## IBM、NEC、日立、富士通 4社が連携 「エンタープライズLinux」 ～Linuxの機能強化をオープン・ソースコミュニティへ共同提案～

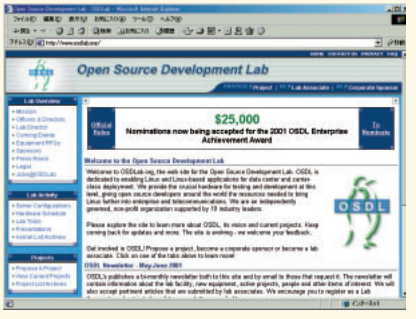
IBM、NEC、日立製作所、富士通の4社は、5月30日、オープン・ソースの開発プロセスに基づき、エンタープライズLinuxの機能強化およびオープン・ソースコミュニティへの共同提案活動を、4社の協業のもとで推進していくことで合意したと発表した。

オープン・ソースのOSであるLinuxの堅牢性や信頼性が認められるにしたがって、企業の基幹業務や研究用システムとしてのニーズが増えてきていることに対応して、4社はシステム大規模化への対応、信頼性、可用性の強化と言った面で積極的にオープン・ソース・コミュニティと連携し、共同してプロジェクトへの提案、開発のサポートなどをしていく方針だ。

最初に提案を予定しているプロジェクトは、障害やシステムのさまざまなイベントを検出し解析する問題判別機能の強化で、スケーラビリティや可用性の強化についても検討していく。

なお、本件とは別件で日本にOSDL（Open Source Development Lab）のアジア拠点を設立する計画も進行している。これには、現在のところ上記4社を含む19社が参加を表明しており、オープンソース界への企業の熱意がうかがわれる。

誰もが自由に配布、使用、改良できるということが大前提のオープン・ソースの世界への企業参加に疑問を投げかける人たちもいるかもしれない。しかし、ことの是非はともかくとして、このような変化はLinuxとそれを取り巻くオープンソースの世界が新たな段階に入ったことを示しているのではないだろうか。企業のオープンソース活動への参加がどのような形をとるにせよ、建設的に発展することが望まれる。




OSDLのWebページ
http://www.osdlab.org/

# News Headline from 日刊アスキーLinux

www.Linux24.com

## 米Compaq、Linuxイニシアティブを発表 クラスタソフトをリリース

2001年6月22日

米Compaq Computerは6月18日、従来のオープンソース・コミュニティへの取組みをベースに、新たに6つの戦略的Linuxイニシアティブを発表した。主な内容は以下のとおり。

- ProLiantサーバ上でBeowulfクラスタを提供
- 米OracleのLinuxラボ参画等によるLinuxソリューション開発の拡大
- Linux and Tru64 UNIX Affinity Programの提供
- SSIクラスタソフトのリリース
- Linuxシステムエンジニアの育成と認定
- 企業に向けLinuxソリューションの強化

なかでも大きな特徴として、複数のサーバを単一サーバのように見せかける、「SSI」（シングル・システム・イメージ）クラスタソフトのリリースという点が挙げられる。これにより、ロードバランシング、サーバの拡張／削減、シングルパスワードによるサインオン、ソフトウェアのアップグレード、アプリケーションへのアクセスなど共通のシステムタスクが簡素化される。最初に基盤部分をリリースし、その後、SSIソフトウェア全体をリリースするという。

Compaq Computer
（http://www.compaq.com/）

## 日本IBM、Linuxクラスルームシステムを大東文化大学に構築

2001年6月20日

日本IBMは6月19日、LinuxとWindowsの両方を学習できるクラスルームシステムを、大東文化大学に構築したと発表した。同社のパートナーである日本情報通信がシステムの構築を担当した。

具体的には、Red Hat Linux 7JとWindows 2000のデュアルブート環境となるThinkPad約1000台と、サーバとなる、eServer xSeries 240×2台、Netfinity 5600×14台で構成されている。ThinkPadには、大東文化大学用特別モデルとしてCD-RWを装備した。

そのほかにも、板橋と東松山の2つのキャンパス、約1万4000人の学生のアクセス管理やデータ管理を実現する、Windows 2000 Active Directoryを構築した。

7月からは、同クラスルーム・システムとは別に、ThinkPad約200台を希望する学生に貸与する。また本年度中に、板橋地域の小・中学校における情報システム構築を手伝うボランティアを学生から募集する予定だという。

日本IBM（http://www.ibm.com/jp/）
日本情報通信（http://www.niandc.co.jp/）
大東文化大学
（http://www.daito.ac.jp/top.html）

## チェコのBlokman Traiding、初代PlayStation用Linuxのアルファ版を公開

2001年6月6日

チェコのBlokman Traidingは、PlayStationで動作するLinuxのアルファ版を公開した。

Blokman TraidingのWebサイトによると、これまでMIPSプラットフォームで動作するハード／ソフトウェアの開発を行っており、その際にPlayStationおよびPlayStation 2も開発に使用していたという。しかしPlayStationの開発情報は入手が困難であったため、自社で開発環境を構築。その結果生まれたカーネル2.4ベースの開発環境である「RU.nix」をGPLライセンスで公開した。「このプロジェクトは、Linuxキットを開発することでPlayStationおよびPlayStation 2を低価格のデスクトップコンピュータにすることを目指して」いるという。

ダウンロードはftp://ftp.ware.ru/pub/unix/linux/kernel/kernel-psx-2.4.0.tar.gzから可能になっている。サイズは26Mバイト。展開後に生成されるPSXLinuxディレクトリにあるREADME.PSXによると、必要なソフトウェアは以下のとおり。

**egcs-1.1.2-1 MIPSクロスコンパイラ**
**binutils-2.9.5-1 MIPS**
**Serial Boot for PSX**

インストールの方法については、「Linux/MIPS HOWTO」を参照することになる。ここではMIPSプラットフォーム上にLinuxをインストールする際の一般的な方法を知ることができる。また、ハードウェアとしては、PlayStation用のブートディスクまたはデータ転送用のシリアルケーブルが必要になるという。

RU.nixについてはすでにいくつかのニュースサイトで報じられており、米Slashdotなどでも話題になっているが、インターフェイスがシリアル以外は独自規格であり、拡張性がないことや、対応したキーボードやストレージがない、PlayStationをデスクトップコンピュータとして使用するメリットが感じられない、といった意見も多く、実際にインストールしたという情報は確認できなかった。

ロードマップでは、7月にベータ版を公開、10月には安定版を発表すると同時に、ハードウェアへの拡張メモリや各種ポートといったアドオンキットの供給開始が行われる。年末にはゲーム開発キットおよびオフィスツールを供給し、将来的にはPlayStation 2用の製品も発表するとしている。

Blokman Traiding（チェコ）
（http://www.runix.ru/）

## アットホームジャパン、日本IBM、松下電送システムが、Linuxベースのインターネット動画配信システム構築で協業

2001年6月6日

アットホームジャパン、日本IBM、松下電送システムは6月5日、Linuxをベースにしたインターネット動画配信システムの構築／ソリューションで協業すると発表した。7月より営業活動を開始する。

具体的には、日本IBMのPCサーバ「eServer xSeries」をベースに、アットホームジャパンが持つコンテンツのデジタル

化および圧縮技術と、松下電送システムの導入支援、施工／保守管理サービスを組み合わせた包括的なソリューションを提供する。

3社は、LinuxとPCサーバをベースにした安価なシステムにより、ブロードバンド時代へ向けて、インターネット動画配信システムを導入したい企業のニーズに応えていく。今後は、キャッシングの分野などでも協業する予定だという。

**アットホームジャパン**
**（http://www.jp.home.com/）**
**日本IBM（http://www.ibm.com/jp/）**
**松下電送システム**
**（http://www.panafax.co.jp/）**

### タクシア、インターネットアプライアンス向け組み込みLinux「TASTE」の国内販売を開始
**2001年6月1日**

5月31日、独TUXIAの日本法人であるタクシアは、組み込みLinux「TASTE」の国内販売開始を発表した。

TASTEは、米National Semiconductorの低消費電力型プロセッサであるGeode上で動作するLinuxシステム。カーネルは2.4をベースに、glibc6に対応する。独自の技術で1/3まで圧縮されている。Mozillaベースの組み込みブラウザ「Nanozilla」やさまざまなプラグインのサポートといったものがモジュールとして提供されている。また、組み込みLinux開発環境である「TSE」を用いた、モジュールを組み合わせてのシステム開発が可能。

独TUXIAのCSO、Jean Louis van der Velde氏が世界戦略についてスピーチし、Linuxのマルチメディア対応のために重要なのは「標準化」と「モジュール化」であると述べた。標準化について、TASTEはオープンソースの標準に基づいて開発されていることで、サードパーティによるマルチメディアアプリケーションの開発プラットフォームとして優位にあると同時に、世界各地に展開することによって、各地のパートナー企業との共同開発を行えると述べた。モジュール化についても、製品全体ではなくコンポーネント単位での追加が可能であるとしている。

**タクシア（http://jp.tuxia.com/）**
**独TUXIA（http://www.tuxia.com/）**

### Samba 2.0.9日本語版1.0リリース
**2001年5月29日**

日本Sambaユーザ会は5月28日、Samba 2.0.9日本語版1.0をリリースした。Sambaは、UNIXマシンをWindowsのファイル／プリンタサーバとして機能させるソフトウェア。

オリジナルのSamba 2.0.9をベースに、Samba 2.0.7日本語版で実装されたSWAT（SambaのWebによる管理ツール）の国際化や、Windowsの日本語機種依存文字のサポートなどをそのまま実装している。Samba 2.0.9日本語版独自の特徴としては以下のものが挙げられる。

- **smb.confの日本語テンプレートを同梱**
- **smbclientコマンドでの日本語サポートの強化**
- **HEX、CAPのオペレーションを支援するsmbchartoolの提供**
- **各種セキュリティホールの修正**
- **O'Reilly「Using Samba」の完全邦訳版を同梱**

今後は、Samba 2.0.7日本語版での不具合を修正した日本語版1.1のリリースに向けて開発を行っていく。先月リリースされたオリジナルのSamba 2.2.0は、日本語環境で使用すると不具合が生じるため、日本Sambaユーザ会では、日本語版の開発の成果を元に、Samba 2.2.0の動作検証や改良作業を進めているという。

ダウンロードは、http://www.samba.gr.jp/download.htmlにあるミラーサイトより可能。主要OSのバイナリパッケージも近日中に提供される。

**日本Sambaユーザ会**
**（http://www.samba.gr.jp/）**

### 組み込みLinux新会社イーエルティ、国内組み込みツールベンダーと米MontaVistaにより設立
**2001年5月23日**

国内の組み込みツールベンダーと米MontaVistaにより、組み込みLinuxソリューションプロバイダ、イーエルティ（以下ELT）が5月22日に設立された。

国内の組み込みアプリケーション開発環境の多くが非GNUベースであるという現状をふまえ、米MontaVistaの組み込み専用OS「Hard Hat Linux」と、パートナー各社のコンパイラ、ICE、デバッガなどを包括することで、GNU／非GNUを問わない開発／ランタイム環境を提供する。

国内のパートナーとなるのは、モンタビスタソフトウエアジャパン、ガイオ・テクノロジー、ワイ・ディー・シー、ガイア・システム・ソリューション。

モンタビスタソフトウエアジャパンは、ELTのニーズに応じてHard Hat Linuxのカスタマイズやサポートを行う。両社はリソースをシェアし、開発やマーケティングにおいて協業していく。

ガイオ・テクノロジーは、組み込みソフト開発環境「OPENplus」をHard Hat Linuxに対応させる。同社は、主にiTRONをサポートしてきたが、ハイエンドレベルにおけるHard Hat Linuxのニーズに応え、iTRONのユーザーが従来の操作性のまま移行できるようなコンパイラやデバッガを提供していく。

ワイ・ディー・シーはICEの日本最大メーカーであるという特徴を活かし、デバッグ環境などを提供していく。

エルミックシステムは、組み込みシステム専用のインターネット接続用プロトコルスタック「Kasago TCP/IP」などを扱っており、ELTの通信事業者向けソリューションを主にサポートしていく。

代表取締役社長の坂本氏は、「後発として他社との差別化をはかるため、開発を重視した『ツールのインテグレータ』を目指していきたい」と語っていた。

**米MontaVista（http://www.mvista.com/）**
**モンタビスタソフトウエアジャパン**
**（http://www.montavista.co.jp/）**
**ガイオ・テクノロジー**
**（http://www.gaio.co.jp/）**
**ワイ・ディー・シー**
**（http://www.ydc.co.jp/）**
**エルミックシステム**
**（http://www.elmic.co.jp/）**
**ガイア・システム・ソリューション**
**（http://www.gaiaweb.com/gss/main.html）**

Hardware　発売日　2001年6月23日

## Linuxにも対応する高解像度、低価格インクジェットプリンタ「Z33 Color Jetprinter」他

URL　http://www.lexmark.co.jp/

発売　レックスマーク インターナショナル株式会社
TEL　03-3444-8161
価格　1万7800円〜

レックスマーク インターナショナルは、インクジェットプリンタ「Z」シリーズに「Z33 Color Jetprinter」、「Z23 Color Jetprinter」、「Z13 Color Jetprinter」の3機種を加え、6月23日にZ13を、6月29日にZ23とZ33を発売した。

新機種は、2400dpi×1200dpiで1ドット当り7pLという高解像度を実現。肌の色合いなどを忠実に再現するために日本向けのカラーテーブルを採用した。

Z33は、9ppm（ブラック）／5ppm（カラー）の印字速度が可能。Z33にはカラーインクとブラックインクの2個のカートリッジ、Z23／13にはカラーインクカートリッジが1個付属する。接続インターフェイスはUSBを使用する。対応するLinux OSは、Linux Mandrake 7.2、Caldera 2.5、Red Hat Linux 7.0、SuSE 7.0。Linux用プリンタドライバは同社ホームページからダウンロードできる。写真印刷用のValue Packが同梱されている。

Hardware　発売日　2001年6月29日

## PostScript 3を採用したカラーレーザープリンタ BEAMSTAR-PriusLaser4220

URL　http://www.hitachi.co.jp/beamstar/

発売　株式会社日立製作所
TEL　046-235-8250
価格　39万8000円

日立製作所は、カラーレーザープリンタ「BEAMSTAR」シリーズに、A4用紙に対応した新製品「BAEMSTAR-PriusLaser4220」を追加し、6月29日より発売した。

BEAMSTAR-PriusLaser4220は、従来モデル（BEAMSTAR-NP）に比べ、印刷速度が1.5倍高速化し（カラー印刷6枚／分、モノクロ印刷24枚／分）、両面印刷に対応した（両面ユニットが別途必要）。解像度は600dpi×600dpiまたは1200dpi×600dpi。オプションで本体にHDDを搭載すれば部数単位で出力するマルチプリントが可能。同じデータを何度も取り込む必要がないため多部数の印刷も一括処理できる。

また、PDFとの親和性の高いPostScript 3をページ記述言語として採用したことにより、PDFファイルを効率よく印刷できる。

プラットフォームは、Windows、Macintosh、UNIXに対応しており、各プラットフォームが混在する環境でもBEAMSTAR-PriusLaser4220 1台で印刷できる。

Hardware　発売日　2001年6月初旬

## デスクトップ型NAS製品「Snap Server 2000」日本語版

URL　http://www.snapappliances.com/japan/

発売　日本クアンタム ストレージ株式会社
TEL　03-5321-7948
価格　29万8000円（80Gバイト）

日本クアンタム ストレージは、NAS製品のデスクトップモデル「Snap Server 2000」日本語版を6月初旬より出荷開始した。

Snap Serverは、ネットワークに直結し、簡易な設定を行うだけで、ファイルサーバとしての利用ができるNAS製品。今回出荷するSnap Server 2000は、最大容量80Gバイト、RAIDサポートは0（ストライピング）、1（ミラーリング）をサポート（出荷時デフォルトはRAID 0設定）。ミラーリング使用時は実行容量の半分になる。RAID機能を使わず、2台のディスクドライブを個別に使うこともできる。

本製品には米PowerQuestのバックアップソフトである「DataKeeper」日本語版が同梱されている（Windows用）。Snap ServerはUNIXをルーツに、ファイルサーバ機能に最適化したOSを備えておりボード上のFlashROMに格納されている。

同製品はWindows、NetWare、Macintosh、UNIX／Linuxなどが並存する環境で使用できる。

Hardware　発売日　2001年6月7日

## ブックサイズPC POWERSTEP-Mini Ce-733

URL　http://www.amulet.co.jp/

発売　アミュレット株式会社
TEL　03-5295-8418
価格　6万9800円

アミュレットは、Linux専用機としての実績を持つ、POWERSTEPシリーズの新モデルPOWERSTEP-Mini Ce-733を6月7日から発売した。

POWERSTEP-Miniは、ほぼB4ブックサイズPC。筐体の寸法は78mm（H）×300mm（W）×320mm（D）。基本モデルはCPU Celeron 733MHz、メモリ128Mバイト（最大512Mバイト）、IDE HDD 20Gバイト、ATAPI CD-ROMを搭載している。PCI拡張スロット2基を装備し拡張性も備えている。CPUは、オプションでPentium IIIにアップグレードできる。

OSは、カーネル2.2以降を使用した、Red Hat LinuxやTurbolinuxなどの主要なディストリビューションをサポートしている。

Hardware

発売日 2001年6月末

## クルーソー搭載小型サーバ CS56シリーズ

URL http://www.nec.co.jp/ynec

発売 米沢日本電気株式会社
TEL 03-3798-0701
価格 13万8000円（500台／ロット）～

NEC米沢は、CPUに低消費電力を特徴とするCrusoeを使用した小型サーバ「CS56シリーズ」を6月末より発売した。

CS56はCrusoeを搭載することにより低消費電力化すると同時に、内蔵電源を含め一切の冷却ファンをなくし、静粛性を高めている。HDDは最大2基搭載できる。PCIスロットにRAIDボードを搭載すれば（CS56-003-42モデルでは標準装備）2台のHDD間でミラーリングができる。UPSバッテリを内蔵しており、停電時には自動的にシャットダウンを行い、HDDのクラッシュを防ぐ。

主な仕様はCPU TM5600 600MHz、メモリ64Mバイト（最大192Mバイト）、HDD 20Gバイト（最大40Gバイト×2）、100Base-TX 2基。筐体のサイズは220mm（H）×145mm（W）×240mm（D）。OSはLASER5 Linux 6.5 Secure Server Editionが対応している。

価格は、ベーシックモデルのOEM参考価格。

Hardware

発売日 2001年5月29日

## 2Uラックマウントサーバ VA Linux 2200 Series

URL http://www.valinux.co.jp/systems/

発売 VA Linux Systemsジャパン株式会社
TEL 03-5323-3500
価格 71万8000円～

VA Linux Systemsジャパンは、2UラックマウントサーバVA Linux 2200シリーズの最新版VA Linux 2241／2251を5月29日から発売した。

2241／2251は2個のCPU Pentium III 866MHz／1GHz、メモリ256Mバイト（最大4Gバイト）、SCSI HDD（18～73Gバイト）を2251は最大5個まで、2241は最大4個まで搭載できる。さらに、2251は薄型CD-ROM（24倍速）を標準搭載。2241はテープ装置かCD-ROM（40倍速）のどちらかをオプションで選択できる。HDDのホットスワップに対応するためにRAID構成もサポートしている。

VA Linux 2200シリーズにはIntel EMPリモートサーバ管理ポートが設定されている。ポートはVAのクラスタ管理ソフトウェアVACM（リモートから電源のON／OFF、リセット、リブートを可能にする）に対応している。OSはVA Linux 6.2.4を使用。

Hardware

発売日 2001年6月27日

## 9万8000円の低価格IAサーバ eServer xSeries 200／300

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

発売 日本アイ・ビー・エム株式会社
TEL 0120-04-1992
価格 9万8000円～

日本IBMは、eServer xSeriesの新製品、「eServer xSeries 200」2モデルと「eServer xSeries 300」1モデルを6月27日から発売した。

xSeries 200（写真）はタワー型で、8478-31XモデルがCPU Celeron 800Mhz、メモリ64Mバイト（最大1.5Gバイト）。8478-6GJモデルがPentium III 933MHz、メモリ192Mバイト（最大1.5Gバイト）。IDE HDDはいずれも1基で20.4Gバイト（最大90Gバイト）を装備している。価格は9万8000円から。

xSeries 300は、1Uラックマウント型で、CPU Celeron 800MHz、メモリ128Mバイト（最大1.5Gバイト）、IDE HDD 20.4Gバイト（最大40.8Gバイト）、CD-ROMドライブ付属。故障時にデータを守るServerRAIDアダプタも追加できる。価格は19万8000円から。

サポートしているLinux OSは、xSeries 200がRed Hat Linux 7JとTurboLinux Server 日本語版 6.1。xSeries 300がRed Hat Linux 7.1、とTurboLinux Server 日本語版 6.5。

Hardware

発売日 2001年6月11日

## Itanium搭載サーバ NL Server 6401

URL http://www.nlcomputer.com/

発売 ノーザンライツコンピュータ株式会社
TEL 044-850-2238
価格 250万円～

ノーザンライツコンピュータは、「NL Server」シリーズに64ビットCPU Itaniumを搭載した「NL Server 6401」を6月11日から発売した。

NL Server 6401は、CPU Itanium（733MHz／800MHz）でデュアルCPUに対応。メモリ512Mバイト（最大8Gバイト）、Ultra 160 SCSI HDD 9Gバイト（最大72Gバイト）、ドライブベイ×2、CD-ROM、1.44M FD対応スーパーディスク（媒体容量120Mバイト）を搭載している。筐体は4Uラックマウント型だが、フロアスタンドを使用すればタワー型の設置も可能。

Red Hat Linux 7.1での動作確認済み。NL Server 6401は科学技術計算やシュミレーションなどの高度な処理能力が要求される分野、アプリケーションソフトウェアやドライバの開発者を対象としている。価格は最小構成で250万円。

Software　発売日　2001年5月7日

## インストーラ開発環境 InstallAnywhere 4.0

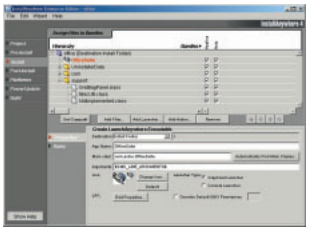
URL　http://www.boc.co.jp/installanywhere/

発売　文化オリエント株式会社
TEL　022-373-0360
価格　14万円〜

文化オリエントは、マルチプラットフォームに対応したインストーラ開発環境InstallAnywhere 4.0を5月7日より発売した。

InstallAnywhereで作成したインストールプログラムは、Java上で動作するため、配布先のシステムに依存しない。また、日本語を含む29言語（Standard Ed.は9言語）に対応し、1つのインストーラで各国語対応できる。操作の簡単なGUIを備えスクリプト等の習得は不要。直感的な操作で高度なカスタマイズができ、短時間でインストーラ開発を可能にした。

対話性を重視したGUIインストーラだが、サーバ環境での作業を支援するコンソールからのインストールやサイレントインストールなどの新しいインストールモードも備えており（Enterprise Ed.のみ）、インストールウィザードの処理ステップも自由に指定できるようになった。動作環境はJava 1.1.8以上、メモリ64Mバイト以上。



Software　発売日　2001年7月7日

## 日本語入力システム Wnn7 Personal for Linux／BSD

URL　http://www.omronsoft.co.jp/

発売　オムロンソフトウェア株式会社
TEL　044-246-6006
価格　9800円

オムロンソフトウェアは、Linux、FreeBSDに対応した日本語入力システム「Wnn7 Personal for Linux／BSD」を7月7日より全国パソコンショップで発売する。

今回Wnn7は入力効率を上げるために、一度入力した言葉を記憶し使用頻度に応じて予測変換するようにすると同時に、逆引き変換、連想変換、よくあるミスを補正する入力補正などの機能を加えた。さらに、辞書を大幅に強化し、基本辞書を全面的に見直した最新辞書（約50万語）を搭載。サーバで辞書を一括管理してメモリやディスク容量を節約できる。Wnnの環境設定機能を提供するxwnmoはGTK+で作成されており、カスタマイズが可能。対応するLinux OSはRed Hat Linux 7.1、Turbolinux日本語版6.0、LASER5 Linux 6.4、Vine Linux 2.1.5、Kondara MNU／Linux 2000。ディスク容量60Mバイト以上、メモリ64Mバイト以上が必要。


Software　発売日　2001年5月16日

## Webストアフロント構築を支援 AbleCommerce 3.0

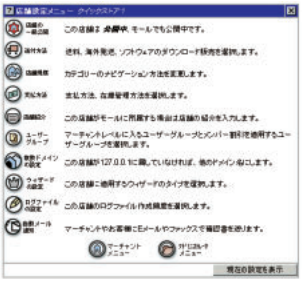
URL　http://www.ablecommerce.co.jp/

発売　株式会社エーブルコマース・ジャパン
TEL　03-5775-6594
価格　17万5000円（Windows NT版）

エーブルコマース・ジャパンは、Webストアフロント構築のためのソフトウェアAbleCommerce 3.0を5月16日から発売した。現在発売されているものはWindows NT版だが、近くLinux版も発売する予定。

