディスクの作成

インストールするマシンがCD-ROMから起動できる場合は、CD-ROMからブートしてインストーラを起動します。CD-ROMから起動できない場合は、以下の手順でインストーラ起動用のフロッピーディスクを作成します（ここでは、フロッピーディスクドライブがA:であるとして解説します）。

（1）Windowsのエクスプローラで、CD-ROMのフォルダを「col」→「tools」と開き、「rawrite」フォルダ内にある「rawrite2」をダブルクリックします。

（2）DOS窓が開き、ファイル名の入力を促してくるので、

\col\launch\floppy\install.144

と正確にタイプして[Enter]を押します。さらに、フロッピードライブ名を求めてくるので、「A」をタイプして[Enter]を押します。最後にフロッピーがセットされていることを確認して[Enter]を押すと、フロッピーの作成が始まります。

2 インストーラの起動と使用する言語の選択

CD-ROMブートができるマシンではCD-ROMを、できないマシンでは、作成したブート用フロッピーディスクとCD-ROMをセットして、マシンを起動します。するとインストーラが起動するので、「Standard install mode (recommended)」を選択して[Enter]を押します。

次にLinuxで使用する言語の選択ですが、画面のメニューのようにCaldera LTPでは日本語環境が用意されていません。ここでは「English」を選択して「Next」を押します。

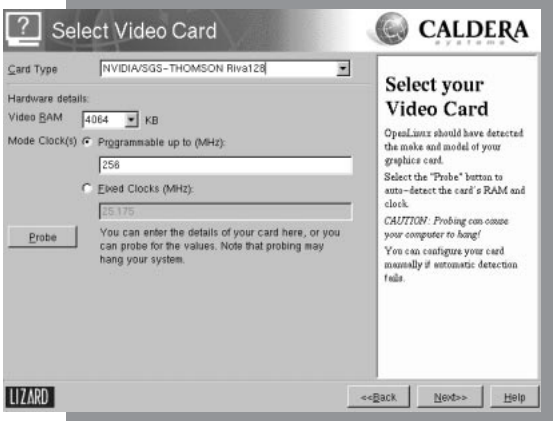
3 マウスとキーボードの設定

まずはマウスの設定です。使用するマウスがPS/2タイプなら「PS/2 mouse」を選択し、ホイールがついていないタイプならば、画面のプルダウンメニューで「Standard without wheel」を選択します。さらに2ボタンマウスであれば、画面の下側にある「Emulate 3rd button」にチェックを入れておくとよいでしょう。

次にキーボードの設定です。デフォルトでは英語キーボードを使用するように設定されています。日本語キーボードを使用する場合は、「Select Model」で「Japanese 106-key」を、「Select Layout」で「Japanese」を選択します。

ビデオカードの認識

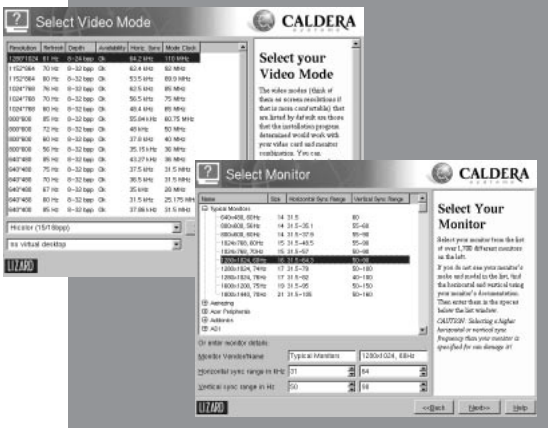
多くの場合、ビデオカードは自動で認識されます。自動認識されていれば、「Card Type」と「Video RAM」に、それぞれビデオカードの種類と、ビデオカードに搭載されているメモリ量が表示されます。もし、ビデオカードが自動認識されていないようならば、「Card Type」でビデオカードの種類を選択し、「Video RAM」の欄にKバイト単位でメモリの量を入力します。ビデオカードのメモリ量がたとえば16Mバイトであれば、入力する数値は（16×1024=）16384になります。設定が終了したら「Next」を押します。



モニタの設定

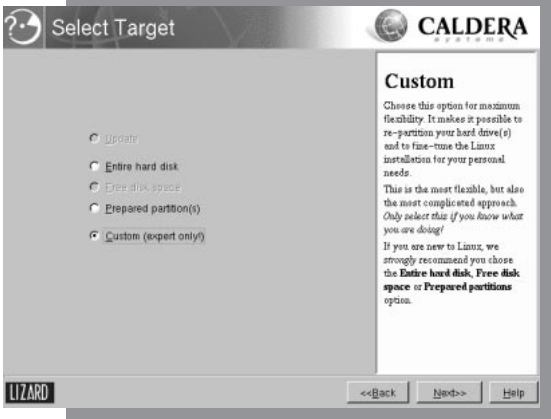
まず、使用するモニタをメニューから選択します。リスト中に使用するモニタがなければ、「Typical Monitors」からモニタに合うタイプを選択します。この「Typical Monitors」からモニタのタイプを選択する場合は、1024×768の解像度で使用できるモニタに、上限解像度を超える1280×1024などを選択しないように気をつけてください。モニタのマニュアルを参照すれば、確実に設定できるでしょう。