オンラインストアに必要な、リアルタイムでの決済処理、価格設定や商品構成に関わるストアフロントの管理、ポイントカードシステム、FAX、電話による注文受付、トラフィックや顧客の購入パターンのレポートや分析、eコマースサイト構築に使うテンプレートとWYSIWIGエディタ、企業の基幹業務を一元管理するERPシステムとの統合、モバイル対応などの機能を装備。Webデザインは、ウィザードで変更できる。

本製品はCold Fusionで開発されているため、既存のデータベースとの統合が簡単にできる（Cold Fusion Professional版が必要）。本製品には、アド・オン、プラグインやソースコード版もあり、より柔軟な製品のカスタマイズにも対応している。



Software　発売日　2001年7月（予定）

## BtoC向け電子商取引サイト構築ソフト Dynamic Personal WebShop 1.2 Intro（PostgreSQL version）

URL　http://www.azi.co.jp/

発売　クラスキャット ネットワークス株式会社
TEL　-
価格　約25万円〜（推定価格）

クラスキャット ネットワークスは、ECサイト構築ソフトウェアの新製品Linux対応「Dynamic Personal WebShop 1.2 Intro（PostgreSQL version）」を7月をめどに発売を予定している。

今回発表した製品は、同社「Personal WebShop」シリーズの新製品で、JSP（JavaServer Pages）およびJavaBeansテクノロジーをベースにオンラインショップのデザイン部分（JSPを利用）と、運用・管理を行うプログラム部分（JavaBeans）を分離し、ショップのデザイン変更を可能にし、同時に運用・管理に携わるプログラム部分はコンポーネントベースで拡張できるようにしたパッケージソフトウェア。

オンラインショップの運用・管理に関しては、すべてWebブラウザで行え、専門的な知識は不要。受注管理はデータベースに登録、管理する。

Software　発売日　2001年5月21日

## 画像ファイルの運用・管理をサポート MGI Zoom Image Server

URL　http://www.livepicture.co.jp/product/

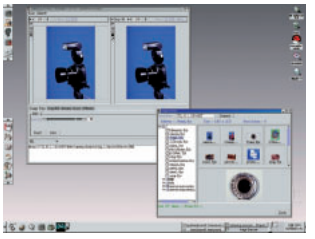
発売　ライブピクチャージャパン株式会社
TEL　03-5545-3241
価格　96万円〜

ライブピクチャージャパンは、企業で保持する画像ファイルを資産として活用するための画像配信／管理サーバソフト「MGI Zoom Image Server」を5月21日より発売した。

本製品は、高精細画像をインターネットで高速配信するソフト「Live Picture Image Server 3.0」の基本機能を引き継いだ新製品。1枚の画像を元に、仮想的にサイズを変更、1部切り出し、回転、色味調整などの処理を施して複数に配信して、画像資源を有効活用できる。PC操作に不慣れなインターネットユーザも、プラグインなしで画像の閲覧をできるようにしてある。

またJDBCによるデータベースとのシームレスな連携が可能。Image Browserにより、遠隔からでもサーバに格納された画像情報をサムネール画像で見ながら確認できる。

対応OSは、Red Hat Linux 6.2（7は動作確認済み）。サポートデータベースは、DB2、Oracle8i。



Software　発売日　2001年6月6日

## セキュアなインターネットログインを確保 SSH Secure Shell 3.0

URL　http://www.dit.co.jp/ssh/

発売　株式会社ディアイティ
TEL　03-5634-7651
価格　6万9900円〜

ディアイティは、次世代SSH Secure Shell 3.0日本語版を6月6日より発売した。

SSH Secure Shell は、インターネットからパスワードを盗むという最もよくあるハッカーの手口から、インターネットユーザやサーバを保護するための代表的な暗号化技術。

今回発売される製品は、X.509電子証明をサポート。PKI（公開鍵インフラストラクチャ）環境に統合され、ログインと認証プロセスのセキュリティが拡張されている。

さらに、ユーザーの機密情報を安全に保存するスマートカードに対応、機密性の高い政府情報の保護に使用されていたRijndael暗号化アルゴリズム、LinuxやSolarisでのサインオンの認証サービスのPAM、MITが開発したネットワーク認証プロトコルKerberos5などもサポートするようになった。

Software　発売日　2001年6月1日

## メール配信システムのスタンダード Sendmail SWITCH 2.1 日本語版

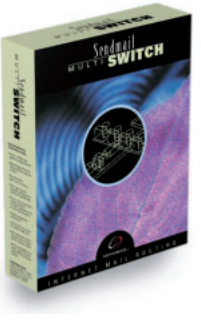
URL　http://www.trans-cosmos.co.jp/

発売　トランス・コスモス株式会社
TEL　03-5561-0152
価格　69万8000円〜

トランス・コスモスは、メールの配信、転送処理を行うメール・サーバソフトウェア「Sendmail SWITCH 2.1」日本語版を6月1日より発売した。

Sendmail SWITCH 2.1は、フリーソフトウェアsendmailをベースにした商用版ソフトで、今回のバージョンでは運用管理やセキュリティ面での機能が強化された。Sendmail SWITCH 2.1にはメール配送システムの中核MTAと設定／管理用GUIツールを一体化したSendmail Single SWITCH、MTAとGUIツールを別にしてメールサーバとしてのセキュリティを強化したSendmail Secure SWITCH、複数のメールサーバの設定／管理を一元化するSendmail Multi SWITCH、これらのオプション製品である特定の添付ファイルに対するフィルタ制御機能を追加するSendmail SWITCH Attachment Filterからなる製品群。

Sendmail SWITCH 2.1は、設定、管理が日本語化したGUIから行え、多彩なフィルタ機能が追加された。対応するOSはRed Hat Linux 6.1／6.2／7.0。




Software　発売日　2001年6月20日

## モバイル端末用、Webコンテンツ変換ソフト CAFEMOON C3GATE Server Version 1.1

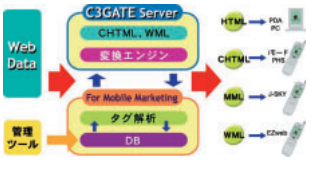
URL　http://www.vertexlink.co.jp/

発売　株式会社バーテックス リンク
TEL　03-5259-5158
価格　50万円（1CPU）〜

バーテックス リンクは、同社の製品「CAFEMOON C3GATE Server」をバージョンアップし、6月20日より発売した。

CAFEMOON C3GATE Serverは、インターネット対応携帯電話、モバイル端末などで異なるWebコンテンツを動的に変換し、複数のインターネット端末に1つのコンテンツで対応できるようにするゲートウェイサーバソフトウェア。

Version 1.1では、iモード用コンテンツ（cHTML）からEZWeb、J-SKY用コンテンツ（WML、MML）に動的に変換する、iモードコンテンツ変換機能、iモードでのアクセスは、直接iモード用コンテンツに配信するiモードスルーモード、EZWeb、J-SKY対応携帯電話の容量制限を越えるcHTMLを自動的に複数ページに分割するページオプティマイズ機能などが追加された。



# 日刊アスキー Linux Business Report

## 日本オラクル iDevelop 2001

http://www.linux24.com/

### Linuxをエンタープライズのプラットフォームとしても認める

ミラクル・リナックスの設立やOracle MasterのLinux資格対応など、従来から強力なLinux推進を行ってきたオラクルだが、先日のLinuxWorldにおける展示内容や、6月20日に都内で行われた『Oracle Linux Summit 2001』のもようを見ていくと、『Oracle Parallel Server』を使用するような、エンタープライズ向けの市場においてもLinuxをプラットフォームのひとつとして位置づけ始めたことがありありとわかるようだった。

言ってみれば、オラクルにとってLinuxは、新しく盛り上がりつつある市場でも、中小規模なシステム向けのOSでもなく、エンタープライズでも使えるような、UNIX互換OSのひとつということになるだろう。こうした現状はまた、Linux関連業界がエンタープライズへのひとつの足がかりを掴んだということにもなる。

### 開発者向けの技術色濃いイベント

今回は、来る9月6日～7日、神奈川県横浜市の、パシフィコ横浜（みなとみらい地区）にて日本オラクル主催で行われるOracleのテクニカルカンファレンス「iDevelop 2001」を紹介する。

iDevelopは、3年前の1999年、米Oracleが開発者を対象に、新技術の内容を広めていくことを目的として開催したのが始まりだ。日本発のイベントであるOracle Open World（OOW）は、どちらかというとOralceの技術を一般の人にも広くわかりやすく伝えているのに対し、iDevelopのほうはあくまでも技術色が濃いイベントとなっている。またOOWは基本的に無償のイベントであるが（米国では1000ドル程度の参加費が必要）、iDevelopは有償のセミナーであり、より深い内容、より技術的に突っ込んだセッションが用意されている。

そして、iDevelopを見ていくうえで忘れてはいけないのが、「OTN」の存在だ。OTNは、「Oralce Technology Network Japan」の略で、インターネット上のOracle開発者向けコミュニティで、無償で参加することができる。OTNの会員になると、技術資料やソフトウェアのダウンロードサービス、サンプルコード閲覧、初心者向け講座、Oracle関連書籍オンライン販売、メールサービス、フォーラムへの参加と、さまざまな特典を得ることができる。国内では約1年前に立ち上がり、現在はなんと6万人の会員が在籍する一大ネットワークとなっている。そして、iDevelopは、OTNのリアルにおけるイベントという位置づけも持っているのである。

### 今年は9iの話題が盛りだくさん

今年のiDevelopは、やはり登場間近のOracle 9iの話題が中心になりそうだ。米国などでの発表に続いて、先日のOracle Linux Summit 2001でも紹介され、国内でも徐々にその概要が広まりつつある新バージョンの9iだが、このiDevelopで初めて、深い技術的な内容が披露される。

Oracle9iは、今までのOracleデータベースとは違い、「E-Businessを実現するプラットフォーム」としての製品となっている。つまり、今までは単独のRDBMSを『Oracle』と呼んでいたわけだが、9iでは大きく変わり、『Oracle9i Database』と、『Oracle9i Application Server』の2コンポーネントで構成される。現在のWeb上におけるビジネスシステムの構成に則した形での製品形態となっているわけだ。

個々のコンポーネントの大まかな特徴は、まずDatabaseのほうはデータベースカーネルにJavaVMを組み込んだほか、XDBへの対応やマルチメディアデータ対応など、従来のインターネット対応路線を一層推し進めた形となっている。

また、もうひとつの大きなコンセプトとして、Oracle Parallel Serverの進化型であるOracle Real Application Clusterの実装をはじめとした、「ラージE-Business」への対応があげられる。今までのインターネット対応データベースという性格から、今度はE-Business上のサービスを視野に入れた展開を図っていく。

一方、Application Serverのほうは、ApacheをベースとしたWebサーバを備えた、Javaの実行環境である。今回の9iでは、大量アクセスへの対処として、Oracle Web Cacheが実装さ

れた。

iDevelopでは、E-Business環境でどういったソリューションが最適なのか、といった話題から、プロダクト自体の強化ポイントなどについて、すでにOracle9iをテストなどで使用した技術者が講演する。

## 多彩なセッション

セッションは、「Product」、「Development」、「Management」、「Solution」4つのトラックに分けられている。セッションリストは表1を見ていただくとして、その内容を少しだけ紹介すると、たとえばProductトラックでは、Oracle9i関連では、Oracle9iのキーテクノロジーや戦略をデモンストレーションを交えて解説する「E-Businessプラットフォームの新基準ーOracle9iー」セッションをはじめ、Oracleのクラスタリングテクノロジーを解説する「Real Application Clusterで実現する可用性とスケーラビリティの両立」セッション、XMLのネイティブ格納を実現する新しいデータ型の概要と使用法を解説する「Oralce 9i強化ポイントーXML完全対応ー」セッションなどが行われる。

Developmentトラックには、Oracle 9i Application ServerのJ2EEコンテナ「OC4J」上で動作させるJ2EEアプリケーションの構築技法を扱う「Oracle9iAS Containers for J2EEによるJ2EEアプリケーション開発」が用意される。

また、Solutionトラックにおいては、高可用性システムを設計、構築する際に見落としがちな穴とその埋め方について解説する「高可用性七つの落とし穴」セッションがある。これらはすべてOracle9iをベースに、さらなるスキルアップを目指す技術者のための講座となっているわけだ。

●

徐々に大規模事例も紹介され始めているLinuxだが、現在はまだ単純なWebサーバやメールサーバとしての利用がメインであり、商用のRDBMSが動作しているWebアプリケーション事例というのは、ほかのプラットフォームと比べると少ない。こうした中で、Oracle9iの登場がLinuxの広がりを促進することは間違いないだろう。

| セッション名 | トラック |
|---|---|
| Product | Real Application Clustersで実現する可用性とスケーラビリティの両立 |
| | Oracle9i強化ポイント　ー計画停止ゼロ／オンライン・メンテナンス操作ー |
| | Oracle9i強化ポイント　ー高度なアクセス制御技術ー |
| | Oracle9i強化ポイント　ーXML完全対応ー |
| | Oracle9i強化ポイント　ーB2Bのための機能ー |
| | Oracle9i強化ポイント　ー大切なデータを失わないためにー |
| | Internet File System |
| | OLAP Server アーキテクチャとチューニング Tips |
| | Oracle9i強化ポイント　ーデータウェアハウスー |
| | Oracle9iAS Containers for J2EE　ーOracleの新しいJavaプラットフォームー |
| | Java+XML+SQLで行うWebシステム・サービス開発 |
| | Discoverer4i活用法　ーWebからのデータ分析ー |
| | E-Businessプラットフォームの新基準　ーOracle9iー |
| Development | Oracle Developer　～失敗しないWebへの移行～ |
| | チーム開発における情報の一元管理 |
| | Oracle9iAS Containers for J2EEによるJ2EEアプリケーション開発 |
| | Oracle9iAS Portal によるEIP構築 |
| | Oracle9iAS Wirelessによるマルチデバイス対応アプリケーションの開発 |
| | Oracle9i Liteによるモバイルアプリケーション開発技法 |
| | Oracle9i に最適化されたカーネル2.4　ーMiracle Linux V2.0の全容ー |
| Management | GUIで行うOracle9iプラットフォーム統合管理 |
| | GUIで行うパフォーマンスチューニング |
| | 最新Oracle9iASシステム構築技法 |
| | Oracleパフォーマンスチューニング |
| | システムマネージメントTips集 |
| Solution | Oracle9iで実現するセキュアなPKIシステム |
| | Oracle9iで実現するE-Businessワークフロー |
| | 高可用性七つの落とし穴 |
| | ECOstructureで実践！　E-Bussinessインフラストラクチャの構築 |
| | Oracle Web Services |

表1　iDevelop 2001 セッションリスト
※セッションは一部変更されることがあります

## iDevelop 2001

http://otn.oracle.co.jp/idevelop/index.html

**開催日：2001年9月6日～9月7日**

**会場：パシフィコ横浜**

### 参加料金

| 資格 | 7月31日までに登録 | 8月31日までに登録 | 通常料金 |
|---|---|---|---|
| ORACLE MASTER Platinum<br>Certified Developer | 5万8000円 | 6万8000円 | 7万8000円 |
| ORACLE MASTER Gold<br>ORACLE MASTER Linux+<br>ORACLE MASTER Silver<br>OTN Professional<br>一般<br>グループ登録（5名以上） | 6万8000円 | | |

# Distribution

新着ディストリビューション

## Plamo Linux 2.2.1

あるユーザーが使いやすいと考える環境を配布（ディストリビュート）するのはごく自然な行為だ。商用ディストリビューションは、多くの人を満足させるために、ある人にとっては必要だが、またある人にとっては不要なソフトウェアも収録している。こじまみつひろ氏が開発するPlamo Linuxは、そんなディストリビューションの対極に位置する存在だ。

## Turbolinux Workstation 7(Monza) Beta

いよいよオフィスでも家庭でもLinuxの時代がくるのか。Turbolinuxは、古くからLinuxの日本語化に力を入れてきたディストリビューターだ。新しいカーネルが多くのハードウェアをサポートし、GNOMEやKDEがWindowsに匹敵するGUI環境を提供している今、新しいTurbolinuxには、日本人がそれらの技術を享受できる環境を構築してくれるよう、大きな期待がかかっている。

## Kondara MNU/Linux 2.0

神速は譲らず。ここのところ新しいバージョンがご無沙汰だったKondara MNU/Linuxだが、6月9日にリリースされたKondara MNU/Linux 2.0は、自身がもっともエッジの効いたディストリビューションであることを証明してくれた。常に最先端を走り続けるディストリビューションの実力を垣間見てみよう。

## SuSE Linux 7.2-International-2

目的を達成することが第一。SuSE Linuxの特徴であるYaST2は、そんな哲学を体現した強力なGUI設定ツールである。古来よりUNIXの特徴はその無口さにあったが、Linuxに期待する多くのコンシューマを裏切らないように、SuSE Linuxは、あえてその思想にアンチテーゼを示しているかのようだ。

# Plamo Linux 2.2.1

こじまみつひろ氏が開発するディストリビューションの最新版Plamo Linux 2.2.1（以下、Plamo 2.2.1）が6月10日にリリースされた。

Plamo 2.2.1は、Plamo 2.1で採用されていたカーネル2.2.16が2.2.19に、glibc2.1.94が2.2.2に、XFree86 3.3.6がi810グラフィックチップセットをサポートした同じXFree86 3.3.6に変更されたものだ。

今回採用されたカーネル2.2.19には、ジャーナリングファイルシステムReiserFS用のパッチと、IDEパッチが適用されている。

## Plamo Linuxってなにさ？

現在ではRed Hat Linuxが、Linuxディストリビュータの代名詞的な存在となった感があるが、Red Hat Linuxが一般的になる前までは、Slackwareというディストリビューションがポピュラーであった。

しかし、Slackwaweは日本語環境として使うことがまったく考慮されておらず、日本のLinuxユーザーの間では、Slackwareを日本語化するための、JE（のちにPJEと改名）というパッケージが開発されていた。

こじま氏もJEを利用して日本語化したSlackwareユーザーの1人だったが、毎回Linuxをインストールするたびに、JEをインストールするのが面倒くさいということで、SlackwareとJEで重複するパッケージをまとめて、Plamo Linuxを作り始めた。これが、Plamo Linux誕生の由来だ。

SlackwareとJEを足し合わせてPlamo Linuxが誕生したのだが、英語版のRed Hat LinuxとJEを足し合わせて作られたのが、読者もよくご存知のVine Linuxなのだ。

Slackwareから派生したPlamo Linuxは、Red Hat系ディストリビューションで使われているRPMや、Debian系ディストリビューションで使われているdebといったパッケージ管理システムを持たない。

RPMパッケージしかインストールしたことのない読者にはピンとこないかもしれないが、パッケージ管理システムを持たないSlackwareやPlamo Linuxにアプリケーションを追加インストールする際は、アプリケーションのソースを入手してユーザー自身がコンパイル後にインストールするのだ。

アプリケーションのバージョンアップは、実行ファイルや文書がインストールされたディレクトリを思い出し、古いファイルを削除したうえで行う。

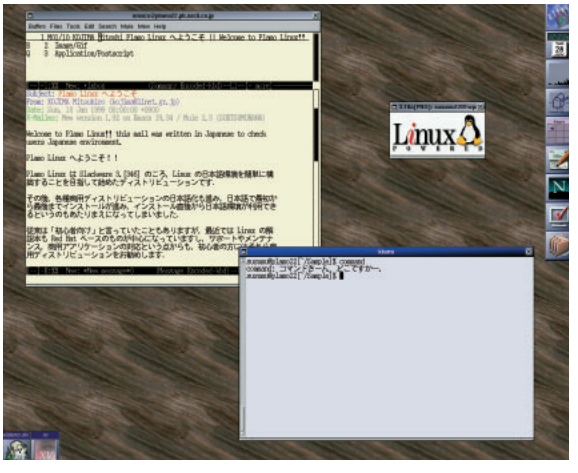
## どんな人が使ってるの？

Plamo Linuxのデスクトップ画面（**画面1**）を見ると、Plamo Linuxのユーザー層が見えてくる。

AfterStepウィンドウマネージャの画面上には、日本語ターミナルkterm、Emacs上で動作するメーラMew、WebブラウザのNetscapeのショートカットが配置されている。

これらは、比較的新しいUNIX系OSのユーザーにとって3種の神器ともいえるものだ。GNOMEやKDEは使っているけど、なぜか最初にktermとEmacsを立ち上げてしまう人達が、Plamo Linuxのユーザーに多いタイプなのだろう。

Red Hat系のディストリビューションからLinuxに入ったけど、RPMがなにをやっているのかがわからなくて気持ち悪いという人は、日本語環境の構築は面倒でないPlamo Linuxを使ってみてはどうだろうか。さいわいにも、ちまたに多く出回っているi810グラフィックチップに対応しているのだから。



**画面1　Plamo Linuxのデスクトップ画面**
Plamo Linuxには、3種類のWindowマネージャが用意されていて、画面はその中でも一番派手なAfterStepだ。それでもGNOMEやKDEに慣れきったユーザーには地味に写るだろう。

| ディストリビューション | URL |
|---|---|
| Plamo Linux | http://www.linet.gr.jp/~kojima/Plamo/ |
| Slackware | http://www.slackware.com/ |

**表1　Plamo LinuxとSlackware LinuxのURL**

# Turbolinux Workstation 7(Monza) Beta

ターボリナックスジャパンから、デスクトップ用Turbolinuxの次期バージョンとなるTurbolinux Workstation 7(Monza) Beta（以下、Workstation 7β）がFTPサイトに公開された。

Workstation 7βは、2.4カーネルなど、基本システムに最新バージョンが採用されており、現行のバージョン6を大幅に刷新する内容となっている。正式リリース前のバグ出しが目的のWorkstation 7βは、その内容が日ごとに更新されている段階だ。また、Workstation 7βはバイナリCDが2枚組になっている。

## 基本部分は総入れ替え

Workstation 7βは、現行のバージョンと比べて、カーネルなどの基本部分が大きく更新されている。

### ■最新のカーネル2.4

現在のところ、採用されているカーネルは最新バージョンの2.4.5だ。このカーネルは、IPv6対応のためのUSAGIパッチや、IBMの開発するジャーナリングファイルシステムJFSに対応するためのパッチなどが当てられ独自に拡張されている。

カーネル2.4の採用でUSB機器にも対応できるWorkstation 7βは、オフィス環境だけでなく、家庭内などでも広く使われることを目指している。

### ■Mongoose採用

Workstation 7βには、Turbolinux Server 6.5で採用されたMongooseというグラフィカルなインストーラが採用されている。前バージョンまで採用されていたテキストベースのインストーラも、使いやすいものであったが、今回からのMongooseの採用で、Xの設定などがより簡単になり、とくに初心者にとってインストールしやすくなっている。現時点のWorkstation 7βでは、インストール時に、標準のext2ファイルシステムのほかにも、ext3、ReiserFS、JFSといった新しいファイルシステムも選択できるようになっている。

### ■FHSに準拠

FHS（Filesystem Hierarchy Standard）は、各Linuxのディストリビューションだけでなく、UNIX系OSのディレクトリ構成を統一しようという指針だ。現在、同じLinuxでもディストリビューションが違うと、デーモン起動用のスクリプトが異なるディレクトリに格納される場合がある。

これは、ディストリビューションによって、FHSのバージョンアップに追従していたり、していなかったりするのが理由だ。

Workstation 7βは、新しいFHSに対応したので、古いFHSを採用していたバージョン6と若干ディレクトリ構成が異なる。Tubolinuxユーザーはこの点を気にとめておく必要があるだろう。

FHSのバージョンが上がったことによって、ドキュメントは/usr/docから/usr/share/docへ、Apacheなどのデーモンを起動させるスクリプトは、/etc/rc.d/init.d/から/etc/init.dへ、パッケージのインストール後にファイルが頻繁に更新されるディレクトリは、/var以下に移されている。たとえばApacheのドキュメントルートは、FHSに従い/home/httpdから

**本誌付録CD-ROMのTurbolinux Workstation 7(Monza) Betaは正式版リリース前のベータ版です。非商用ソフトだけが含まれており、ターボリナックスジャパン株式会社からサポートを受けることはできません。Turbolinux Workstation 7(Monza) Betaのフィードバックは、（http://the.turbolinux.co.jp/bugzilla/）で受け付けております。不具合など発見されましたらぜひお知らせください。**