次にLinuxで使用する解像度を設定します。まず、800×600など小さい解像度を選択して、「Test this mode」を押し、Xの表示をテストします。表示が成功するとKDEのデスクトップ画面が表示されます。低解像度を選択してXがうまく表示されたら、今度は一段階解像度を上げ、「Test this mode」を押してXの表示をテストします。解像度を少しずつ上げて、最適なものを選択してください。



パーティション作成方法の選択

ハードディスクをCaldera LTP専用で使うときは「Entire hard disk」を、Linuxで使用するパーティションをあらかじめ用意しているときは「Prepared partition(s)」を、これからLinux用のパーティションを作成するときは「Custom(expert only!)」を選択します。「Entire hard disk」を選択すると、既存のデータはすべて消去されます。ここでは柔軟にパーティションを作成できる「Custom(expert only!)」を選択して「Next」を押します。



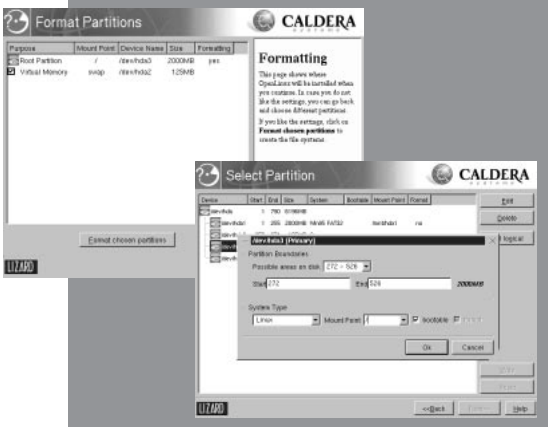
パーティションの作成

Linuxをインストールするには、最低限Linuxシステム用のパーティションと、Swap用パーティションを作成する必要があります。設定画面でパーティションを選択して「Edit」を押すと、画面のようなダイヤログが表示されるので、

■Linuxシステム用パーティション
　　System Type「Linux」　　Mount Point「/」

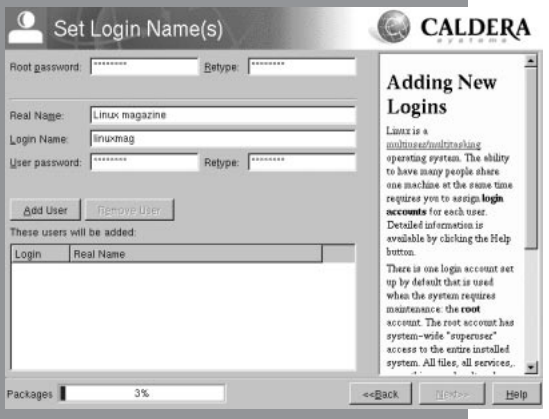
■Swap用パーティション
　　System Type「Swap」　　Mount Point なにもしない

と、それぞれ設定します。作成するパーティションサイズは「End」の欄に数字を入れて設定します。Caldera LTPはすべてのパッケージをインストールすると、1.5Gバイト程度のディスクを使用するので、Linuxシステム用に2Gバイト程度（余裕があればそれ以上）のパーティションを作成するとよいでしょう。Swap用のパーティションは、搭載しているメモリの1～2倍程度作成します。次の画面で「Format chosen partitions」を押して、パーティションをフォーマットします。



インストールするパッケージタイプの選択

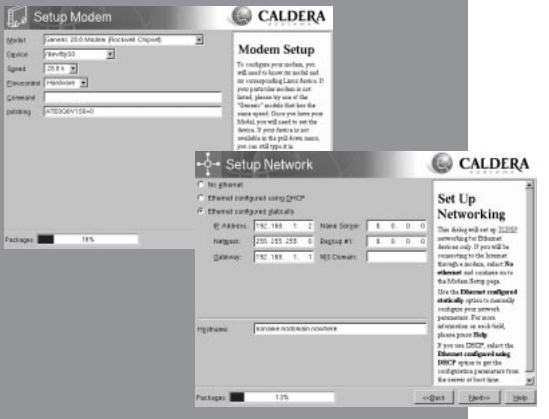
デフォルトでは「Recommended」が選択されています。ディスクの要領に余裕があれば「All Packages」を、Linuxをサーバ用に使うのならば「Minimum Installation」を選択するとよいでしょう。こまかくパッケージを選択する場合は「Refine Selection」をチェックして「Next」を押します。



パスワードの設定

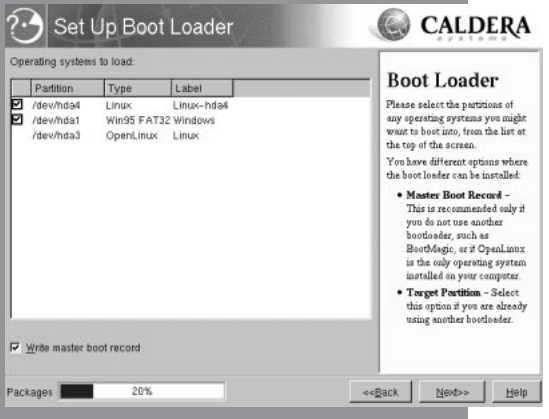
ここではLinuxシステム管理者rootのパスワード設定と、一般ユーザーのアカウント作成を行います。まず、「Root password」と「Retype」の欄に、5文字以上の同じパスワードを入力します。

次に一般ユーザーを作成します。「Real Name」に本名を、「Login Name」にログイン名（ログイン時に使用します）を、「User password」と「Retype」に同じパスワードを入力します。「Add User」を押すと一般ユーザーが作成されます。ここまでの作業が終わったら、「Next」を押して次へ進みます。



ネットワークの設定

イーサネットを利用するときは手前の画面にネットワーク情報を入力し、TAやモデムを使ってネットワークを利用するときは、後ろの画面で通信機器を設定します。ここではネットワーク設定の詳細な解説を省略します。



ブートローダの設定

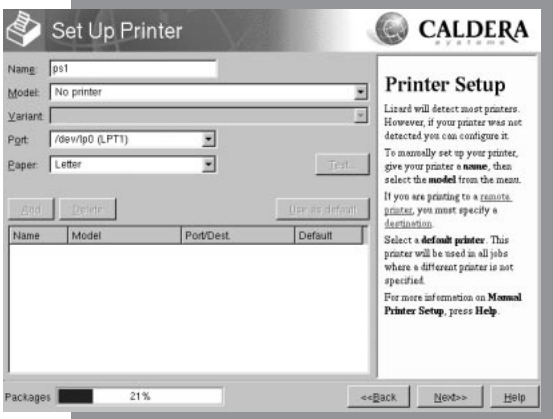
Caldera LTPではGrubというブートローダを使用します。このGrubを使ってCaldera LTP以外のOSを起動する場合は、四角の枠にチェックを入れます。どれをチェックしてよいかわからないときは、すべてをチェックしておくとよいでしょう。

System Commanderなどの専用ブートマネージャを使用するときは「Write master boot record」のチェックを外します。

プリンタの設定

この画面でプリンタを設定しますが、プリンタごとに設定内容が異なるので、ここでは解説を省きます。

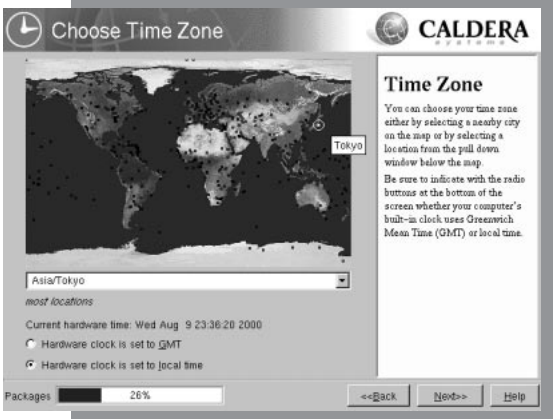
12



タイムゾーンの設定

時刻を設定します。日本の時刻に合わせる場合は、世界地図上の日本をマウスでポイントします。地図の下にあるプルダウンメニューで「Asia/Tokyo」を選択してもかまいません。また、1台のマシンでWindowsなどほかのOSを使用するときは、「Hardware clock is set to local time」をチェックして「Next」を押します。

13



ゲームをする

パッケージのインストールが終了するまで「上海」ライクなゲームを楽しんでください（ちなみにCaldera Open Linux 2.3は「テトリス」ライクなゲームでした）。

14

緊急時用フロッピーディスクの作成

ハードディスクのトラブルなどでOSが起動しなくなったときのために、緊急時用のフロッピーディスクを作成します。空のフロッピーディスクをドライブにセットして「Write Disk」を押すと、緊急時用フロッピーの作成が始まります。

フロッピーディスクの作成が終わったら「Finish」を押して、インストール作業を終了します。お疲れさまでした。

15