画面1　turbofscfg
turbofscfgは、パーティションのマウントポイントなどを設定する/etc/fstabと、ブートローダ/etc/lilo.confを編集するテキストベースの設定ツールだ。

| 基本システム | 7(Monza) Beta | 6.0 |
|---|---|---|
| カーネル | 2.4.5 | 2.2.16 |
| glibc | 2.2.3 | 2.1.3 |
| XFree86 | 4.1.0 | 3.3.6 |
| GCC | 2.95.3 | 2.95.2 |
| GNOME | 1.4.0.4 | 1.0.55 |
| KDE | 2.1.1 | 1.1.2 |
| RPM | 4.0.2 | 3.0.3 |

表1　Workstation 7βとWorkstation 6で採用されている基本システムのバージョン

| ツール名 | 概要 |
|---|---|
| turbodhcpcfg | DHCPサーバを設定する |
| turbofscfg | マウントポイントやLILOを設定する |
| turbohwdetect | システムのハードウェア情報を取得する |
| turbolangsel | デフォルトの言語を選択する |
| turbonetroutescfg | 静的ルーティング情報やパケットフォワードを設定する |
| turbonfsexportcfg | NFSを設定する |
| turboprobe | ハードウェアを自動認識して得た情報をファイルに記録する |
| turbousercfg | ユーザーとグループを設定する |

表2　Workstation 7βで新たに収録された独自の設定ツール
これらテキストベースのツールを使って、Linuxのシステム管理を行う。

/var/wwwへ移った。

## 充実した独自設定ツール

Turbolinuxの特徴のひとつに、コマンド名がturboで始まるTurbolinux独自の設定ツールがある。

バージョン6と比べると、Workstation 7βには**表2**のツールが新たに加わっている。たとえば、turbofscfg（**画面1**）は、ディスクのパーティションや、CD-ROMドライブなどを、それぞれどこにマウントするかといった情報を定義する/etc/fstabと、起動時に読み込むカーネルなどを指定する/etc/lilo.confを編集するツールだ。

このほかにもTurbolinuxの独自ツールは多数収録されているが、現時点で、それぞれのツールは、前バージョンにあったTurboCentroのようなツールでひとつにまとめられているわけではない。正式版リリース時には、これらのツールがTurboCentroのようなツールで統合されることを期待したい。

## Linuxをオフィスで使うには？

Workstation 7βには、GNOME 1.4とKDE 2.1.1（**画面2**）といった新しいバージョンのデスクトップ環境が採用されている。新しいGNOMEにもKDEにも、それぞれNautilusやKonquerという名前のGUIシェル環境が統合されているので、初心者ユーザーがコマンドを意識することなくLinuxを使っていけるようになるだろう。

さて、Linuxをオフィスや家庭で使うには、ExcelやWordといったオフィスソフトがどうしても必要になる。

GNOMEもKDEも、それぞれの環境にオフィスソフトを充実させることに力を入れていて、たとえば、Workstation 7βにも、GNOMEにはGnumeric（**画面3**）、KDEにはKSpreadというExcelに相当するツールが収録されている。

記事中にある**表1**のファイルはExcelで作成したもので、試しにこのファイルを、GnumericとKSpreadで開いてみたところ、Gnumericではファイルを開けたが、KSpreadでは、開いた直後に異常終了してしまった。

とくにオフィスでは、Microsoft Officeで作成されたファイルを日常的に編集する機会が多いので、Turbolinux Workstation 7の正式リリース時に、どこまでこれらのツールの完成度が上げられていくのか楽しみなところだ。

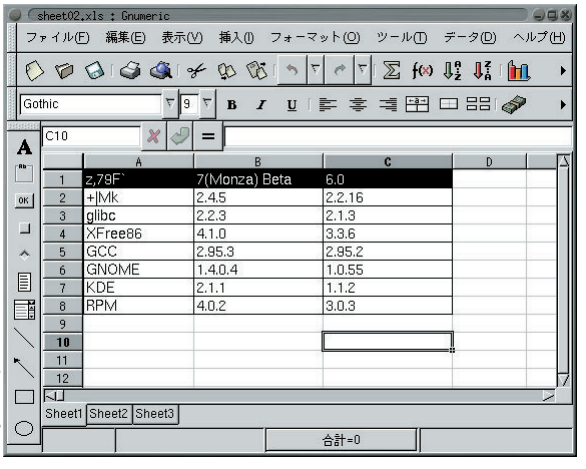
## あとは正式リリースを待つのみ

新しいTurbolinuxは、基本システムやデスクトップ環境が大きく変更されることがわかった。

新しい2.4カーネルではサポートされるハードウェアが増え、Workstation 7βで採用されたGNOMEやKDEには、オフィス環境に必要なツールがひと通りそろっている。

古くからLinuxの日本語化に力を入れてきたターボリナックスには、これらの環境をベースにして、Linuxを日本のオフィスや一般家庭で使えるレベルに引き上げることを大きく期待したいところだ。


画面2　KDE 2.1.1のスクリーンショット
使い勝手はさらにWindowsに近くなった。標準のWebブラウザにはMozillaが採用されている。




画面3　Gnumericで、Microsoft Excelのファイルを開いたところ。日本語部分だけでなく、英語部分も若干文字化けしている。

# Kondara MNU/Linux 2.0

最新ソフトウェアのパッケージ化や、バグへの対処で高い機動力を誇る日本語対応ディストリビューションKondara MNU/Linuxの最新版2.0（以下、Kondara 2.0）がリリースされた。

6月9日のFTP公開に続き、商用ソフトやサポートを含めた製品版も、6月29日から販売が開始された。


画面1　アプリケーションインストーラ
パッケージのインストール直後に、このプログラムが起動される。ユーザーはこのプログラムを使って、商用ソフトをインストールする。

## Kondaraらしい最新構成

Kondara 2.0の基本システムは、カーネル2.4.4、glibc2.2.2、XFree86 4.0.3と、Kondara 2000を含めた1.2系のKondaraから大幅にバージョンアップされている。

なかでも、カーネルはバージョンが新しいというだけでなく、USAGI Projectのパッチを組み込んで、次世代インターネットプロトコルとして期待されているIPv6に対応したり、ネットワークを流れる情報を、IP層という深いレベルで暗号化するIPSecにも標準対応するなど、ネットワーク関係の最新技術への対応度も高い。

また、UNIX系OSが同じディレクリ構成をとるように目指して作られた、FHS（Filesystem Hierarchy Standard）の新しいバージョンに準拠したので、デーモン起動スクリプトなどを格納するディレクリが変更されている。


製品名　Kondara MNU/Linux 2.0
価格　1万2800円
（アップグレード版は9800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/

## 無制限サポート

Kondara 2.0の製品版は、**表1**のような収録物で構成されている。

製品版の一番の特徴は、件数、質問内容それぞれについて制限のないユーザーサポートだ。製品版ディストリビューションに付属するサポートは、インストールに関してだけとか、質問は数件までといった制限があるのが普通だが、Kondara 2.0に付属するサポートにはこれらの制限がなく、ユーザーは次期バージョンの発売日まで何度でも、どんなことでも質問できる。

Linuxを触り始めてみたものの、周りにはLinuxに詳しい人がおらず、同じ疑問のところをぐるぐるまわって抜け出せないLinux初心者は多いはずだ。このようなLinuxユーザーにとって、Kondara 2.0付属のサポートは、まさに天からの恵みの雨となるだろう。

## GUIなきインストーラ

Kondara 2.0のインストーラは、なんとテキストモードのみである。

Red Hat Linuxを始めとして、昨今のディストリビューションが軒並み

| 収録物 | 概要 |
|---|---|
| PC/AT Binary CD（2枚） | PC/AT用のインストールCD |
| Alpha Systems Binary CD（2枚） | Alphaマシン用のインストールCD |
| Source CD（2枚） | PC/AT用とAlpha用で共通のソースパッケージ集 |
| Bundle Software CD | 翻訳魂 Ver1.0、DynaLabフォント5書体、Wnn6 Ver3.02、System Commander Liteなどの商用ソフトを収録 |
| Development Tools CD | Compaqが提供するAlpha用のFortran、C、C++コンパイラと数値計算ライブラリ |
| インストールガイド | Kondara MNU/Linuxのインストールを解説 |
| リファレンスガイド | インターネット接続やWebブラウザとメーラの使用方法などを解説 |

表1　Kondara MNU/Linux 2.0の収録物
Compaqが提供するAlpha用のコンパイラをバンドルするところなどは、KondaraがAlphaユーザーに高い支持を受けているゆえんだ。

| サーバ名 | バージョン（旧バージョン） | 概要 |
|---|---|---|
| Apache | 1.3.19（1.3.12） | もっともポピュラーなWebサーバ |
| PHP | 4.0.5（3.0.15） | Webサーバとデータベースを連携する言語 |
| BIND | 9.1.2（8.2.2pl5） | ドメイン管理サーバ |
| OpenSSH | 2.9p1（なし） | telnetに代わりセキュアな通信を実現する |
| Postfix | 20010228pl02（sendmail 8.9.3） | 設定が容易なsendmailとの互換性が高いMTA |
| PostgreSQL | 7.0.3-6（6.5.2） | 代表的なオープンソースのRDBMS |
| ProFTPD | 1.2.1（WU-FTPD 2.6.0） | 高機能と高い安全性が特徴のFTPサーバ |
| Samba | 2.0.9（2.0.6） | Windowsとのファイルやプリンタ共有を実現する |
| Squid | 2.4（2.2.STABLE5） | Webキャッシュサーバ |
| xinetd | 2.1.8.9（inetd 0.16） | 柔軟な設定と高負荷に対応するスーパーサーバ |

表2　Kondara 2.0に収録されるおもなサーバアプリケーション
（）内は、Kondara Zooを含めて、バージョン1.2系のKondaraに収録されていたバージョン、もしくは対応するアプリケーション名だ。

GUIインストーラを採用していることを考えると、インストーラがテキストモードのみというのは、時代に逆行していると言えなくもない。

しかし、開発にかかる工数と比べて、ユーザーが得るメリットがあまり大きくないという判断から、あえてインストーラからGUIモードが外されたのかもしれない。GUIインストーラに大きな工数がかかっているとすれば、それを外すことで、新しいバージョンのKondaraをより早くリリースできるからだ。

このほかにも、Kondara 2.0では製品版に付属する商用ソフトのインストールを、パッケージインストールの直後にまわすことで（**画面1**）、インストール用のバイナリCDを製品版とFTP版で共通化している。

また、インストール中にジャーナリングファイルシステムReiserFSを利用できることも、これまでどおり継承されている。

## サーバアプリケーションも最新構成

Kondara 2.0では、採用されるサーバ用のアプリケーションが**表2**のように変更されている。

メールを配送するMTAはKondara 1.2系時代のsendmailからPostfixに、FTPサーバプログラムはWU-FTPDからProFTDに、サーバプログラムを一括管理するスーパーサーバはinetdからxinetdに変更され、機能面と使いやすさの両面が改善されている。

また、ドメインを管理するためのBINDや、リレーショナルデータベースとWebサーバの連携システムの開発に必要なPostgreSQLとPHPは、メジャーバージョンの上がったものが収録されている。

さらに、Kondaraに収録されたWebキャッシュサーバのSquid 2.4は、キャッシュディレクトリごとにひとつのデーモンが張り付く新機能DISKDを備えており、応答速度が注目されるところだ。このほかにも、Dynamic DNSを利用するためのdhidや、HTTP経由で流れてくるデータから、特定のファイルのみを削除するデーモンプログラムjunkbusterなどの収録は、常時接続時代の一端を感じさせる。

## GNOME 1.4＆KDE 2.2.0

Kondara 2.0のデスクトップ環境には、GNOME 1.4（**画面2**）とKDE 2.2.0（**画面3**）と、どちらも最新バージョンが採用されている。これらのバージョンからは、GNOMEにはNautilusが、KDEにはKonquerorがGUIシェルとして統合され、ユーザーはより操作に一貫性のとれたGUI環境を得たことになる。このほかにも、デフォルトのWebブラウザがNetscapeからMozillaに変更されるなど、Kondara 2.0は、新しい機能の取り込みでヘビーユーザーを満足させつつ、充実したサポートで初心者にも優しいディストリビューションだといえるだろう。

画面2　GNOME 1.4
最新版GNOME 1.4とGUIシェルNautilusを組み合わせて使っている。

画面3　KDE 2.2.0
最新版KDE 2.2.0とKonquerorの組み合わせて実現されたデスクトップ環境

# SuSE Linux 7.2-International-2

ドイツを始め、ヨーロッパで高いシェアを誇るSuSE Linuxの最新バージョンSuSE Linux 7.2（SuSE 7.2）が6月15日にリリースされた。SuSE 7.2は基本システムに、カーネル2.4.4、glibc2.2.2、XFree86 4.0.3を採用している。また、標準のデスクトップ環境はKDE 2.1.1で、GNOME 1.4も利用可能だ。

## ついに国際化版登場

SuSE 7.2からは、**表1**のように2種類の国際化版がリリースされる。

現在のところ、Webページなどから入手できるのはドイツ語をサポートしたバージョンのみだが、**表1**に記載した国際化版も、製品として近々発売予定だという。

今回は、発売を間近に控えている国際版のひとつInternational-2を紹介する。このInternational-2は、ハンガリー語やイタリア語とともに、日本語サポートしているバージョンだ。

International-2は、先月紹介したSuSE Linux 7.1 Japanese editionから成果物を引き継いでいるので、内容面ではSuSE 7.1Jとほぼ同等である。

インストーラやSuSE Linuxの特徴であるYaST2（**画面1**）は日本語化されており、X上での日本語の入力や表示にも対応している（**画面2**）。

## なぜSuSEを使うのか？

日本語環境を提供するLinuxディストリビューションは、すでに数多くリリースされている。それなのに、なぜあえてSuSEを使うのだろうか？

国内外を問わず、Linuxディストリビューションは、それぞれ独自の設定ツールを開発している。しかし、WindowsからPCに入ったユーザーにとって、どの設定ツールもWindowsの設定ツールよりも高いレベルにあるとは言いがたい。

だが、SuSEの独自設定ツールYaST2は、そのようなユーザーでも違和感なく使える高いレベルのツールだ。システムの各設定は、YaST2のモジュールとして実装されているので、ユーザーがなにかを設定したいと思った場合は、まずYaST2を起動すればよい。

たとえば、Xを再設定する場合、設定に必要なRPMパッケージがインストールされていないとわかると、YaST2はそのRPMパッケージが収録されたCDをセットするようユーザーを促がす。ユーザーは、YaST2の出す指示に従って操作すれば、目的の設定を達成できるというわけだ。

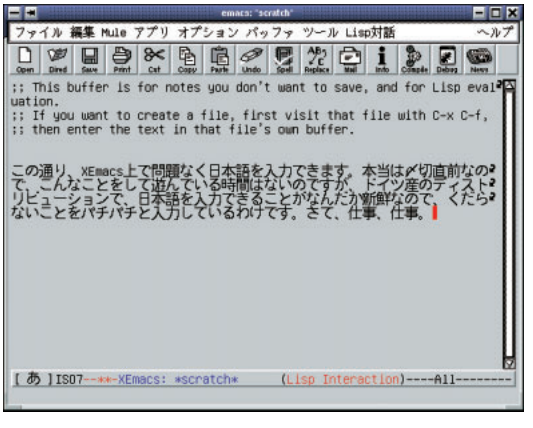
「～は存在しません」というメッセージを出すよりも、設定が目的のユーザーにとっては、「必要な～をセットしてください」というメッセージのほうがはるかに有効である。

そのようなポリシーで開発されたSuSEで日本語を使えるようになれば、初心者ユーザーにとって本当に有効なLinuxディストリビューションが誕生するといえるのではないだろうか。

| パッケージ名 | 標準の言語 | サポートされる言語 |
|---|---|---|
| German | ドイツ語 | 英語、ドイツ語 |
| International-1 | 英語 | 英語、フランス語、スペイン語、オランダ語など |
| International-2 | 英語 | 英語、日本語、ハンガリー語、イタリア語など |

表1　SuSE Linux 7.2に用意される3種類のパッケージ
SuSE Linuxは今回のバージョン7.2から、国際化を果たした。その一環として、日本語もサポートされている。


画面1　SuSE独自の設定ツールYaST2
Linuxに関する設定を、ほとんどこのツールで行える。




画面2　XEmacs上での日本語入力
日本語入力も問題なくできる。

# Distribution ▶▶▶

## Yellow Dog Linux 2.0発表

5/29にテラ・ソフト・ソリューションズが、Power Macハードウェア上で動作する「Yellow Dog Linux 2.0」（以下YDL 2.0）を発表した。

YDL 2.0には、新たにグラフィカルなインストーラが付属し、簡単なインストールが可能になった。カーネル2.2.19および2.4.4、XFree86 4.0.2、KDE 2.1、Mozilla、Open Officeが基本システムとして採用し、Javaがサポートされている。テラ・ソフトは、Linux上から古いMac OSを動かせるようにするアプリケーション「Mac-On-Linux」のセットアップを簡略化している。この新しいソフトウェアはTitanium PowerBook G4やワイヤレスネットワーキング用のAirPortといったアップルの最近の製品もサポートしている。YDL 2.0は、同社のWebサイトにて現在発売中。

Yellow Dog Linux 2.0
(http://www.yellowdoglinux.com/)
テラ・ソフト・ソリューションズ
(http://www.terrasoftsolutions.com/)

## ハイパーコア Debian GNU/Linux スナップショット CD 販売開始

有限会社ハイパーコアは6月5日、同社のインターネット通信販売サイト「ハイパーコアダイレクト」にて「Debian GNU/Linux Testing-Lite スナップショット (Woody)i386 4CD」（以下、Woody i386)の販売を開始した（4枚組で、2100円）。Woody i386は、Debian Projectが開発・テスト中の次期リリースWoodyから、人気のあるパッケージを選択、バイナリとソース合わせてCD-ROM4枚に収録したオリジナル商品だ。Debian GNU/Linuxの最新テスト版を試用するのに最適だ。

ハイパーコア（http://www.hypercore.co.jp/）

## Itanium対応ディストリビューション続々と発表

インテルが5月29日にItaniumプロセッサ（64ビットCPU）の正式リリースを発表したことを受けて、Itanium対応ディストリビューションが続々と発表された。

5月29日にRed Hatが「Red Hat Linux 7.1 For Itanium Processor」を発表、翌5月30日に米ターボリナックスが、「Turbolinux Operating System 7」を発表した。米ターボリナックスは、1年以上前にすでにItaniumプラットフォーム対応の、最初のディストリビューションを発表している。これは、Caldera、CERN、HP、IBM、Intel、Red Hat、SGI、SuSE、Turbo、VAなどで構成されるグループである、Trillianプロジェクト（現IA-64プロジェクト。IA64プロセッサが出荷される前に、IA64版Linuxを完成させることを目標に設立されたプロジェクト。現在は、オープンソース開発体制に移行している）によって開発されたカーネルとツールをベースとしたものであった。今回のリリースはそれに次ぐものとなる。

他にも、6月7日にCalderaが「OpenLinux Server 64」のベータ版を発表した。正式版の出荷は2001年秋になる予定だ。6月20日にはSuSEが「SuSE Linux 7.2 for IA64」を発売している。

Red Hat（http://www.redhat.com/）
Turbolinux（http://www.turbolinux.com/）
Caldera（http://www.caldera.com/）
SuSE（http://www.suse.com/）
IA-64 Linuxプロジェクト（http://www.linuxia64.org/）

## HOLON Linux3.0発売決定！

株式会社ホロンは7月13日に「HOLON Linux 3.0」（以下ホロン3.0）を発売する。ホロン3.0は、安定性重視のためカーネル2.2.19ベースとし、カーネル2.4.5を最新ハードウェアのために併用しているほか、glibc2.2.3、XFree86 4.0.3(3.3.6)、KDE2.1、GNOME1.4などを基本システムとして採用している。また、Intel/PPCのCPUに対応し、それぞれ並版と豪華版の2種類で計4つのパッケージが用意される。価格はホロン2.0と同じ（PC／AT互換機用豪華版11800円、並版3980円、PPC用豪華版9800円、並版4800円）。

ホロン3.0は、国内ディストリビューションとしては初めて、DOSパーテーションをリサイズする機能が追加された。また、CD-R作成ソフトウェア（X-CD-Roast)や音楽CDをMP3にして保存、また保存したMP3を聞くためのソフトウェア（ripperX）など、多数のアプリケーションも追加されている。次期バージョンも現バージョンに引き続き、安心できるサポート体制が用意されている。90日間無制限無償電話サポートと、無償サポート期間終了後の有償での延長サポートを受けることができる。

株式会社ホロン（http://www.holonsoft.co.jp/）

# Products


34 規模に応じてハードウェア構成を選べるプロキシ専用サーバ
**SOOXi 2001アプライアンスProxy/Cacheサーバ**

36 デュアル構成に対応したCPUとチップセット
**Athlon MP＆AMD-760MP**


## 規模に応じてハードウェア構成を選べるプロキシ専用サーバ

### SOOXi 2001アプライアンス Proxy/Cacheサーバ

**インターネットの普及によってWebブラウザは、なくては仕事にならないほど重要になってきた。会社や学校などで複数のPCからWebアクセスを行う場合には、インターネット回線の帯域を節約し、高速なアクセスが実現するためにプロキシサーバを導入したい。**

| | |
|---|---|
| 製品名 | SOOXi 2001アプライアンスProxy/Cacheサーバ |
| 価格 | 12万9800円〜 |
| 問い合わせ先 | ノーザンライツコンピュータ株式会社<br>TEL 044-850-2238<br>http://www.nlcomputer.com/ |

ノーザンライツコンピュータから、SOOXi 2001アプライアンスサーバシリーズの追加モデルとして、「SOOXi 2001アプライアンスProxy/Cacheサーバ」（以下Proxy/Cacheサーバ）が7月9日より発売される。

Proxy/Cacheサーバは、LAN内からインターネットへWebアクセスを行う場合に、クライアントPCの代わりにWebサーバへアクセスし、受け取ったデータをクライアントへ返すといった代理サーバ（Proxy）の機能を持っている。単に代行するだけではなく、その情報をいったんハードディスクに貯めておき（Cache）、LAN内の複数のPCから同一ページへのアクセスがあった場合には、ハードディスク内のデータをPCに送ることで、インターネットへのアクセスを減らすことができる。

すなわち、回線の有効利用が図れるため、結果的に回線を増強することなく高速アクセスが実現できるのだ。

また、リバースプロキシとしてWebサーバ側に接続することで、Webサーバへの負荷を減らすためにも利用可能である。

ハードウェアは、同社のNL Server シリーズより選ぶことができる。1U／2U／4U／5Uといったラックマウント型やタワー型などから設置場所や性能に応じて選択する。ブックサイズのコンパクトサーバNL Server 500ならば最小構成で12万9800円となっている。

今回試用したマシンは、基本ハードウェアに1Uラックマウントサイズの NL Server 1400で、CPUにPentiumIII1GHzのデュアルプロセッサ、メモリ2Gバイトを搭載しており、かなりハイスペックな仕様だ。ハードディスクは、前面よりホットスワ

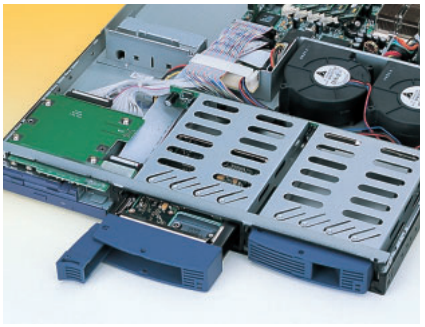


写真1　ドライブベイ
1Uラックマウントサイズの前面に、スリムタイプのフロッピー／CD-ROM、2基のハードディスクが装着される。

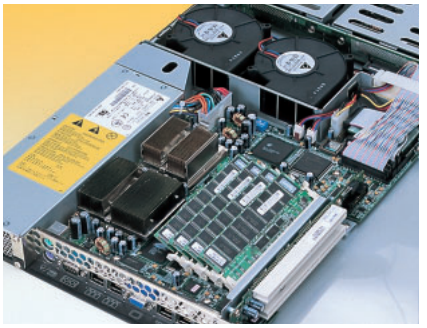
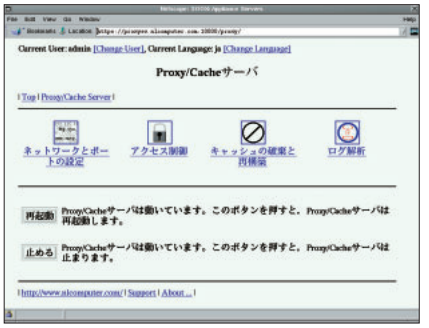
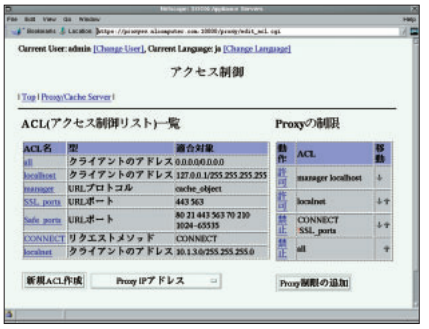


写真2　NL Server 1400のマザーボード
デュアルPentiumIII1GHz。2Gバイト（512Mバイト×4枚）メモリ。オンボードにSCSIコントローラを実装している。




画面1　Proxy/Cacheサーバのメニュー
プロキシ機能の動作のON/OFFのほか、ネットワークポートやアクセス制御の設定とログ解析が行える。




画面2　アクセス制御のメニュー
左側のACL一覧で設定する対象を定義し、右側のProxyの制限でそれぞれの許可／禁止の順序を設定する。

ップ可能なベイが2基分用意されていて、Ultra160 SCSI対応10000rpmと高性能なAtlas 10K II 36.7Gバイトが1基搭載されていた。

## Webブラウザで簡単設定

Proxy/Cacheサーバのインストールは、ユーザー（管理者）が行うようになっている。まず、あらかじめWindowsマシンなどで、フロッピーにeth0というファイル名で、192.168.1.10といったIPアドレスを書き込んでおく。

次に、Proxy/Cacheサーバの電源を入れ、付属の専用のインストールCDを装着する。しばらくして「ピ・ピ・ピ」という音が鳴ったところで、先ほどのフロッピーを挿入すると、ハードディスクが適切なパーティションで分けられ、Linuxと設定管理ツール「SOOXi Esperanto」がインストールされる。

自動的に再起動されるので、別のマシンのWebブラウザから先ほどのIPアドレスへアクセスする。このとき、SSLを使用してポート10000番に接続するため、https://＜IPアドレス＞:10000/のように入力することが必要だ。以後、すべての設定はWebブラウザから行うことができる。

Esperantoのトップメニューから“Proxy/Cacheサーバ”というメニューを選ぶと**画面1**のように表示される。画面の左下にある［再起動］と［止める］というボタンで、プロキシの動作を制御する。セキュリティに関わるアクセス制御を行うのが、**画面2**である。ここで許可／禁止するIPアドレスやプロトコル／ポート番号を設定することができる。

なお、Proxy/CacheサーバのOSは、Linuxカーネル2.4.5をベースにノーザンライツが独自に開発したものである。プロキシ機能は、Webプロキシ／キャッシュソフトのSquidによって実現されている。また、ログ解析ツールとして、CalamarisとWebalizerの2種類が搭載されている。

SOOXi 2001アプライアンスサーバシリーズには、このほかに、Webサーバ、Mailサーバ、DNS/DHCPサーバ、NAS（ネットワークストレージ）サーバが用意されている。設定ツールのメニュー内容は機能に応じて異なるが、それぞれのサーバの機能を限定しているため、設定しなければならないメニュー項目自体が少なく、導入から稼働まで非常に短期間に行うことができる。また、利用規模に応じてハードウェアを選べるもうれしい点だ。

| | |
|---|---|
| CPU | PentiumIII866MHz～1GHz（デュアル／シングル） |
| チップセット | ServerWorks ServerSetIIILE |
| RAM | 128M～4GバイトECC SDRAM（PC133） |
| ハードディスク | 9～72Gバイト（Ultra160 SCSI、最大2台、ホットスワップ可能） |
| CD-ROM/DVD-ROM | 24倍速CD-ROM（スリムタイプ） |
| フロッピードライブ | 3.5インチ1.44Mバイト |
| グラフィックス | オンボードATI RAGE XL（4Mバイト、AGP） |
| ネットワーク | オンボード10/100BASE-T×3ポート（Intel 82559） |
| SCSIコントローラ | オンボードAdaptec AIC-7892B（Ultra160） |
| インターフェイス | RS-232C×1、PS/2×2ポート |
| PCIスロット | フルサイズ2スロット（64ビット、66MHz） |
| キーボード／マウス | オプション |
| サイズ（mm）、重量 | 429（W）×618（D）×41.9（H）、最大10kg |
| 電源 | 200W |

表1　SOOXi 2001アプライアンスProxy/Cacheサーバ（NL Server 1400）の主な仕様

# Athlon MP＆AMD-760MP

**Athlon用のデュアル対応チップセットAMD-760MPと、同チップセットに正式対応したCPUであるAthlon MPがリリースされた。AMDがサーバ／ワークステーション市場へ切り込むための強力な尖兵だ（写真は同CPU、マザーボード搭載のリファレンスマシン）。**

| | |
|---|---|
| 製品名 | Athlon MP、AMD-760MP |
| 価格 | Athlon MP 3万4450円（1.2GHz）、2万7950円（1GHz）<br>（いずれも1000個ロット時） |
| 問い合わせ先 | 日本AMD株式会社<br>http://www.amd.com/japan/ |

AMDは、同社製のx86 CPUとしては初めて正式にマルチプロセッサに対応したCPUであるAthlon MP（**写真1**）と、2 CPUまでの構成で利用できるチップセットAMD-760MP（**写真2**）を発表した。

Athlon MPのコアは、現在「Athlon」として販売されているCPUの次世代に相当するもので、「Palomino」（馬の種名）というコードネームを持つ。Athlon MPは、現行のコアと同様の0.18μmプロセスで製造されているが、（1）低消費電力化（同クロック比で20%低減）、（2）データプリフェッチ機能、（3）3DNow!命令群の拡張（3DNow!プロフェッショナル）など、大幅な改良が加えられている。

## 大幅に改良されたCPUコア

データプリフェッチは、プログラム実行時に読み出されるデータを予想し、あらかじめキャッシュに読み込んでおく機能だ。CPUの性能（クロック）向上に比較して、メモリ高速化のペースは遅い。そこでデータプリフェッチによって、CPUがメインメモリからデータが読み出されるのを待つケースが減らせれば、性能向上が期待できる。

AMDは、K6-2以降のCPUで3DNow!と呼ばれる命令群を追加している。3DNow!は、SIMD（Single Instruction Multiple Data）型の命令を中心としている。同じ目的のものとして、インテルのSSE（Streaming SIMD Extension）がある。どちらも利用するためには、ソフトウェア側の対応が必要だ。

Athlon MPで採用された3DNow!プロフェッショナルは、SSEの命令を取り込んだものであり、あわせてSSEで利用するレジスタも付け加えられている。

3DNow!もSSEも、ソフトウェアの対応が必要という点は共通であるため、既存の3DNow!だけでなくSSEにも対応しているAthlon MPは、より多くのソフトウェアでSIMD型命令が利用できると考えられる。

## デュアルCPU対応のチップセット

AMD-760MPチップセットは、ノースブリッジAMD-762（**写真2**）とサウスブリッジAMD-766の2チップ構成である。CPUとノースブリッジの接続は、Pentium Ⅲのデュアル構成で見られるような共有バス型ではなく、Smart MPテクノロジーと呼ばれる1対1の接続方法が採用されている。そのためデュアルCPU構成時でも、各CPUがシングル構成と同等の帯域（2.1Gバイト／秒）を利用できるのが特徴だ。

メモリは、ECC付きPC2100 DDR SDRAMを利用して、4Gバイトまで対応可能だ（**写真3**）。

AMD-766サウスブリッジは、Ultra ATA/100、4×AGPといった一般的なデバイスサポートに加え、33MHz／64ビットPCIもサポートする（66MHz PCIは次期モデルで対応予定）。

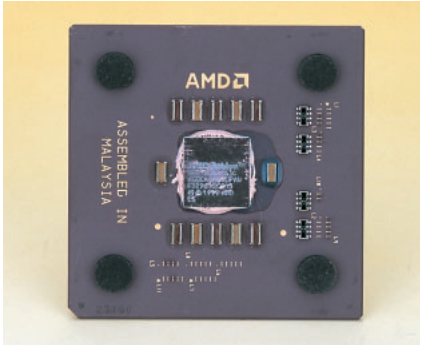



写真1　Athlon MP
1.2GHzモデル。同時に1GHzモデルもリリースされている。


写真2　AMD-760MP
ノースブリッジのAMD-762。外観はAMD社のK6シリーズによく似ている。

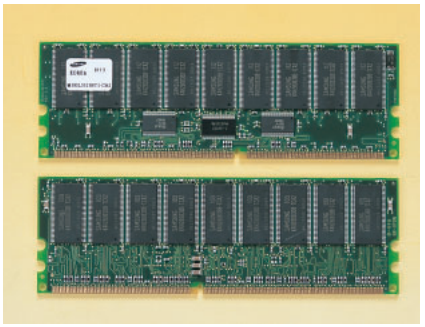


写真3　使用されているDIMMモジュール
ECC付きのRegistered DDR SDRAMを利用する。

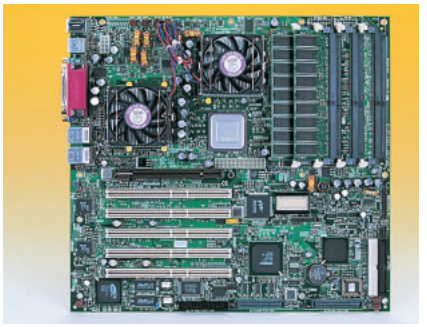


写真4　Tyan Thunder K7
初めてAMD-760MPを搬したマザーボード。サーバ／ワークステーション用途に適した構成だ。

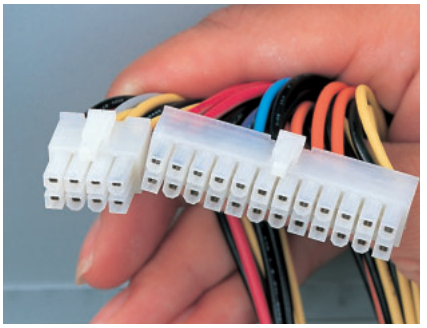


写真5　特殊な電源コネクタ
24ピン／8ピンの独自形状の電源コネクタが必要だ。

## 最初のマザーボードはTyanから

Athlon MP、AMD-760MPの発表と同時に、同チップセットを搭載したデュアルAthlon対応のマザーボード、Thunder K7（**写真4**）がTyan Computerから発売された。Ultra 160m/SCSI×2、100BASE-TX×2など多くのデバイスが搭載されていることや、1Uのような薄型ケースでの使用を想定している（4つのDIMMスロットが傾けて配置されている）ことからわかるように、サーバ／ワークステーション用に設計された製品である。

今回、このThunder K7を搭載したAthlon MPデュアル構成のリファレンスマシンを試用できたので、その内容を紹介しよう。

Athlon MPが現行のAthlonよりも低消費電力とはいえ、その絶対値は決して低いものではない。ましてや、デュアル構成ではそれが倍になるわけで、消費電力は通常のPCに比べかなり高い。そのため、Thunder K7では通常の20ピンATXコネクタとは異なる24ピンのコネクタが用いられている（**写真5**）。また、Pentium 4のデビュー以来、マザーボード上にCPU専用の4ピンコネクタが設けられるようになった。しかしThunder K7では8ピンの独自形状のコネクタが採用されている。

このようにThunder K7を利用する際には、専用の電源が必要になる。実際、秋葉原でもマザーボードと電源がセットで売られているようだ。リファレンスマシンには、460Wの電源が搭載されていた。

## Linuxも問題なく動作

リファレンスマシンは、前述のThunder K7にAthlon MP 1.2GHzを2個搭載し、Registered型のECC付きDDR SDRAMを512Mバイト搭載していた。Red Hat Linux 7.1をインストールしたところ、チップセット名は「unknown」と表示されるものの、CPUが2個あることも正しく認識され、問題なく利用することができた。

デュアルCPUの効果を見るために、カーネルのコンパイル速度を比較してみた。Red Hat Linux 7.1のカーネル2.4.2を用いている。/usr/src/linuxディレクトリに移動後、

```
# make menuconfig
```

として、何も設定を変えずに保存・終了して「config」ファイルを作成し、

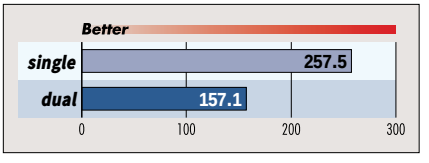
```
# make dep
# make clean
# time make bzImage
```

として所要時間を測定した。デュアルCPUの場合は、makeコマンド実行時に「-j3」オプションを付加した。

**グラフ1**に結果を示す。一般にカーネルコンパイルでは、CPUの性能だけでなく、ハードディスクの性能なども影響するため、デュアルCPUにしてもシングルCPUの2倍までは性能が上がらない。デュアルAthlonのリファレンスマシンでは、両方のCPUを利用することで、60%以上高速化された。Athlon MPデュアルマシンは、現状で最も速いx86ベースのワークステーションの1つと言えるだろう。


画面1　GNOMEの画面
デュアルCPU＆大容量メモリで使うGNOMEは、非常に快適だ。

グラフ1　カーネルコンパイル（秒）



［続報］PlayStation 2でLinuxが動く！

# プレステ2 Linux詳報

## PlayStation 2 Linux その実像に迫る！

**7月20日、ついにPS2 LinuxKit Beta Release 1が出荷された。早く使ってみたいと待ちこがれていた方も多いだろう。編集部でも早速入手したので先月号に続き、詳報をお届けしよう。**

いまさら説明する必要もないと思われるが、PS2 LinuxKit Beta Release 1（以下、PS2 LinuxKit）は、ソニー・コンピュータエンタテイメント（以下、SCEI）が発売したPlayStation 2（以下、PS2）用のLinuxキットだ。PS Linux Users Groupが精力的に行っていた、「PS2で動作するLinuxをSCEIに公開してもらうための署名運動」によって実現したキットである。コミュニティの活動がメジャーメーカーを動かした特異な例といえるだろう。


写真1　SCPH-18000のPCカードスロット
ここにハードディスク＋ネットワークインターフェイスカードを接続する。現状では、PCカードを持たないPS2ではPS2 LinuxKitを使用することはできない。

さて、PS2 LinuxKitの構成だが、次のようになっている。

- **PS2 Linux Beta Release1（DVD-ROM）**
- **キーボード（USB）**
- **マウス（USB）**
- **ハードディスク**
- **PCカード（イーサネット＋ハードディスクインターフェイス）**
- **VGA接続ケーブル**

このキットにはPS2本体は付属していないので、別途PS2本体を用意する必要がある。ただし、どのPS2でも動くわけではなく、PCカードスロットを装備している型に限られ（型番SCPH-10000、SCPH-15000、SCPH-18000の3種類。箱と本体背面に記載）、PCカードスロットを内蔵していないPS2（型番SCPH-30000、SCPH-35000）では使えない。よって、現在では入手困難な若干古い型のPS2でなければならないことになる。

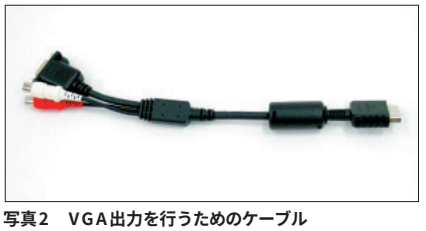


写真2　VGA出力を行うためのケーブル
このケーブルを経由してSync on Green対応のディスプレイに出力する。編集部で確認したところ、EIZO FlexScan L461では出力することができなかった。

また、PS2用のメモリカードと画面出力のためのディスプレイも別途必要となる。PS2を使ったことがない読者のために解説すると、メモリカードは本体のメインメモリとして使用するものではなく、外部記憶メモリとして使うためのものだ。通常は、ゲームのセーブデータなどを保存しておくためのものである。ディスプレイは「Sync on Green」に対応している必要がある。

Sync on Greenとは、RGBの緑（G）の信号線に垂直同期信号と、水平同期信号を混合した複合映像同期信号を指すもので、ワークステーションなどで利用されている信号である。ほとんどのPCのVGA出力では各信号が分離されているので、ディスプレイがSync on Greenに対応している必要はない。最近のディスプレイではコストダウンのためにSync on Greenに対応していないものもかなりあるようだ。このため、むしろ古いディスプレイのほうがSync

on Greenに対応している可能性が高い。また、メーカーの仕様表記ではSync on Greenに対応しているものであっても、PS2 LinuxKitの出力が表示できないなどの問題もあるようで、このあたりはしばらく混乱が続きそうだ。

キーボードとマウス（**写真3**）はUSBのカスケード接続でPS2本体に接続する。

ハードディスク（**写真4**）は付属しているPCカード（**写真5**）経由でPS2本体に接続することになる。ハードディスクドライブはSONYのロゴが記載されているものの、SeagateのST340823Aだと思われる（**写真6**）。おそらくOEM製品だろう。このドライブは、40Gバイト、5400回転、平均シークタイム8.9ms、UltraATA/100対応のものだが、PCカード経由で接続されるため、ハードディスクドライブそのものの性能は活かせない。

さらに、このPCカードには10BASE-T/100BASE-TXのコネクタもあり、ネットワークカードとしても機能する。

## Linuxのインストールとブート

Linuxの起動は付属のDVD-ROM（**写真7**）から行う。まず、DVD-ROMからインストーラを起動し、メモリカードにカーネルイメージを書き込み、/パーティション以下をハードディスクにインストールすることになる。フルインストールには約1時間ほど要する。やはり、PCカード接続のハードディスクの速度がボトルネックとなるようだ。

なお、SCPH-18000ではPS2の初回起動時に言語設定や時刻設定などを行う画面が表示されるため、あらかじめ家庭用テレビに接続して設定しておく必要がある。

インストール後でも、Linuxの起動にはDVD-ROMが必要となり、残念ながらハードディスクやメモリカードから直接Linuxを起動することはできない。これは、PS2のプロテクトの問題で、SCEIがライセンスしたCD-ROMやDVD-ROMでないとPS2のBIOSで弾かれてしまうためだ。

## PS2のLinux

PS2 LinuxKitには、一般的なLinuxのプログラムやサーバプログラムが含まれている。たとえば、SambaやApacheなども含まれており、これらのサーバプログラムを使ってサーバを構築することも可能だ。telnetやSSH、FTPなども用意されており、PCサーバを使ってWindows上にXを表示させることもできる。

なお、先月号の速報ではベースとし

写真5　ハードディスク＋ネットワークインターフェイスカード
このカードを経由して、ハードディスクユニットとネットワークケーブルを接続する。残念ながら、PCカードのため、ハードディスクの転送速度は遅い。

写真3　PS2 LinuxKitに付属するキーボードとマウス
キーボード、マウスともにUSB接続される。かなりしっかり作られたキーボードで使いやすい。今後、ゲーム用として販売される可能性が高い。

写真4　ハードディスクユニット
電源スイッチはついておらず、カーネルが読み込まれると自動的に電源が入るようになっている。ファンを内蔵しているが、こちらもカーネルが読み込まれるまで回らない。

写真6　ハードディスクドライブ
SONYのロゴが記載されているものの、dmesgの出力を見る限りSeagateのST340823Aだと思われる。なお、分解するとメーカー保証対象外になるので注意されたい。

写真7　PS2 LinuxKitに付属するDVD-ROM
このDVD-ROMは特殊なフォーマットで、PS2規格DVD-ROMと呼ばれる。このDVD-ROMがなければ、ハードディスクにインストールしたLinuxを起動することはできない。

ているディストリビューションはRed Hat Linuxとお伝えしたが、sdrコマンドが用意されている点やグラフィカル画面を見ると（**画面**）、実際にはKondara MNU/Linuxベースだと思われる。RPMデータベースを参照してみたところ、「release Kondara」という表記が散在していた。

PS2用のLinuxは、CPUがEE（Emotion Engine。R5900という型番が付いている）という点を除けば、普通のx86用Linuxと同じように使用することができる。ただし、CPUが違うため当然バイナリパッケージなどは動作しないので、ソースコードからコンパイルする必要がある。編集部でいくつかのプログラムをコンパイルしてみたが、おおむね正常に動作した。ただし、x86のアセンブラを使用したものなどはコンパイルすることはできない。このため、HDBenchや午後のこーだなどはコンパイルすることができない。

カーネルのバージョンは2.2.1、コンパイラはgcc 2.95.2、glibcのバージョンは2.2.2と若干古いものを採用している。X Window Systemは XFree86 3.3.6に GS（ Graphic Synthesizer。PS2のビデオチップ）ドライバを追加したものとなっている。ただし、GS用には最適化されていない。

X Window Systemの設定は、XFree86の通常の設定ファイルである/etc/X11/XF86Configではなく、/etc/X11/XGSConfigで行う。ドライバ名はgsxとなっている。Xの解像度や色数を変更する場合は、このファイルを変更することになる。


**画面　グラフィカルログイン画面のダイアログ**
**左のペンギンの絵が違っているが、そのほかの部分はKondara MNU/Linuxのログイン画面とそっくりだ。そのほかKondara独自のコマンドなども含まれている。**

さて、動作速度について気になるところがだが、コンソールでの動作であれば軽快に動く。Xも、Window Makerであればサクサク動くので、十分実用になるだろう。ただし、GNOMEでGimpなどを動かした場合は若干もたつく感がある。これは、PS2のメインメモリが32Mバイトしかないので、ディスクスワップが発生しているためだ。

## グラフィックス

PS2といえば、当然3Dグラフィックスに関する環境が気になるところだが、こちらはPS2用の低解像度ドライバと、Mesaのコンソール用ドライバが用意されている。いずれもサンプルプログラムが用意されていて、実際に動かすことができる（Mesaのサンプルプログラムには一部正常に動作しないものがあった）。なお、MesaのドライバはGS用に最適化されていないので、速度はいまひとつというところだ。

また、X Window System用のMesaドライバは用意されていないので、X上で動くOpenGLを使ったゲームなどを実行することはできないが、Mesaを使わないX用のゲームなどは十分なスピードで動作する。とはいえ、GSの性能を活かすことができるX用のMesaドライバも用意してほしいところだ。

PS2はテキスト画面を持っていないため、コンソール画面の出力はフレームバッファを経由して行われる。このため、仮想コンソール画面などは利用できない。たとえば、X Window Systemが起動している状態で、Ctrl+Alt+F1などを押すと画面表示が乱れてしまう。

フレームバッファを利用したコンソールなので、ハイレゾリューションで表示することもできる。これにはsetcrtmodeコマンドを使用する。たとえば、

```
# setcrtmode 1280x1024
```

とすることで、広い画面を利用することができる。

ただし、konコマンドが用意されていないので、コンソールで日本語を表示することはできないようだ。

## 内蔵DVD-ROMの注意点

PS2はDVD-ROMドライブを内蔵しているが、PS2 LinuxKitでは一般的なDVD-ROMやCD-ROMをマウントすることができない。マウントできるのは、PS2規格のCD-ROMとDVD-ROMだけだ。これは仕様だとSCEIは公表しているが、CD-ROMやDVD-ROMメディアが読めないのは非常に不便だ。これがハードウェア上の仕様なのか、ソフトウェア上の仕様なのかは現状では不明だが、ソフトウェア上の仕様であればぜひとも改善してほしい点だ。しかし、ハードウェア上の仕様だという可能性も高い。これは、PS2ソフトのコピー防止のために、一般的なDVD-ROMやCD-ROMをわざと読めないようなドライブを採用している可能性があるからだ。

なお、一般的なDVD-ROMやCD-ROMをマウントしようとするとエラーとなり、以降、「No Medium found」となってどのメディアも認識しなくなってしまう。

このように、現状ではPS2のDVD-ROMドライブでDVD-ROMやCD-

ROMを読み込むことはできないので、ネットワーク経由でDVD-ROMやCD-ROMメディアを読み込む必要がある。あるいは、USB接続のDVD-ROMドライブやCD-ROMドライブを接続することで、DVD-ROMやCD-ROMを読み込むことができるかもしれない。実際、編集部では確認できていないが、モジュールを組み込むことでCD-ROMが読み込めたという情報も入手している。これに関しては機会があれば検証してみたいと思う。

## テレビ出力

PS2 LinuxKitはSync on Green対応のディスプレイが必要だと前述したが、PS2そのものはテレビ画面（NTSC）に出力することを前提にして作られたものである。そこで、家庭用テレビに出力できないか試したところ、VGA出力（640×480）という低解像度ではあるものの、家庭用テレビに出力することができた。なお、Linuxの起動から、X Window Systemまですべて家庭用テレビだけで表示可能だ（**写真8・9**）。

PS2 LinuxKitインストール後に、メモリカードをマウントする。

```
# mount /mnt/mc00
# cd /mnt/mc00
```

次に、p2lboot.optファイルを修正し、「display=vga」行を「display=ntsc」に変更する。以上で次回起動時からNTSC出力が可能となる。なお、X Window Systemを起動する場合は、

```
$ startx -- -screen 0 NTSC
```

とすることで、NTSCに出力することができる。

逆に、ゲームファンであればPS2用ゲームがVGA出力できるかどうかが気になるところだろう。結論からいえばこれはできない。これはゲーム側の対応が必要になると思われる。ただし、今後のソフトウェアの対応次第によっては、VGA画面出力が可能になるかもしれない。

## 今後の展望

PS2 LinuxKitは、ベータ版とはいえ非常に安定して動作しているし、サーバとして稼働させることもできる。今後、製品版の出荷が期待されるところだが、同時にPCカードを持たないPS2へのリリースも期待される。PCカードを持たないPS2では拡張ベイが用意されており、そこにハードディスクユニットが内蔵される。もしかすると、PCカードを経由しないぶん、こちらのほうがハードディスクの性能が活かせるかもしれない。

また、クローズドだったPS2のハードウェアに関するドキュメントも豊富に付属しているため、個人でPS2用のゲームを開発することも不可能ではない。これからのLinuxコミュニティの成果も期待させる。

なお、PS2 LinuxKitでゲームを開発する場合、PS2用低解像度グラフィックスライブラリを使うか、コンソール用Mesaドライバを使うことになるわけだが、そうなると、汎用性のあるMesaドライバを使いたいと考えるのが順当だろう。このとき、メモリを消費するX Window Systemを使うのを避け、コンソール用に開発する可能性も高い。となると、GS用に最適化されたコンソール用Mesaドライバの登場が待たれるところだ。


写真8　家庭用テレビ（NTSC）に出力したブート画面
多少ちらつくものの、一応読みとることができる。往年のMSXの画面を見るようで懐かしい。インストーラが対応すれば、ディスプレイがなくても使うことができる。


写真9　家庭用テレビに出力したX
640×480という低解像度のため、かなり厳しい感はあるが、使うことはできる。写真8同様、家庭用テレビを撮影したため見にくい点はご容赦いただきたい。


File Preferences
Config Control Status
Networking
Client tasks
Host name and IP network dev
Name server specification (DNS
Routing and gateways
Host name search path
Network Information System (N
IPX interface setup
Server tasks
Misc
Users accounts
File systems
Miscellaneous services
Peripherals
boot mode
Host name and IP devices
You are allowed to control the parameters which are specific to this host and related to its main connection to the local IP network.
Host name
Adaptor 1 2 3 4 5 6
Host name + domain
localhost.localdomain
Accept
Cancel
Help
Resolver configuration
Routes to other networks
Name service access
Edit IP aliases configurations
Boot mode configuration
Boot this menu enabled
Default operation mode
Graphic & Network
Text mode & Network
Delay to activate
Prompt timeout
20
15
JAPAN

# GUIツールを使いこなせ! Linuxシステム設定ガイド

文：香取精二＋編集部
*Text : Seiji Katori+Linux Magazine*

「テキストの設定ファイルをエディタで修正する」これがLinuxにおけるシステム設定の王道である。しかし、ファイルがどこにあるのかわからなかったり、それぞれの書式がバラバラだったりと、慣れないうちは戸惑う面が多いのも確か。これらの問題を解消してくれるのがGUIインターフェイスを備えた設定ツールだ。本特集では、GUI設定ツールの2大勢力「Linuxconf」と「Control Panel」、それにデスクトップ設定ツール「GNOMEコントロールセンター」を紹介する。これらのツールを活用すれば、システム設定作業がぐっと楽になるゾ！

# LinuxconfでLinuxの設定を極める

文：香取精二
*Text : Seiji Katori*

Linuxマシンの設定をしようと思っても、まず、設定が書かれているファイルのありかがわからない。なんとか見つけても、こんどは設定ファイルの書き方がわからない。そんな悩みもLinuxconfを使えば解決だ。

Linuxマシンの管理は、ターミナル上でエディタを使って設定ファイルを書き換え、コマンドラインで管理コマンドを発行というのが一般的な方法だ。これは、UNIXマシンなどに慣れた管理者にとっては慣れ親しんだインターフェイスで、使いやすいものだ。

しかし、WindowsやMacintoshなどに慣れた人にとっては、「/etc/sysconfig/networkというファイルを編集して、プログラムを再起動して……」などという方法は逆に面倒でわかりにくいだろう。そんな人のためのGUI管理ツールがLinuxconfだ。このLinuxconfを使えば、基本的なLinuxマシンの管理に必要なことはほとんど可能になる。

そして、Linuxconfを使うことのメリットは、直接設定ファイルを編集することなくGUIでさまざまな設定ができるということだけではない。Linuxconfでの設定を有効にするためのデーモンの再起動なども、設定内容に応じて適切に行ってくれる。

今回はRed Hat Linux 7.1付属のLinuxconf（1.24R2）を元に解説を行う。旧バージョンとはちょっと見た目が違っているが、基本的な設定項目などは同じなので適宜読み替えてほしい。

ただし、Red Hat Linux 7.1をワークステーションやサーバとしてインストールした場合は、Linuxconfはインストールされない。その場合は「rpm -ivh」コマンドで以下の2つのパッケージをインストールしよう。

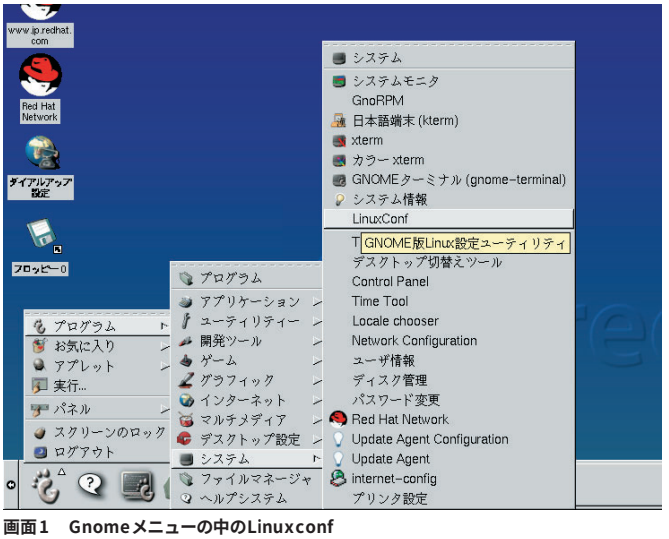
```
linuxconf1.24r2-10.i386.rpm
gnome-linuxconf-0.64-1.i386.rpm
```

ほかのディストリビューションの場合もファイル名のバージョン番号の部分などは違うかもしれないが、同様にインストールしてほしい。なお、X Window Systemを使わない場合はgnome-linuxconf-0.64-1.i386.rpmは不要だ。

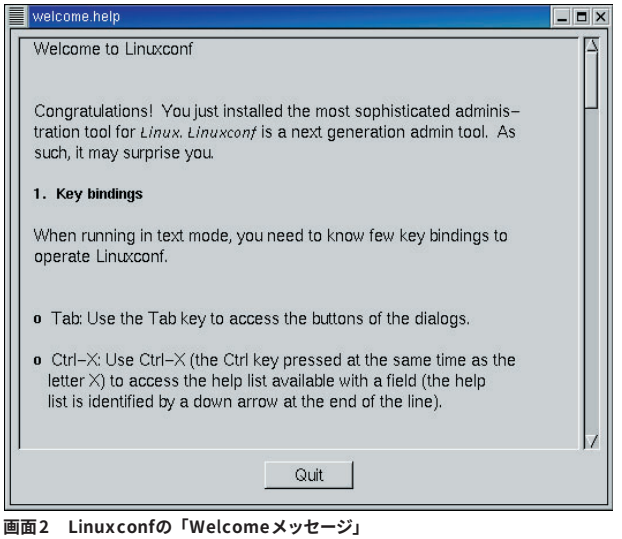
これらをインストールすると、いままではなかったGNOMEのメニューの中に「Linuxconf」が現れるはずだ（**画面1**）。

## Linuxconfの基本操作

それではまず、Linuxconfを起動してみよう。ただし、Linuxconfはマシンの設定を行うツールなので、rootユーザーでないと実行できない。rootでログインしているなら、GNOMEメニューからLinuxconfを選べばよいが、一般ユーザーでログインしているなら




**画面1　Gnomeメニューの中のLinuxconf**
**Linuxconfはインストールされて初めてGNOMEのメニューに追加される。**




**画面2　Linuxconfの「Welcomeメッセージ」**
**表示されるのは最初に立ち上げた1回だけだ。**

ターミナル（ktermやRxvt）上で、rootになって起動すればよいだろう。

```
$ su -
# linuxconf
```

初めて起動しときは、**画面2**のような「Welcomeメッセージ」が表示されるので［Quit］ボタンをクリックしてメインの画面に移ろう（**画面3**）。

おおざっぱに言えば、設定ファイルを編集する代わりに設定を行うのが［Config］タブ、ターミナルでコマンドを打つ代わりに管理コマンドを発行するのが［Control］タブ、そして動作記録を見るのが［Status］タブだ。それぞれ、タブを選んで操作することになる。もっとも、［Config］と［Control］の分類はかなり適当なものだ。

細かい項目は、ツリーの中から選んでいく。項目名の左が［＋］になっているのは、さらに中に項目がある親項目を意味する。これはダブルクリックすると項目が展開され、［＋］が項目が開いているという意味の［－］になる。選んだ項目に関する操作パネルは、ツリーメニューの右側に表示される。

各操作パネルではパラメータを入力後、パネルの下部の［Accept］ボタンなどをクリックすることによって設定が完了する（**画面4**）。設定をキャンセルする場合は、［Cancel］もしくは［Dismiss］を選ぼう。

なお、設定を有効にするために設定ファイルの書き換えだけでなく、プログラムの再起動などが必要な場合は、［File］メニューから［Act/Changes］を選ばないと設定が有効にならない。［Act/Change］を選ぶと、このときLinuxconfが実行するコマンドが確認できる（**画面5**）。実行してよければ［Do it］を選ぼう。

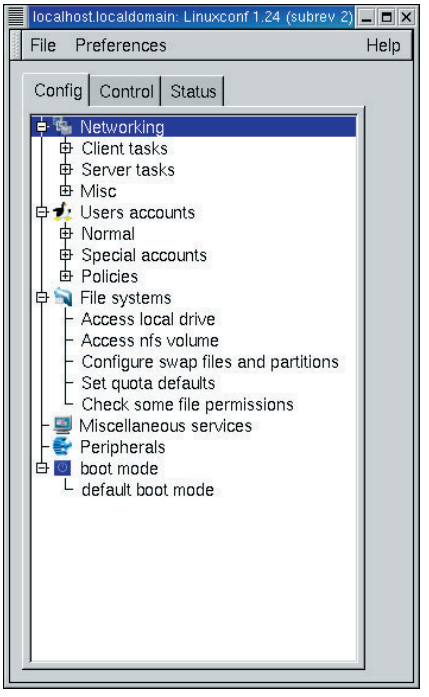
気をつけてほしいのは、ある項目を選んだあとに別の項目を選んだとき、右側の操作パネルは切り替わるのでなく、追加されていくということだ。設定項目だけ入力し、［Accept］ボタンをクリックせずにどんどん操作パネルを追加していくと、設定したつもりの項目を有効にせずに終了してしまうことがある。1つの項目を設定するごとに［Accept］をクリックして、確実に設定を完了していくようにしよう。

## ネットワークの設定

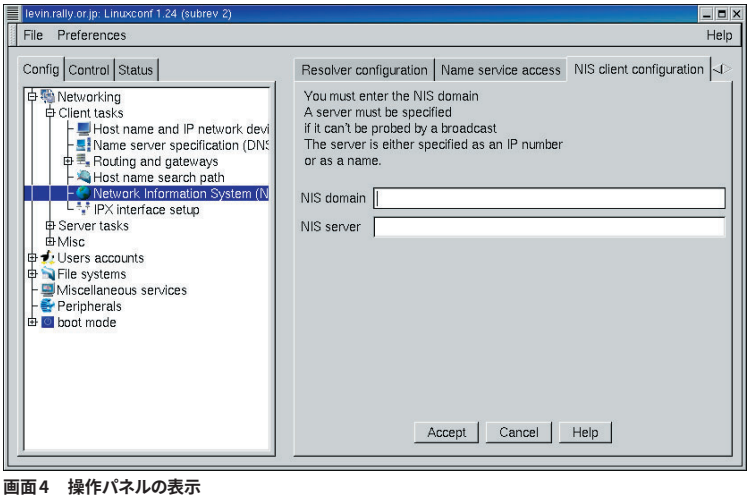
Linuxマシンのネットワーク設定は、ほとんどLinuxconfで事が足りてしまう。［Config］タブの［Networking］の中の各項目で設定する。まずは、そのマシン自体の基本的なネットワーク設定を行う［Client tasks］の中の主要な項目について見ていこう。

### ■ホスト名とIPアドレスの設定

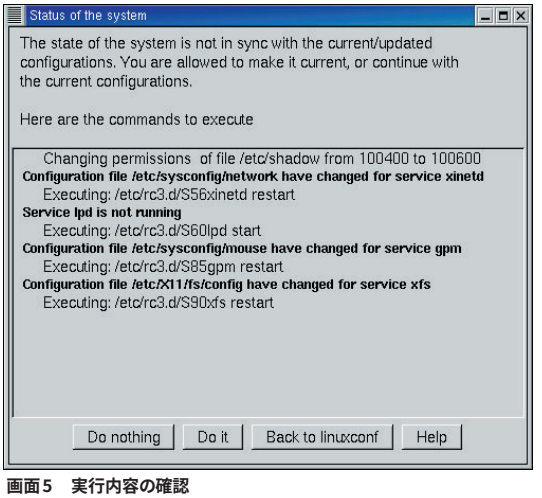
［Networking］－［Client tasks］－［Host name and IP network device］



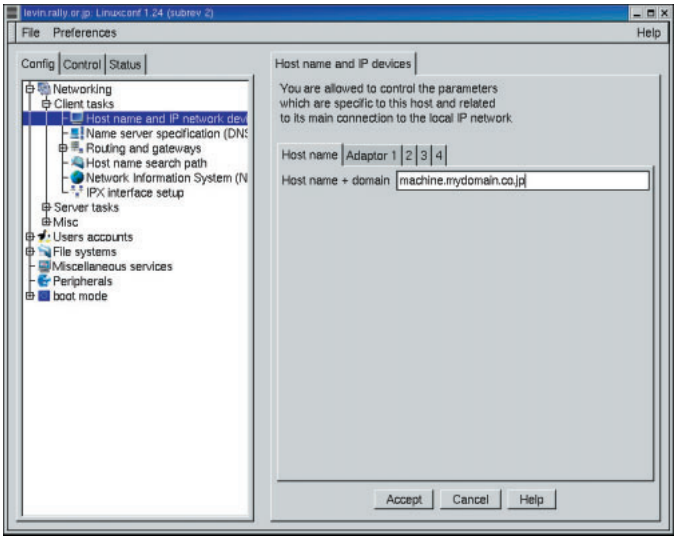

**画面3　Linuxconfのメインの画面**
**立ち上げたときはシンプルなツリー画面のみになっている。**



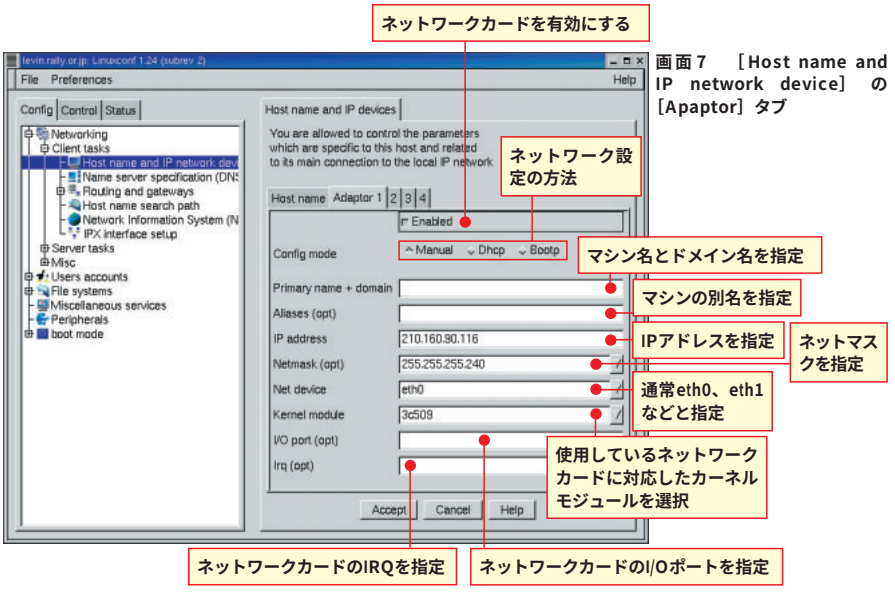

**画面4　操作パネルの表示**
**表示された操作パネルはタブで選んで1つ1つ設定していく。**




**画面5　実行内容の確認**
**単に設定ファイルの内容が書き換えられたものについては表示されない。**




画面6　[Host name and IP network device]の[Host name]タブ



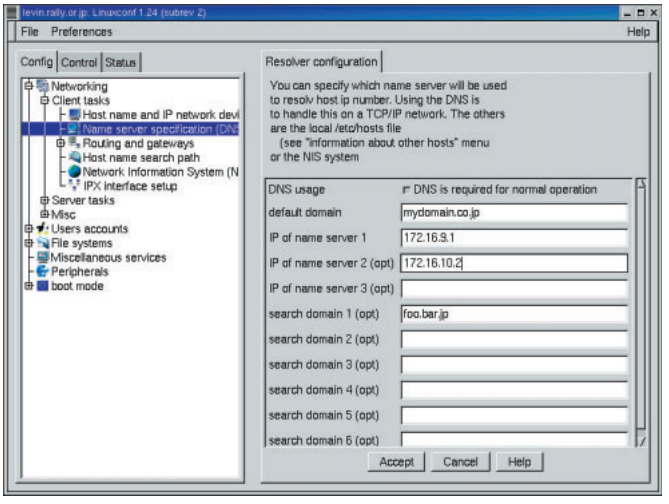

画面7　[Host name and IP network device]の[Apaptor]タブ

インストールのときにネットワークカードが検出されれば、この項目については設定を求められるので、すでに設定されている場合もある。ただし、中には自動検出されないネットワークカードもあるし、カードを追加する場合もある。そういった場合にLinuxconfを使えば、いろいろなファイルを編集したり、カーネルモジュールの設定ファイルを編集したりしなくてもよい。特にモジュールの設定は、リスト表示された中からモジュールを選べばよいので楽だろう。

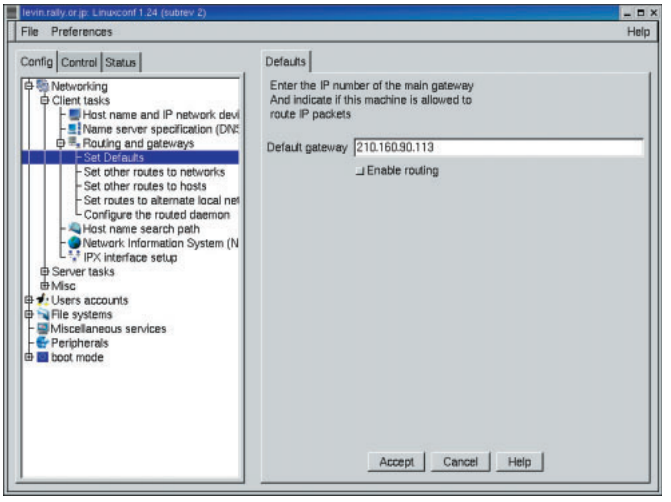
まず、[Hostname]タブでホスト名の設定をしよう。ここにはドメイン名付きのホスト名を設定する（**画面6**）。

次に、[Adaptor 1][Adaptor 2]……でネットワークカード1つ1つについて必要な、IPアドレス、ネットマスク、カーネルモジュールなどの設定を行う。この[Adaptor]タブは、実際に存在するネットワークカードの数に関わらず、最初から4つ存在する。[Adaptor 1]から設定していけばいいだろう（**画面7**）。

[Config mode]でDhcpやBootpを選んだ場合はネットワークの情報はそれぞれのサーバから受け取ることが




画面8　[Name server specification]
DNSサーバは停止している場合に備えて、複数設定しておくことが望ましい。




画面9　[Routing and gateways] − [Set Defaults]

できるので、ハードウェアの設定となる［Net device］と［Kernel module］を指定するだけでよい。もっとも、場合によっては［I/O port］や［Irq］なども指定する必要があることもあるだろう。

### ■DNSサーバの指定

［Networking］－［Client tasks］－［Name server specification］

数台のマシンを閉じたネットワークで使うのでない限り、複数台のマシンを運用する場合はDNSサーバの利用は必須といってもよい。この項目もすでにインストール時に設定している場合が多い。改めて設定する場合は、まず［DNS usage］をチェックし、［defaut domain］にマシンが所属するドメイン名を指定する。次に、DNSサーバのIPアドレスを優先したい順に［IP of name server］に指定しよう。マシン名だけを指定してネットワークにアクセスしたときに、補ってほしいドメイン名があるなら［search domain］に指定しておけばよい。**画面8**の例では、単にアクセス先のマシン名としてlevinを指定した場合、まず最初にlevin.foo.bar.jpというマシンを探し、次にlevin.mydomain.co.jpというマシンを探すことになる。

### ■ゲートウェイとルーティングの設定

［Networking］－［Client tasks］－［Routing and gateways］

通常は［Set Defaults］で、ゲートウェイのIPアドレスを設定するだけでかまわないだろう（**画面9**）。もし、ル

## Column

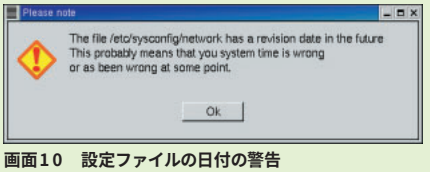
### Linuxconfのエラー

Linuxをインストールしてすぐにlinuxconfを使い、終了したり設定を有効にしたりしようとすると、**画面10**のような警告が出ることがある。このメッセージは、「今から設定しようとしているファイルの日付は未来のものだ」という意味のものだ。

この原因は時差の存在だ。

Linuxのインストーラは、コンピュータ内部にある時計をGMT（世界標準時）とみなしてファイルの更新日を設定していく。ところが、インストール時にタイムゾーンを［アジア/東京］とすると、インストール後はコンピュータ内部の時計はJST（日本標準時）とみなされる。ところが、JSTはGMTよりも9時間先の時間になっているため、相対的に過去へのタイムスリップ現象が発生してしまうのだ。

この問題への一番簡単な解決策は、消極的だがインストール後コンピュータを9時間寝かしておいてから、Linuxconfによる設定を行うことだろう。もしくは、内部時計がGMTとみなされるインストール時のみコンピュータの時計を9時間遅らせておくというのも手だ。




画面10　設定ファイルの日付の警告

この問題は、標準時が世界標準時よりも進んでいる地域でしか起らないので、アメリカやイギリスでは気づかれないのではないだろうか。

また、設定変更などを何もしていなくても、Linuxconf終了時の確認でメッセージが表示されることがある。たとえば、**画面5**の中にある「Service lpd is not running」などのメッセージだ。

lpdはプリンタ用のサーバプログラムで、通常、システムの起動時に起動される設定になっている。ところが、lpdは起動時に1つもプリンタが設定されていないと即終了してしまうのだ。

そこで、Linuxconfが設定を有効にするときに「lpdは起動される設定になっているはずなのに、起動していない。起動しますよ」というメッセージを表示することになるのだ。

lpd以外でも、自動起動するように設定されているプログラムがうまく起動されていないときは、このようなメッセージが毎回表示されることになる。すなわち、このようなメッセージが発生するときは、不要なプログラムが起動される設定になっているか、うまく起動されていないプログラムが存在することが多い。

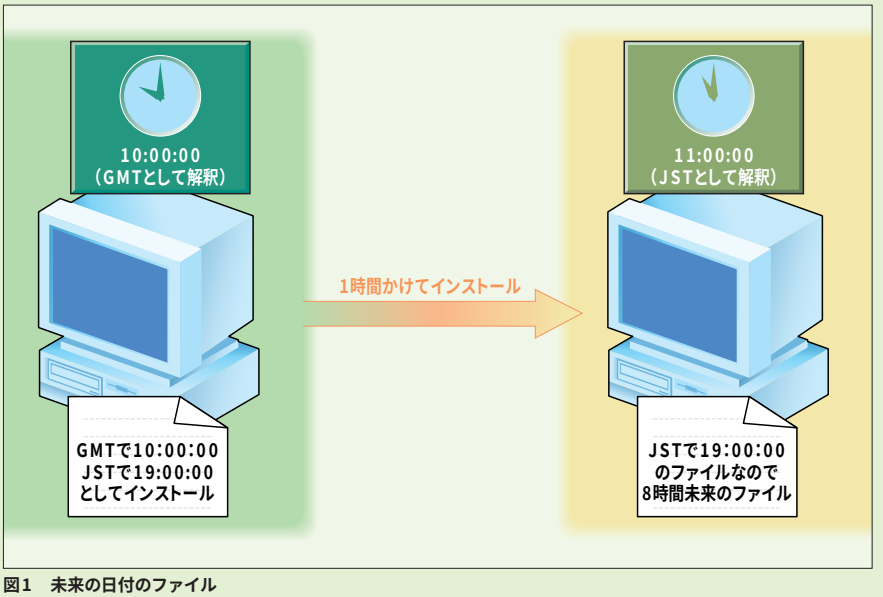



図1　未来の日付のファイル

ーティングを行うのなら、その他の項目で設定できる。

### ■ホスト名検索の優先順位

［Networking］－［Client tasks］－［Host name search path］

TCP/IPネットワーク上で、ホスト名とIPアドレスを変換する方法にはDNS、/etc/hostsファイル、NISなどがある。この項目は、これらのうちどれを優先して使うかを設定する。自分が優先したい順番になっているものをチェックしよう。とはいえ、hostsファイルを最初に参照し、次にDNSによる名前の解決を行うのが一般的なので、通常は［hosts, dns］を選べばいいだろう。初期値もそうなっているはずだ（**画面11**）。

### ■そのほかの［Client tasks］項目

これらのほかに、NISの設定をする［Network Information System］、IPXプロトコルを使うときに設定する［IPX interface setup］などの項目があるが、これらはそれぞれNISやIPXを意識して使うのでなければ特に設定する必要はない。

### ■ネットワークサーバの設定

［Networking］－［Server tasks］

［Exported file systems(NFS)］では、NFSを使って外部に公開するディレクトリの設定を、［IP aliases for virtual hosts］では各ネットワークデバイスに対するIPアドレスの別名を定義できる。

### ■ホスト名とIPアドレスの登録

［Networking］－［Misc］－［Information about other host］

［Misc］の中の［Information about other host］では、DNSを使わずに接続したいIPアドレスとホストのリストを作成する。［/etc/hosts］タブで［Add］をクリックすると**画面12**のようなパネルが開き、マシン名、別名、IPアドレス、コメントを指定できる。

DNSサーバが存在しない場合や、DNSに登録されていないマシンがある場合などはここで指定しておく。

### ■ネットワークの登録

［Networking］－［Misc］－［Information about other network］

ネットワーク名とネットワークアドレスの登録を行う。基本的には［host/network definition］と同じだ。

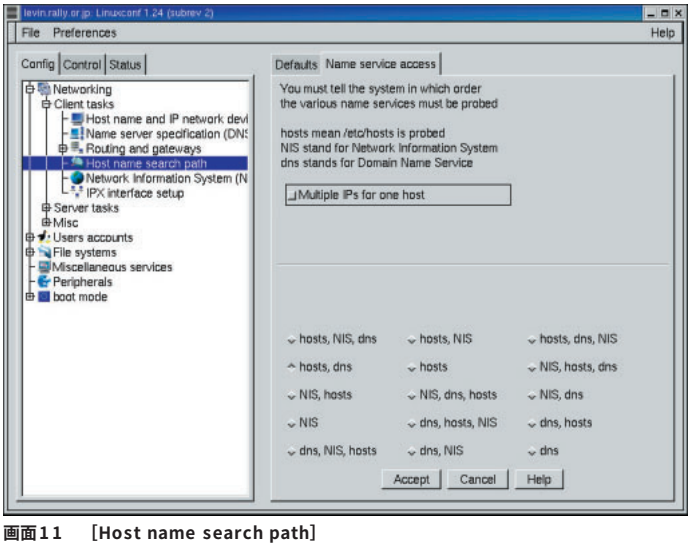
### ■Linuxconfのネットワーク使用

［Networking］－［Misc］－［Linuxconf network access］

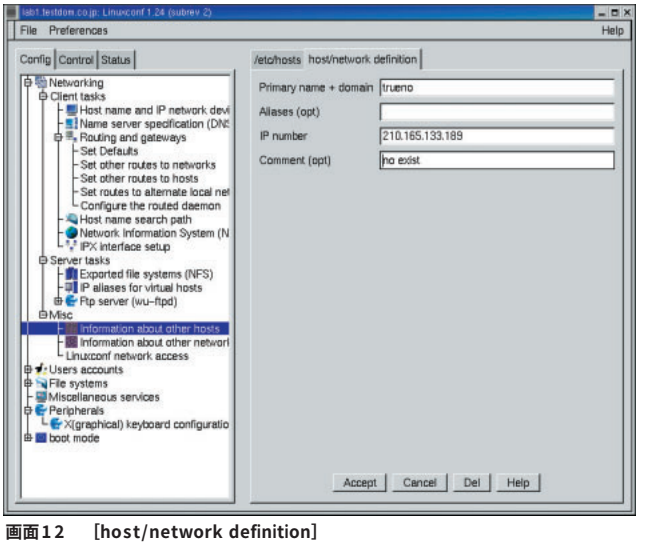
Linuxconfは、ネットワークを経由して起動し、ほかのマシンからリモートでさまざまな設定を行うことができる。このとき、ネットワーク経由で設定できるマシンを限定できるよう、リモートからLinuxconfを使用して設定をすることのできるIPアドレスを指定できる。もっとも、ネットワーク経由でLinuxconfを使用できるようにすることは、セキュリティ上の観点から好ましいものではないので、［Enable network access］はチェックしないほうがよいだろう。もちろん初期値も無効になっている。

## ユーザー管理

ユーザー管理に関する設定は［Config］タブの［Users account］の中の各項目で行う。




画面11　［Host name search path］




画面12　［host/network definition］

### ■ユーザーアカウント

［Users account］－［Normal］－［User accounts］

それぞれのユーザーの追加、削除、設定の変更を行うことができる。ユーザーの追加は［Add］をクリック、設定の変更は各ユーザーをクリックすればよい。各ユーザーの設定では、［User information］という、ユーザー情報設定パネルが表示される。なお、ユーザーの削除は一度各ユーザーをクリックして選択したあと、［Del］をクリックして行う。

設定できる項目は、基本的な内容の［Base info］、アカウントやパスワードの有効期限などを設定する［Params］、そしてLinuxconfにおける各ユーザーの権限を設定する［Privileges］に分かれている。

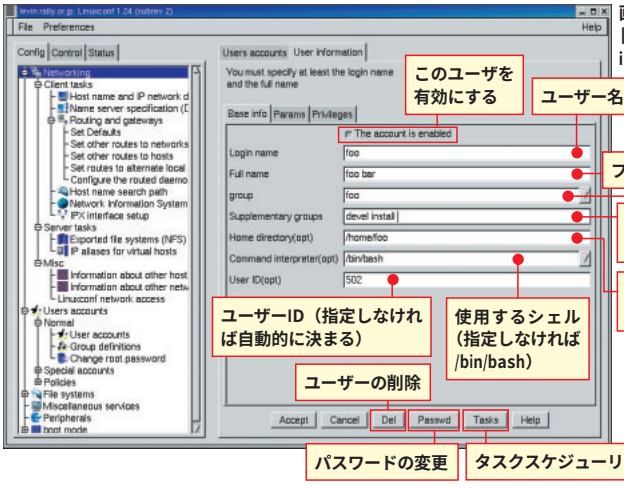
通常のユーザー管理であれば［Base info］で十分だろう（**画面13**）。コマンドラインからユーザーを追加する方法に慣れていたとしても、ユーザーを一覧の表形式で見ながら設定をすることができるのは非常に便利だ。

もし、パスワードを定期的に変更するなど、セキュリティにキチンと気を配りたかったり、ユーザーの入れ替わりが激しい場合は、［Params］が役に立つだろう（**画面14**）。それぞれのユーザーのパスワードや、アカウントの有効期限などを細かく設定することができる。

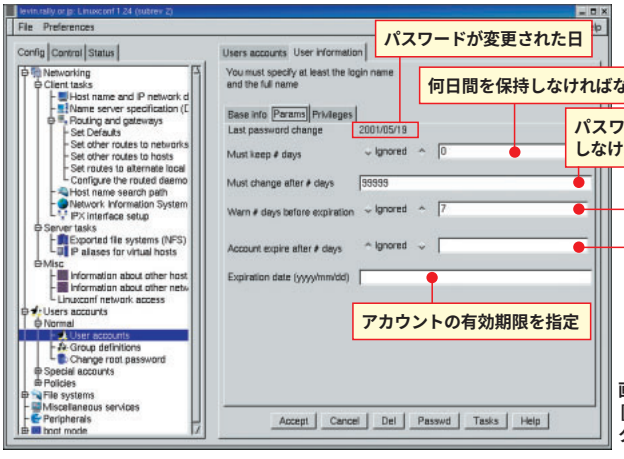
なお、Linuxconfは本来rootユーザーが操作するツールだが、［Privileges］で設定することによってrootユーザー以外のユーザーも許可された機能を選んで利用することができるようになる（**画面15**）。ここでは、［Denied］、［Granted］、［Granted/silent］の3種類を設定できる。

［Denied］にすれば権限は与えられない。

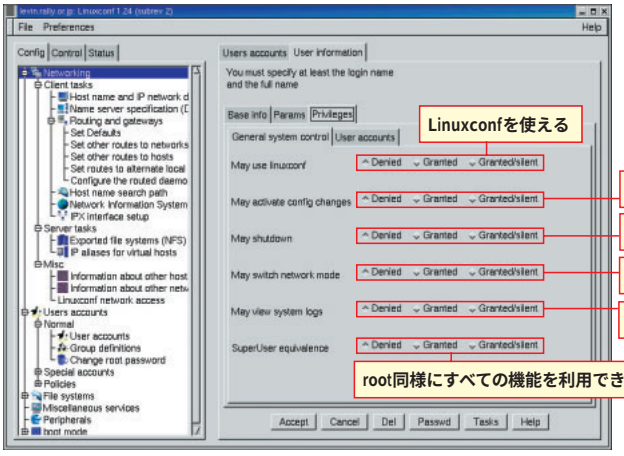
［Granted］にすれば、それぞれのコマンドの実行時にそのユーザー自身のパスワードが必要になる。変更を加えるような機能に割り当てておけば、うっかり変更してしまうこともないだろう。



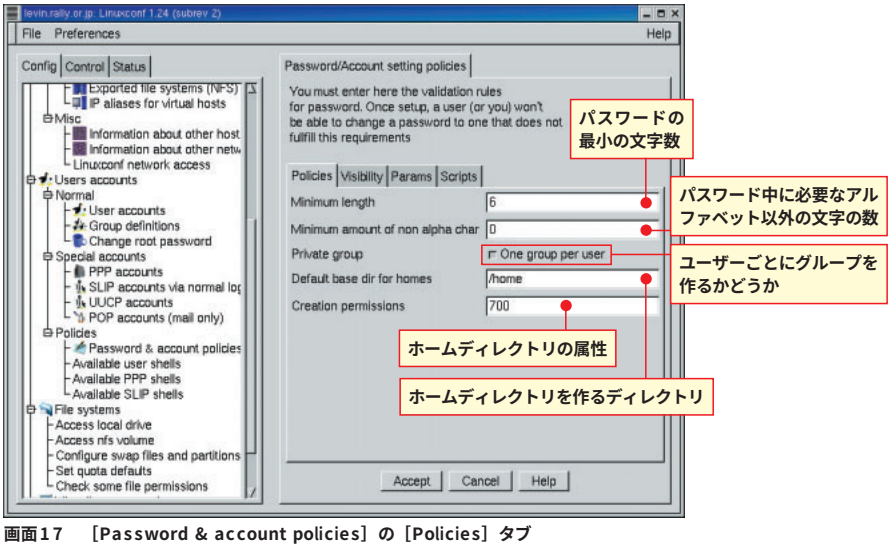

画面13　［User accounts］－［User information］の［Base info］タブ



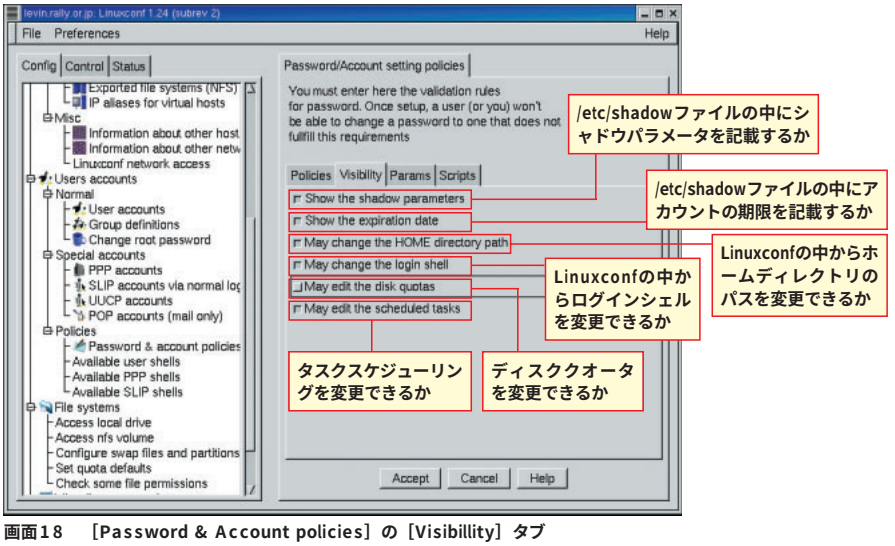

画面14　［User accounts］－［User information］の［Params］タブ



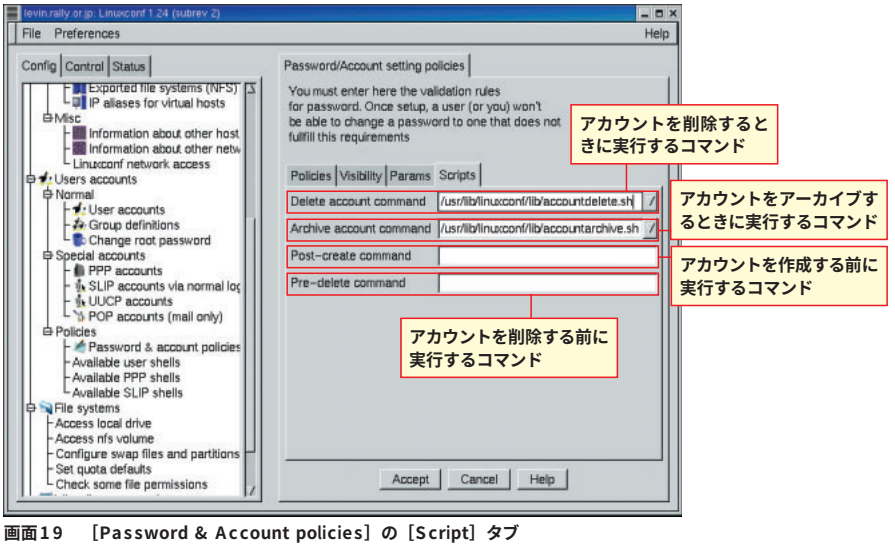

画面15　［User accounts］－［User information］－［Privileges］－［General system control］タブ




画面17　［Password & account policies］の［Policies］タブ




画面18　［Password & Account policies］の［Visibillity］タブ




画面19　［Password & Account policies］の［Script］タブ

［Granted/silent］ならば、コマンド実行時にパスワードは必要なくなる。ログを見るような機能などに割り当てるとよいだろう。

このほかに、ディスクの消費量を制限するディスククォータを有効にすると、［Disk quota］も表示され、各ユーザーごとに設定できるようになる。

### ■グループの管理

［Users account］－［Normal］－［Group definitions］

何人かのユーザーをひとまとめにして管理するのがグループだ。

［Base info］タブではグループ名、グループID、そして［Alternate members］にグループに所属するユーザーを設定することができる（**画面16**）。

［Group disk quota］タブではグループ全体のディスククォータを、［Members default disk quota］ではグループに所属するユーザーのディスククォータの初期値を指定できる。

### ■rootのパスワード変更

［Users account］－［Normal］－［Change root password］

コマンドラインからpasswdコマンドを使うというのもそれほど面倒なことではないが、ほとんどの管理コマンドを使うことができるLinuxconfだから、当然rootユーザーのパスワード変更もできる。選択すると、小さなパスワード入力ウィンドウが表示される。

### ■特殊なアカウントの設定

［Users account］－［Special account］

PPP、SLIP、UUCP、POPといった

特殊なグループに属していることが必要になるアカウントは、ここで設定できる。使い方は普通のユーザー管理とほぼ同じになっている。

### ■パスワードやアカウントの管理方針

［Users account］−［Policies］−［Password & account policies］

ここでは、パスワードやアカウントにおける基本的な設定を決定する。ほとんどの場合は、初期値のままでよく、変更しなくても動作に問題はない（**画面17**）。

ただ、インターネットに直接接続し、かつ多人数で使うようなマシンの場合などは、セキュリティを高めるためにパスワードの制限を厳しくしておくのもよいだろう（**画面18**）。

［Params］タブの設定内容については、［User accounts］の設定を参考にしてほしい。

［Script］タブ内では、アカウントを操作するときに実行するコマンドが指定できる（**画面19**）。アカウントを削除するときにホームディレクトリを丸ごと消去したり、アカウントを作成するときに、必要な初期設定ファイルをホームディレクトリにコピーしたりするといった、ユーザーアカウントを管理するときに毎回行う作業をシェルスクリプトにして登録しておくと便利だろう。

### ■ログイン可能なシェルの設定

［Users account］−［Policies］−［Available user shells］

ログインシェルとして使用可能なシェルを設定する。

よく使われるシェルは、最初からほとんど登録されているが、自分で好みのシェルをインストールして、ログインしたときから使いたいというときには、ここにフルパスで書いておく必要がある（**画面20**）。

### ■特殊なシェル

［Users account］−［Policies］−［Available PPP shells］など

PPPやSLIP用に、ネットワーク設定などを行うシェルスクリプトを設定している。変更する必要はないだろう。

## ディスク管理

ディスクの管理は［Config］タブの［File systems］の中の各項目で行う。

### ■ディスクドライブの設定

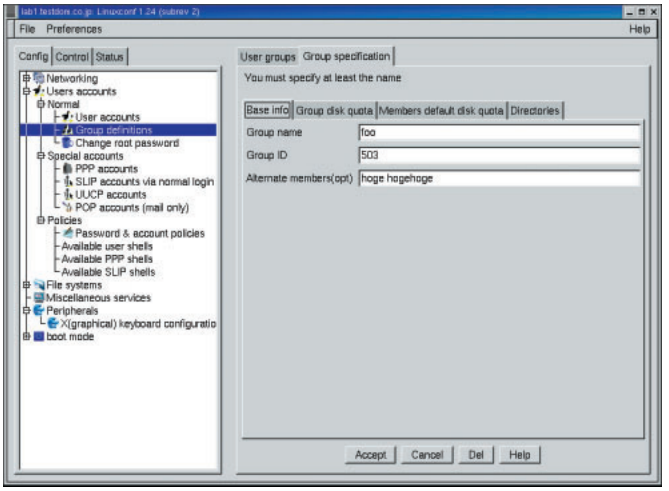
［File systems］−［Access local drive］

Linuxでは、ディスクはマウントすることによってはじめて使用可能になる。このディスクマウントの設定を行うのが、［Local Drive］だ。本来/etc/fstabというファイルを編集して行うものだが、Linuxconfを使うとディスクデバイス名や、オプションなどを選択しながら設定できる。英語ではあるが、オプションの内容もわかりやすく書かれているので、インストール後にディスクを追加したり、設定を変更したりするときなどに役立つだろう。

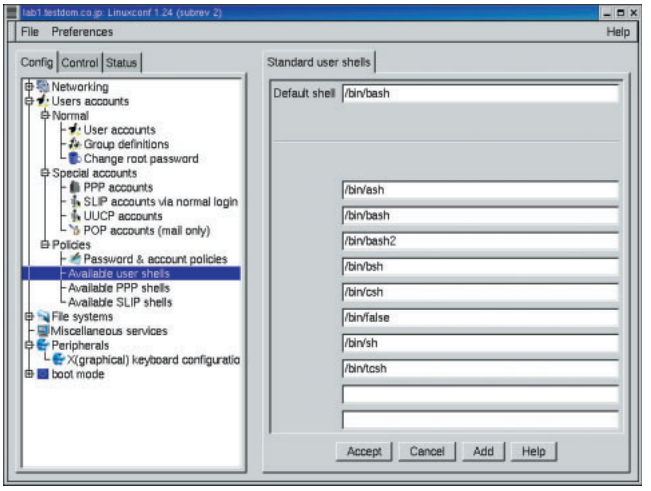
最初に現れるパネルでは、接続されているディスクの一覧が表示されるので、設定したいディスクをクリックしよう。追加したディスクを設定したい場合は、［Add］ボタンをクリックだ。

これらの豊富な項目の中で最低限設定しなくてはならないのが、［Base］タブの内容だ（**画面21**）。それ以外はオプション設定となる。

その場ですぐにマウント、アンマウントしたい場合は、パネルの下の［Mount］、［Unmount］ボタンをクリ




画面16　［Group accounts］−［Group specification］




画面20　［Available user shells］

ックしよう。また、設定したディスクをシステム起動時からマウントしておくかどうかは［Options］タブの［Not mount at boot time］で指定する。

### ■スワップファイルの設定

［File systems］－［Configure swap files and partitions］

この項目では、スワップファイルの追加、変更をすることができる。実際にはほとんど使うことはないだろう。

### ■クォータの設定

［File systems］－［Set quota default］

必要以上にディスクを圧迫しないように、各ユーザーがどれだけのディスク容量を使用できるかなどを設定するのがクォータだ。複数のユーザーが使うマシンなどでは便利な設定だろう。

ただし、初期設定ではそれぞれのディスクがクォータを使用できるようには設定されていない。そのため、まず最初に、［Config］タブの［File systems］－［Access local drive］で、それぞれのディスクがクォータを使用できるように設定しよう。

その後、［Set quota default］を選び、クォータを設定したいディスクをタブで選ぶ。次に、［User default］の［Disk space soft limit］にKバイト単位で数値を設定すればクォータが有効になり、ユーザーは設定されたサイズ以上にディスクを消費できなくなる（**画面22**）。

### ■ファイル属性のチェック

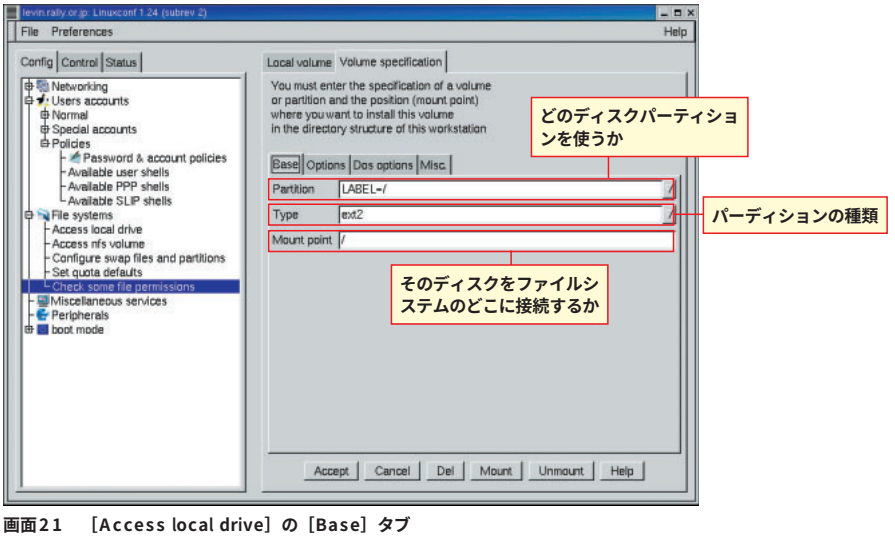
［Check some file permissions］

システムに関わるファイルによっては、セキュリティその他の理由により、特定の属性になっている必要がある。この項目を選択すると、それらが正しい属性がチェックしてくれる（**画面23**）。

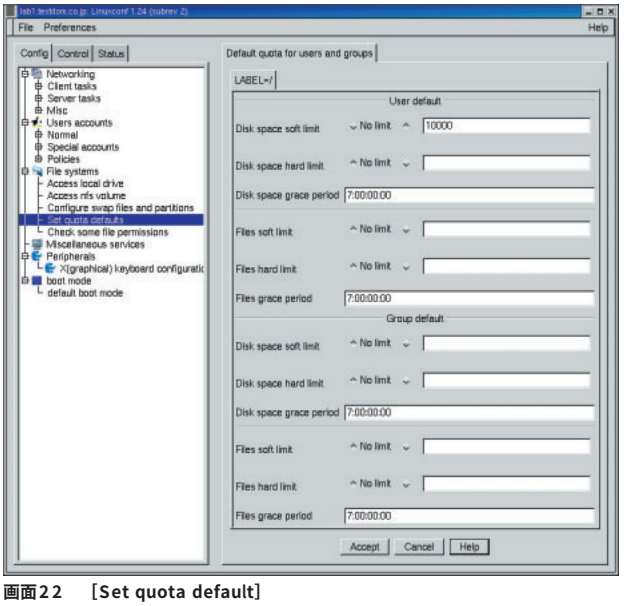
## ブートモードの選択

［boot mode］－［defalut boot mode］

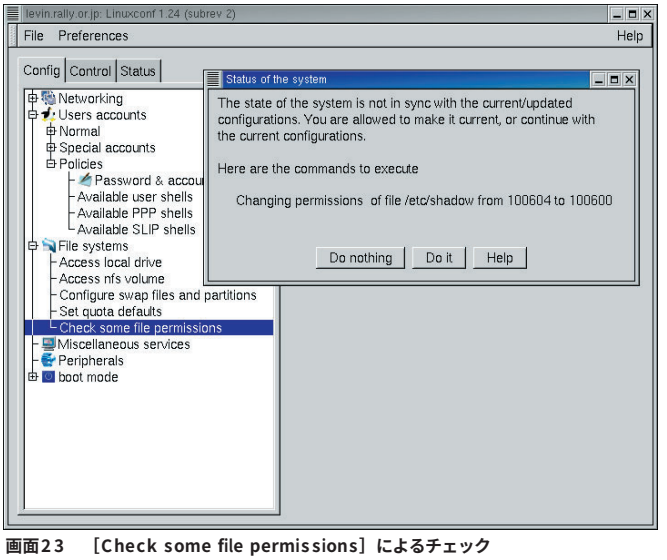
Linuxには、xdmやgdmを使ったグラフィックモードと、コンソールを使




**画面21　［Access local drive］の［Base］タブ**



**画面22　［Set quota default］**



**画面23　［Check some file permissions］によるチェック**
**/etc/shadowというファイルを誰にでも読むことができる状態にすると、警告と修正のダイアログが表示される。**

ったテキストモードの2種類の動作モードがある。これらのモードなどを設定するのがこの項目だ（**画面24**）。

ただし、Red Hat Linx 7.1では、この項目で設定してもモードの変更は有効にならなかった。すべてのディストリビューションやバージョンを調査することはできなかったが、正常に動作するものも多いので実際に試していただきたい。

Red Hat Linux 7.1でグラフィカルログインと、コンソールを使ったテキストモードを切り替えるには、Xconfiguratorを使用するといいだろう。

## 管理コマンドの実行

［Config］タブでは設定の変更を行ったのに対し、［Control］タブではおもにその場ですぐに実行される管理コマンドを発行する。

### ■現在の設定を有効にする

［Control panel］－［Active configuration］

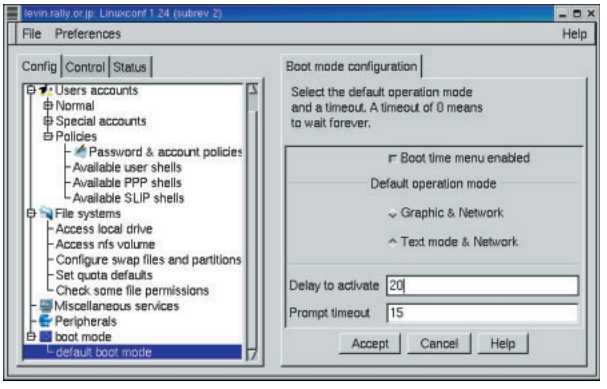
この項目はクリックするだけで、Linuxconfで設定できる項目についてすぐに現在の設定を有効にする。

### ■シャットダウンと再起動

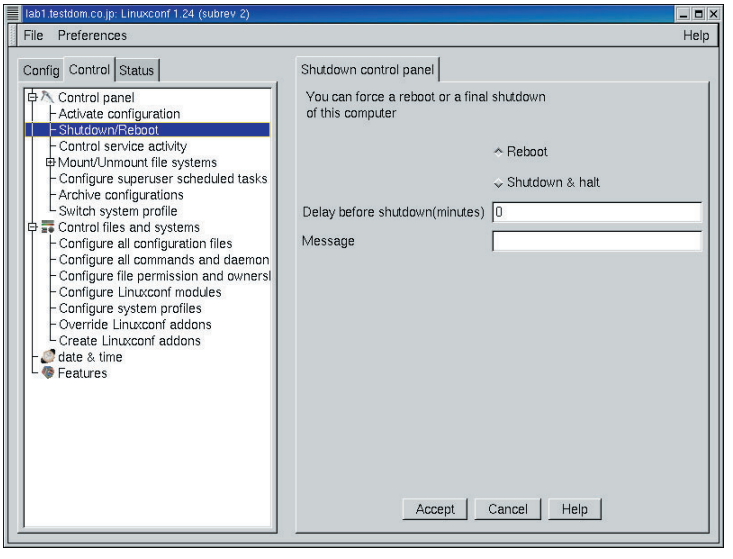
［Control panel］－［Shutdown/Reboot］

マシンをその場で、再起動（［Reboot］）、もしくは停止（［Shutdown & halt］）する（**画面25**）。

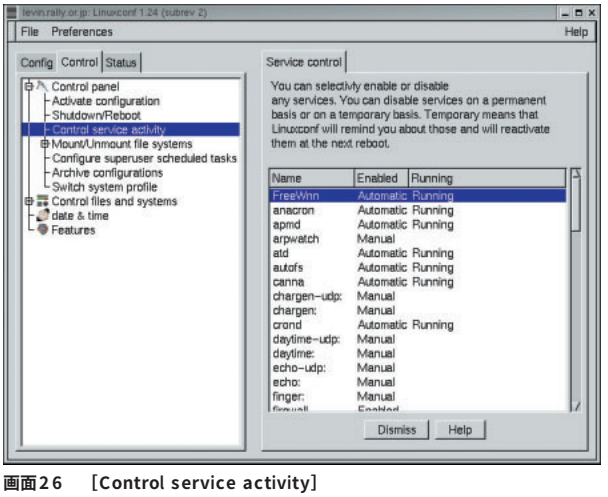
しばらく時間をおいてから停止したい場合には、［Delay befor shut down(minutes)］に時間を設定しよう。また、［Message］に文字列を設定すれば、このマシンにログインしている



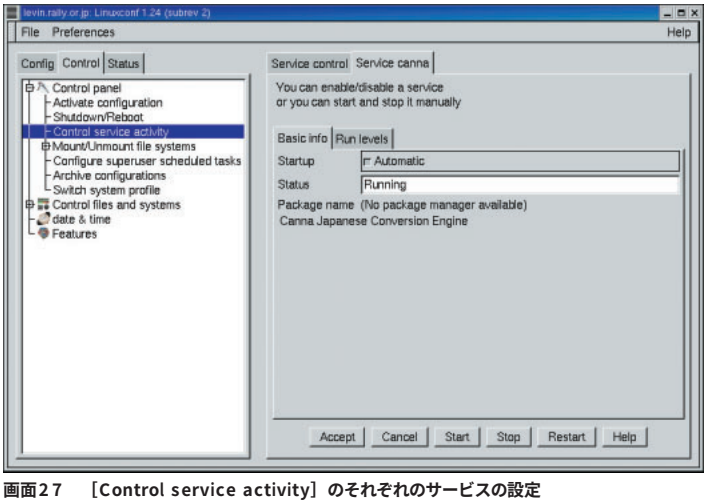

画面24　［Graphic or text］で起動時の設定を変更する




画面25　［Shutdown/Reboot］




画面26　［Control service activity］
［Enabled］で自動起動の設定を、［Running］で現在の状態を確認することができる。




画面27　［Control service activity］のそれぞれのサービスの設定

ユーザー全員に、停止予告のメッセージを流すことができる。

### ■さまざまなサービスの設定

［Control panel］－［Control service activity］

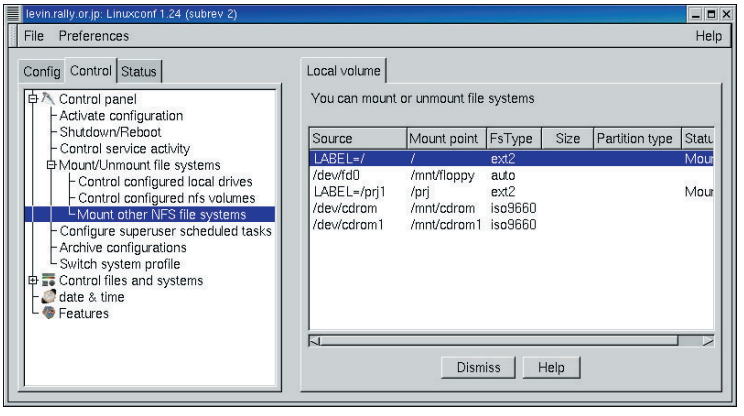
Linux上で動作するさまざまなサービスの、起動、停止、そして自動起動を設定することができる。

まず、リストの中から設定したいサービスについてクリックしよう（**画面26**）。すると、それぞれのサービスの設定パネルが表示される。

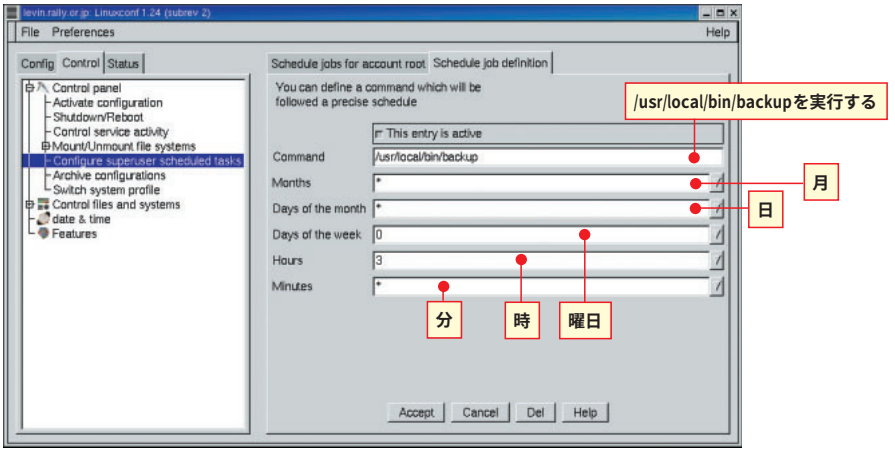
［Basic info］タブの［Startup］では、自動起動の設定を行うことができる（**画面27**）。自動起動に設定した場合は、どのランレベルで自動起動するかも［Run levels］タブで設定しておこう。テキストモードで使用している場合はランレベル3を設定、グラフィックモードで使用している場合はランレベル5を設定する。

もちろん、両方のモードを使うのであれば、両方とも設定する。

パネルの下のボタンをクリックすれば、その場でサービスを起動（［Start］）、停止（［Stop］）、再起動（［Restart］）することもできる。




**画面28 ［Mount/Unmount file systems］－［Control configured local drives］**



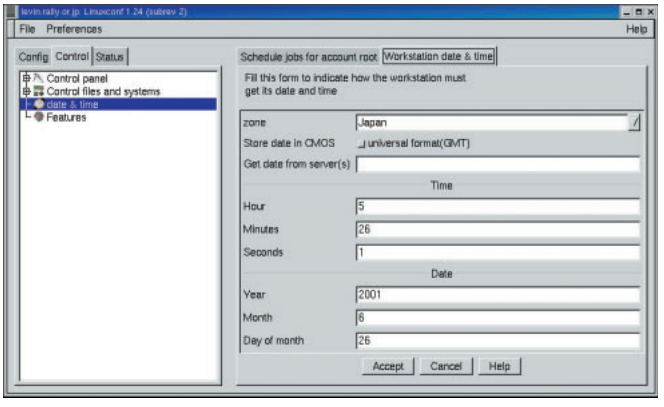

**画面29 ［Configure superuser scheduled tasks］**

### ■ディスクのマウント/アンマウント

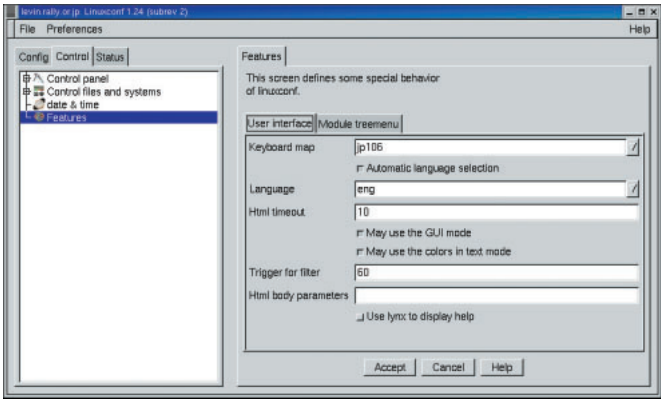
［Control panel］－［Mount/Unmount file systems］

ディスク（ファイルシステム）をマウント、アンマウントする。マシン自体のディスク（local drives）、NFSドライブ共に操作可能だ（**画面28**）。

同様のことは［Config］タブの［File systems］でも可能だが、この［Mount/Unmount file systems］でマウントできるのは、設定ファイル（/etc/fstab）でマウントの設定がされているファイルシステムのみで、細かい設定の変更もできない。ただし、NFSドライブについてはその場での設定による自由なマウントが可能だ。




**画面30 ［date & time］**




**画面31 ［Features］**

### ■自動実行プログラムのスケジューリング

［Control panel］－［Configure superuser scheduled tasks］

Linuxには、定期的にプログラムを自動実行するcronという仕組みがある。この項目では、cronにプログラムを登録できる（**画面29**）。

まず、［Add］ボタンをクリックして定義用のパネルを表示させよう。［Command］には実行するプログラムをフルパスで指定する。それ以下は実行するスケジューリングの設定だ。＊を設定すればすべてを設定したことになる。曜日は0（日曜日）から順に数字で指定することになる。

最後に、［This entry is active］をチェックして設定を有効にしよう。

### ■日付時間の設定

［date & time］

Linuxマシンの日付、時刻の設定をすることができる（**画面30**）。

### ■キーボードや言語の設定

［Features］

Linuxconf自体における、キーボードや言語などの設定をすることができる。初期設定を変更する必要はほとんどないだろう（**画面31**）。

## 動作状況を確認する

［Status］タブでは、カーネルの起動時のメッセージや、Linuxconfで行った設定など、動作状況を記録しているログを見ることができる。この機能は設定というよりもログや動作状況に関する閲覧の機能だ。

### ■起動時のカーネルメッセージ

［Kernel boot messages］

起動時の動作状況を示すメッセージを表示する（**画面32**）。これは実際には、/var/log/dmesgに記録されているものだ。

### ■Linuxconfのメッセージ

［Linuxconf logs］

Linuxconfが、いつ、どのような変更を行ったかなどの動作記録を見ることができる（**画面33**）。

## Linuxconfの追加モジュール設定

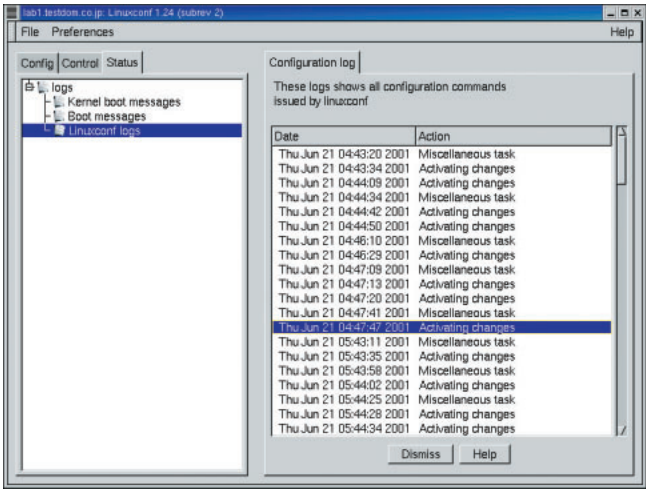
Linuxconfはモジュールにより機能を追加できるようになっている（**表1**）。

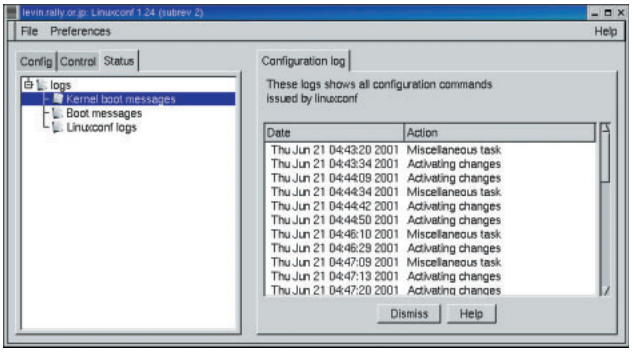
追加機能を有効にするには、［Control］タブの［Control files and systems］－［Configure Linuxconf modules］を使う（**画面34**）。リストの中の必要なモジュールをチェックしたら、［Accept］ボタンをクリックし

| モジュール | 設定できる内容 |
|---|---|
| accountbatch | 別のデータベースファイルによりユーザーを管理する |
| dhcpd | DHCPサーバを設定する |
| firewall | ファイアウォールを設定する |
| isapnpconf | ISA Plug and Playデバイスを設定する |
| kernelconf | 簡単なカーネルの設定する |
| liloconf | ブートローダLILOを設定する |
| mailconf | メールサーバsendmailを設定する |
| modemconf | 外部モデムの接続を設定する |
| motd | ログイン時のメッセージを設定する |
| opensshd | SSHデーモンを設定する |
| squid | キャッシュサーバsquidの設定をする |
| status | 起動時にマシンの状況を表示する |
| syslogconf | システムログを表示、設定する |
| wuftpd | FTPサーバwuftpdを設定する |
| Xkbdconf | Xで使うキーボードを設定する |

表1　おもなLinuxconfモジュール

画面33　［Linuxconf logs］の操作履歴
それぞれの項目をクリックすると、詳細を見ることができる。




画面32　［Kernel boot messages］

て、いったんLinuxconfを終了させ、再度起動しよう。新しい項目が追加されているはずだ。また、インストールされている以外にも、さまざまな機能を持ったLinuxconfモジュールのパッケージも配布されている（**画面35**）。

以下では、いくつかの便利なモジュールを紹介しよう。

### ■liloconf

［Config］［boot mode］－［lilo］

Linuxのブートローダ LILOの設定を行うためのモジュールだ。

最初のLinux起動画面のLILOプロンプトでの待ち時間から、新たに作成したカーネルをLILOに登録するなど、通常必要なことはほとんど設定できるはずだ（**画面36**）。


**画面35　LinuxconfのWebサイト**
最新のLinuxconfのほかに追加モジュールも配布されている。

### ■mailconf

［Config］［Networking］－［Server tasks］－［Mail delivery system］

Linuxで動作するサーバの設定の中で、もっとも複雑で面倒だといわれているのがメールサーバのsendmailの設定だ。これをGUIで設定できるのがmailconfモジュールで、基本的なsendmailの設定から、SPAM（迷惑メール）防止の設定まで可能だ（**画面37**）。

モジュールを追加すると［Config］の［User accounts］にも、［Email aliases］というメールアドレスを別名登録する機能が追加される。

### ■status

［Status］［viewing system state］

マシンの現在の状況を表示するためのモジュールだ。起動時に現在のマシンの状況が簡単に表示される（**画面38**）。さらに［viewing system state］という項目が追加され、さまざまな詳細が確認できるようになっている。

これらは、なんらかのコマンドの結果を表示する場合がほとんどだが、コマンド名を覚えていなくても一覧から選択して表示できるのが嬉しい（**表2**）。

### ■syslogconf

［Status］［Logs］－［System logs］

マシンの状況を記録するシステムログを閲覧および設定するためのモジュールだ。

［Status］タブの［Logs］－［System logs］で閲覧できるほか、［Config］タブの［Miscellaneous services］－［System logs］で設定を変更できる。また、メニューにも［System logs］項目が追加される（**画面39**）。

## テキストモードで使うlinuxconf

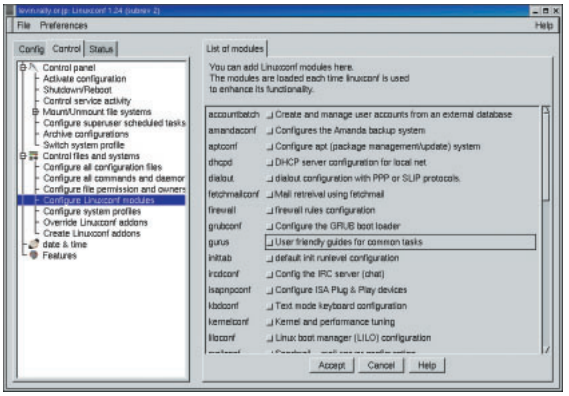
今回はGNOME上で使うLinuxconfを例に説明しているが、X Window Systemをインストールしていないときにも Linuxconfは使うことができる。

```
# linuxconf --text
```

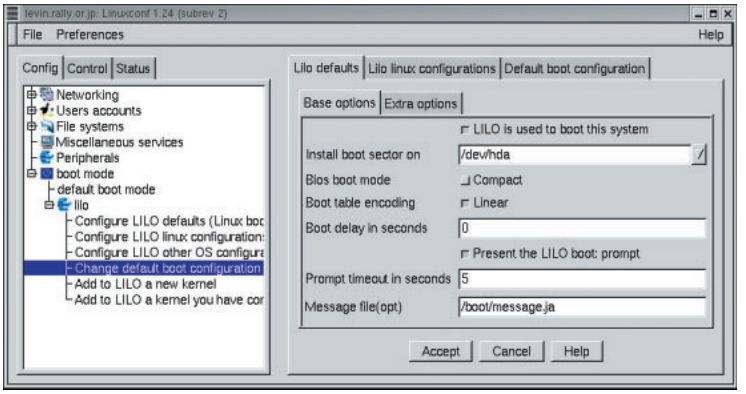
なお、X Window Systemが起動していない状態であれば、単に、

```
# linuxconf
```

とするだけで起動できる。ただし、テ




**画面34　［Control files and systems］－［Configure Linuxconf modules］**




**画面36　［boot mode］－［lilo］－［Change default boot configuration］**
**［Prompt timeout in seconds］で起動画面の待ち時間を設定しよう。**

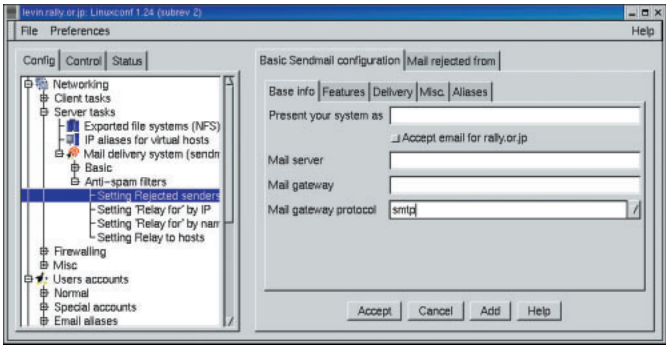
キストモードの画面が日本語化されているLinuxconfを使用する場合は(Vine Linuxなど)、konを実行してあらかじめ漢字コンソール上から起動する必要がある。

基本的な操作方法は、カーソルキーやTabキーで移動、Enterキーでメニューの展開、Ctrl+Xでプルダウンメニューの表示となっている。

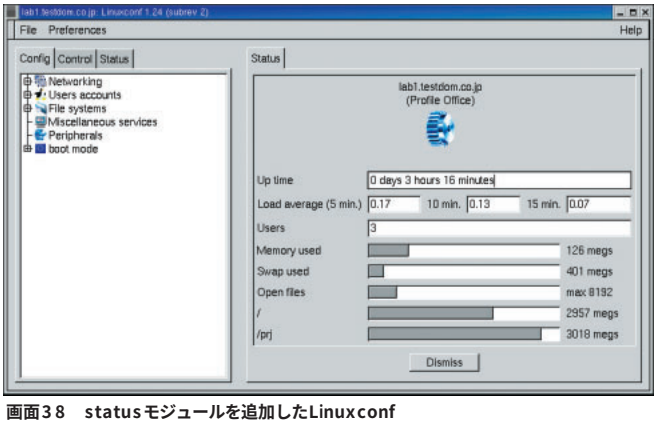
実は、LinuxconfにはGUIモードやテキストモードのほかに、Webブラウザから操作するWebモード、スクリプトなどから操作するのに便利なコマンドラインモードもあるが、これらのモードは通常使うことはあまりないだろう。

| | 項目 | 内容 | 対応するコマンド、ファイルなど |
|---|---|---|---|
| System | | | |
| | System information | ハードウェアを含むシステム全体の概要 | |
| | memory usage | メモリの使用状況 | /usr/bin/free |
| | disk usage | ディスクの使用状況 | /bin/df |
| | disk quota | ディスク利用制限の状況 | |
| | scsi devices | SCSIデバイスの状況 | /proc/scsi/scsi |
| | RAID (MD) usage | RAIDデバイスの状況 | /proc/mdstat |
| | PCI device | PCIデバイスの状況 | /sbin/lspci -v |
| | kernel device | カーネルモジュールの使用状況 | /proc/modules |
| Networking | | | |
| | network device | ネットワークインターフェイスの状況 | /sbin/ifconfig |
| | IP routing table | ルーティングテーブルの状況 | /sbin/route -n |
| | IPX routing table | IPXルーティングテーブルの状況 | /proc/ipx_route |
| CPU | | | |
| | prosesses | 現在動作しているプロセスのツリー表示 | /usr/bin/pstree |
| | interrupt mapping | IRQの使用状況 | /proc/interrupts |
| | I/O port mapping | I/Oポートの使用状況 | /proc/ioports |

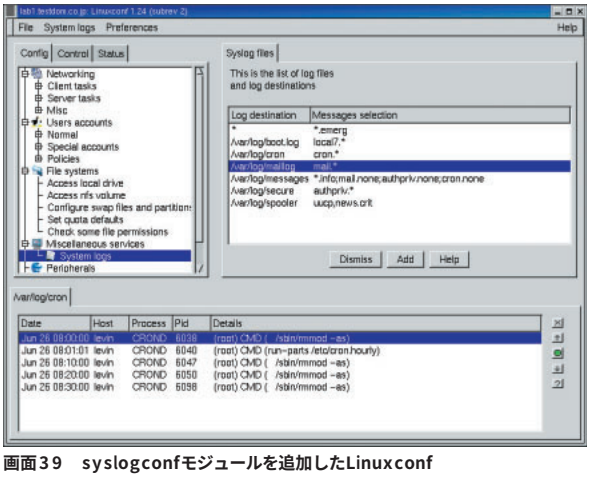
表2　Statusモジュールで追加される項目



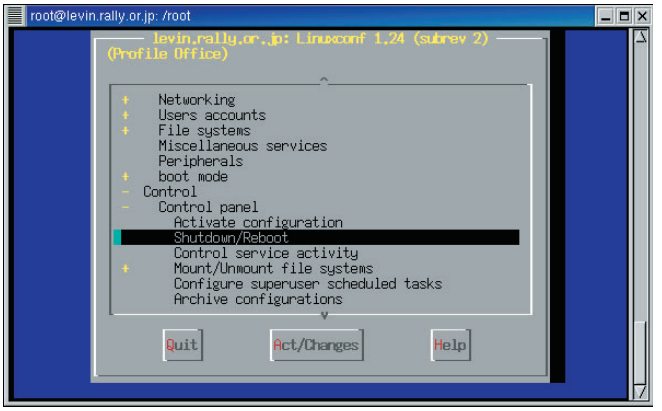

画面37　[Networking] − [Server tasks] − [Mail delivery system]
複雑な設定ファイルsendmail.cfを簡単に設定しよう。




画面38　statusモジュールを追加したLinuxconf
起動時に現在のマシンの状況が表示される。




画面39　syslogconfモジュールを追加したLinuxconf
見たいログをタブ形式で表示できる。




画面40　テキストモードで使うlinuxconf

# Control PanelでLinuxを設定する

文：香取精二
*Text : Seiji Katori*

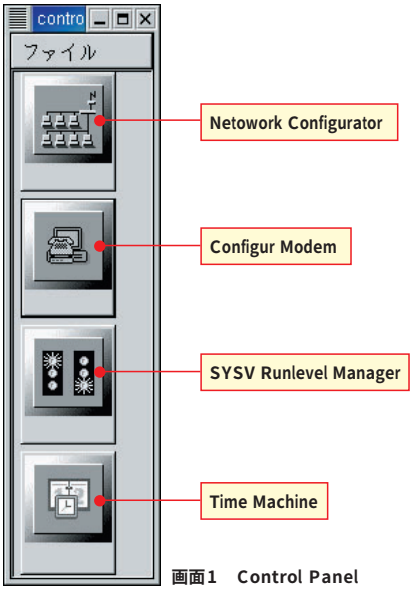
Linuxconfは確かに高機能なLinux設定ツールだ。ただその高機能ゆえに、ちょっとした設定のためにメニューの中から設定項目を探し出すのが面倒だ。また、パネルが増えていく形のインターフェイスになじまないという人もいるだろう。そんなときはシンプルなインターフェイスのControl Panelを使ってみよう。

Control Panelは、やはりX上で動作するGUIのLinuxの設定ツールだ（**画面1**）。ただし、設定できる項目はLinuxconfほど多くはない。また、Red Hat Linux 7.1におけるControl Panelは、以前のものよりも設定可能な項目が減っているのが少し残念だ。

Red Hat Linux 7.1で設定可能なのは、ネットワーク、モデム、ランレベル、日付と時刻だ。ただし、Linuxconfとは違い、普通にLinuxをインストールすれば必ずインストールされるので、追加インストールする必要がないのは大きなメリットだ。

Control Panelは、GNOMEメニューから起動してもよいし、コマンドラインから直接起動してもよい。

```
# control-panel
```




画面1　Control Panel

ただし、やはりシステム設定を行うツールなので、rootユーザーとして起動する必要がある。一般ユーザーの場合は、suコマンドなどでrootになってから起動しよう。

## Network Configurator

TCP/IPネットワークの設定をするのが、Network Configuratorだ。4つの設定項目があり上部のボタンをクリックして切り替える。もっとも、Linuxconfとは異なり、上部のボタンをクリックしても、タブが増えていくということはない。

設定はそれぞれの項目を設定したあと、下部の［Save］ボタンをクリックすれば全項目が保存される。終了はもちろん［Quit］だが、変更を加えていたとしても問答無用で変更を破棄してしまう。［Quit］の前に［Save］を忘れないように注意しよう。

### ■Names

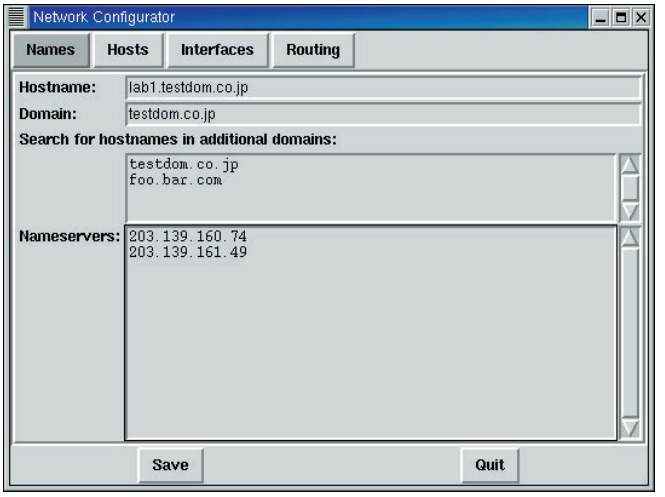
ホスト名ほか、基本的なネットワークに関する設定を行うのが［Names］だ（**画面2**）。

［Hostname］にはドメイン名を付加したホスト名を設定する。［Domain］はもちろんドメイン名だ。

［Search for hostnames in additional domains］には、ホスト名だけを指定してアクセスする場合に、補うべきドメイン名の候補を指定する。ここに何も指定しない場合は、マシンが所属するドメインのホストを探すことになる。

［Nameservers］には参照するDNSサーバのIPアドレスを優先する順に設定しよう。設定されていない場合は、次に説明する［Hosts］の設定を元にマシン名をIPアドレスに変換することになる。

このプログラムが実際に行うのは、/etc/HOSTNAME、/etc/resolv.conf、/etc/sysconfig/networkといったファ




画面2　Network Configuratorの［Names］

イルへの設定の書き込みだ。

DNSで解決できない、もしくはDNSを使わずに解決したいホスト名とIPアドレスの対応は、［Hosts］で指定しよう（**画面3**）。ここに記入された対応は/etc/hostsファイルに書き込まれる。

DNSの設定の手間を考えれば、マシンが数台のLAN環境などであれば、hostsファイルを利用してホスト名の名前の解決をするというのも現実的な対応だろう。

新たなマシンの追加は［Add］、既存のマシンの設定変更は、対象のマシン名をクリックして［Edit］、削除は［Remove］だ。［Nicknames］は付けなくてもかまわない。

## ■Interface

この項目では、マシンに存在するネットワークインターフェイスの設定をする（**画面4**）。

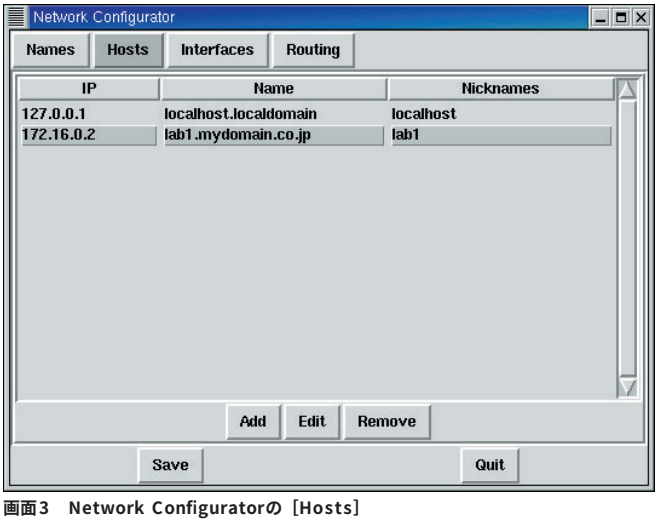
どんな場合にも存在するのが、loというインターフェイスで、これはループバックデバイスという自分自身を指すデバイスだ。今回はネットワーク自体の解説ではないので詳しい説明は省くが、このデバイスはそのままにして何も変更しなくてよい。

まず、新規にネットワークインターフェイスを追加する場合は［Add］をクリックする。ここで、**画面5**のような、ネットワークの種類を選択するウィンドウが表示される。まず、ほとんどの場合［Ethernet］を選択すれば間違いない。

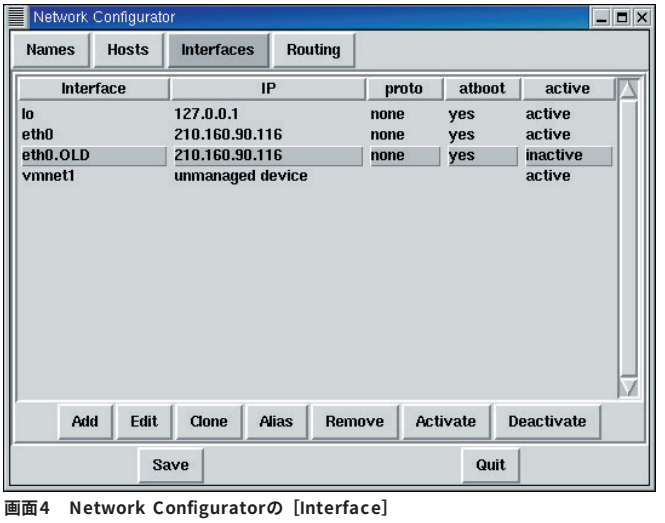
なお、以前のControl Panelではこの中にPPPという項目があったが、Red Hat Linux 7.1ではなくなっている。PPPインターフェイスに関しては、［ダイアルアップ設定］ツールで設定することになったようだ。もっとも、［ダイアルアップ設定］でPPPを設定すると、［Interfaces］のリストの中にPPPインターフェイスが現われ、設定の変更もできるようになる。

次に現れるのが、インターフェイスに対するIPアドレスなどの設定画面だ（**画面6**）。

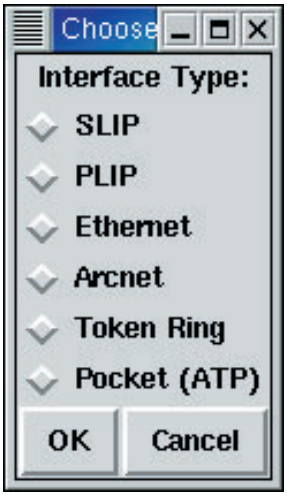
まず最初に決定するのが、［Interface



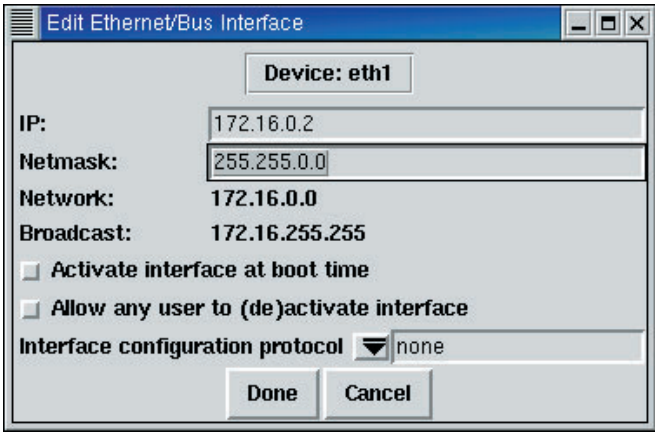

画面3　Network Configuratorの［Hosts］




画面4　Network Configuratorの［Interface］




画面5　インターフェイスの追加




画面6　インターフェイスの設定

configuration protocol］だ。DHCPやBOOTPを使っている場合は、それぞれを設定する。その場合、［IP］や［Netmask］は自動的に設定されるので設定する必要はない。

［Interface configuration protocol］がnoneならば、［IP］にIPアドレスを、［Netmask］にネットマスクを設定する。その結果、ネットワークアドレス（［Network］）とブロードキャストアドレス（［Broadcast］）は自動的に設定される。

さらに、このネットワークインターフェイスを起動時に有効にするのなら、［Activate interface at boot time］をチェックする。通常はチェックするべきだろう。そうでないと、ネットワークを使いたいと思うたびにインターフェイスを［Active］にしなければならない。

なお、［Allow any user to (de)activate interface］を有効にすると、rootユーザー以外にもこのネットワークの有効／無効を操作することができるようになる。

これらを設定して［OK］をクリックすれば、ネットワークインターフェイスが追加される。

［Interface］で通常使われる下部のボタンは、インターフェイスを追加をする［Add］、既存のものを編集する［Edit］、削除する［Remove］だが、それ以外にも便利な機能がある。それが、既存のインターフェイスの複製を作る［Clone］だ。

［Clone］を使い複製を作成すると、ある1つのインターフェイスに複数のIPアドレスを割り当てるなどの設定が可能だ。この異なったIPアドレスは同時には使えないが、［Active］や［Deactive］を使ってそれぞれのインターフェイスの有効／無効を切り替えることにより、たとえば仕事場と家といった異なった複数のネットワーク環境で、うまくネットワークを切り替えて使うことができる。また、新しい設定を試してみたいというときにも、［Clone］をうまく使うと、古い設定を消去することなくテストできるだろう。

このプログラムが実際に行うのは、/etc/sysconfig/network/ifcfg-eth0といった各インターフェイス定義ファイルへの設定の書き込みだ。

### ■Routing

Network Configurationの最後になるのが、［Routing］だ（**画面7**）。デフォルトゲートウェイやスタティックルーティングの設定を行う。

スタティックルーティングを設定しない場合は、デフォルトゲートウェイの設定のみでかまわない。［Default Gateway］にゲートウェイマシンのIPアドレスを、［Default Gateway Device］にネットワークインターフェイスの名前（通常はeth0など）を設定する。

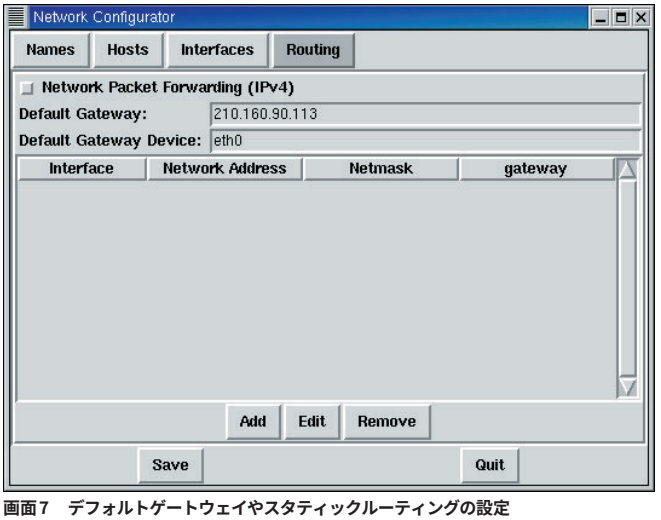
もし、ネットワークカードを複数枚使ってゲートウェイマシンとするならば、さらにスタティックルーティングを設定する必要がある。

その場合はまず、［Network Packet Forwarding］をチェックして、それぞれのネットワークインターフェイスに対して、ネットワークアドレス、ネットマスクなどを設定していく（**画面8**）。

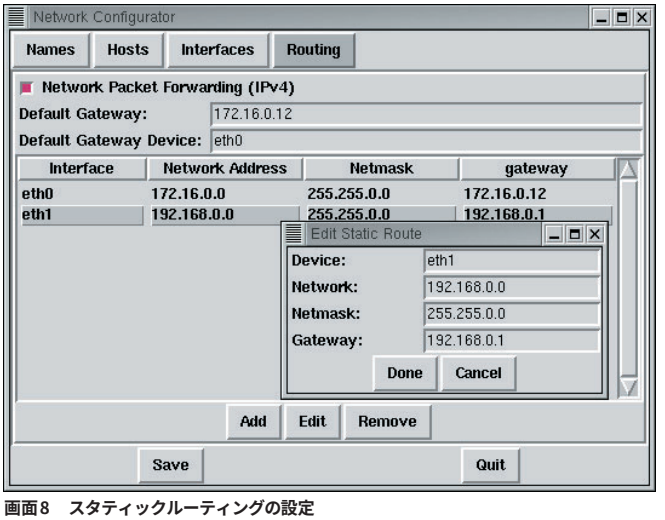
このプログラムが実際に行うのは、/etc/sysconfig/networkファイルへの設定の書き込みだ。

## Configure modem

Linux上で動くプログラムでは、通常、モデムは/dev/modemというデバイスとしてアクセスされる。しかし、Linux自身はコンピュータに複数存在するどのシリアルポートにモデムが接




画面7　デフォルトゲートウェイやスタティックルーティングの設定




画面8　スタティックルーティングの設定

続されているかは判断できない。

そこで、このツールで/dev/modemを、モデムが接続されたシリアルポートにリンクしてやる必要があるのだ。

モデムが接続されたシリアルポートをチェックして［OK］ボタンをクリックするだけなので、難しいことはないはずだ（**画面9**）。

このプログラムが実際に行うのは、シリアルポートのデバイスファイル（/dev/ttyS？）とモデムのデバイスファイル（/dev/modem）とのシンボリックリンクの作成だ。

## SYSV Runlevel Manager

Linuxではいくつかの動作モードがあり、それぞれをランレベルという値で区別している。そして、システムではランレベルに応じてどのようなデーモンプログラムを起動するか、停止させるかがあらかじめ設定されている。

インストール時点では、プログラムのランレベルへの対応はほぼ適切に設定されているし、RPMパッケージなどからインストールした場合も、ランレベルとの対応の設定をしてくれる場合がほとんどだ。ただし、中には自分で設定しなくてはならないものもあるし、ソースからコンパイルした場合などは自分で設定しなくてはならない場合も多い。

SYSV Runlevel Managerは、それぞれの動作モードでどのようなプログラムを起動するか、停止するかを設定するツールだ（**画面10**）。

なお、Red Hat Linuxなどでは、ランレベルの値と動作モードは**表**のように対応している。SYSV Runlevel Managerで設定できるのはランレベル2、3、4、5だ。ただし、実際によく使われるのはランレベル3のテキストログインのマルチユーザーモードと、ランレベル5のグラフィカルログインのマルチユーザーモードだろう。

SYSV Runlevel Managerの画面は大きく3つに分かれている。左側が制御できるプログラムのリスト、右上が各ランレベルで起動するプログラム、右下が各ランレベルで停止するプログラムだ。

制御するプログラムを各ランレベルに追加したい場合は、左の［Available］の中のプログラムをクリックし、［Add］ボタンをクリックする。そこで表示される**画面11**のようなウィンドウで、どのランレベルで起動したいのか（［Start］）、停止したいのか（［Stop］）を選んで、［Done］をクリックしよう。

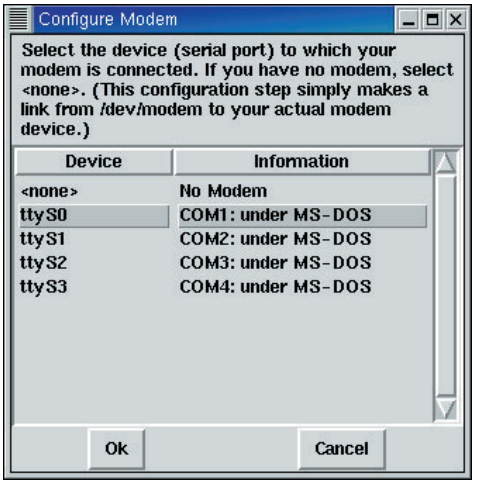
逆にランレベルによって制御するのをやめたいプログラムがある場合は、右側のリストの中から、対応するランレベルの中のプログラムをクリックして、［Remove］をクリックすればよい。リストの中からプログラムが消えるはずだ。

さらに、それぞれのプログラムを起動、停止する順番を制御したい場合もあるだろう。そういったときは、目的のランレベルの［Start］、［Stop］の中から適当にプログラムをクリックして、［Edit］をクリックしてみよう。ここで、**画面12**のような順番を制御するためのウィンドウが開くはずだ。上部のリストの頭の数字が実行される順番を示している。そこで希望の順番になるようにそれぞれのプログラムを選んで、［Change selected item's number to］に数字を入れて実行順を変更していけばよい。

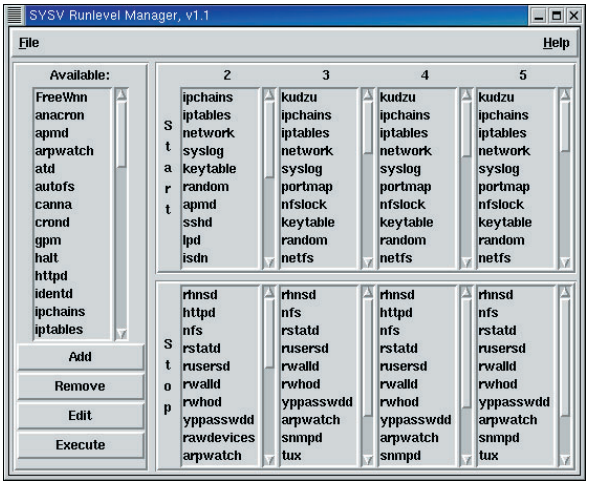
また、単に［Available］の中にリストアップされているプログラムを選ん

| ランレベル | 動作モード |
|---|---|
| 0 | 停止 |
| 1 | シングルユーザーモード |
| 2 | マルチユーザーモード(ネットワーク機能なし) |
| 3 | マルチユーザーモード(テキスト) |
| 4 | 未使用 |
| 5 | マルチユーザーモード(X) |
| 6 | 再起動 |

表　ランレベルの値による動作モード




画面9　Configure modem




画面10　SYSV Runlevel Manager

で［Execute］をクリックすれば、ランレベルに関係なくプログラムをその場で起動、停止することもできる。

以上で設定された内容は、実際には/etc/rc.d/init.d以下にある各プログラムの起動／停止用のスクリプトが、/etc/rc.d/rc？.d以下のディレクトリに、起動するならSで始まる名前で、停止するならKで始まる名前でリンクされる。

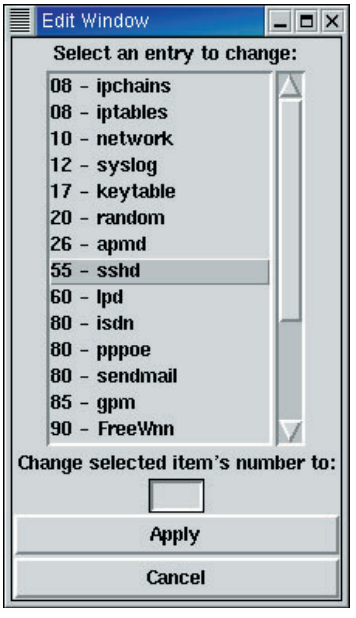

画面11　SYSV Runlevel Managerでのプログラムの追加

## Time Machine

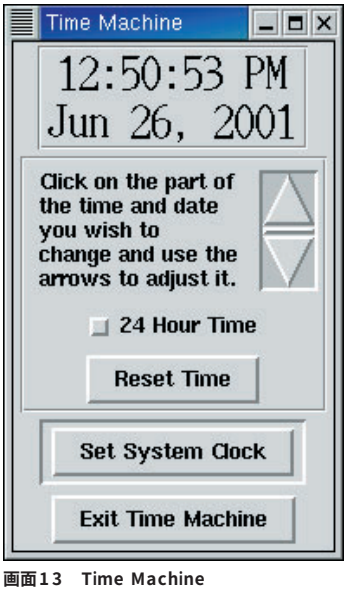
Time Machineはマシンの日付時刻を変更するためのツールだ（**画面13**）。

使い方は簡単だ。まず、上部の日付時刻の中から変更したい部分をマウスでクリックしよう。すると、変更の対象部分が赤くなるので、矢印ボタンで値を変えていけばいい。数値を直接入力できないのがちょっと面倒といえば面倒だ。時刻を合わせたら最後に［Set System Clock］ボタンをクリックすればよい。

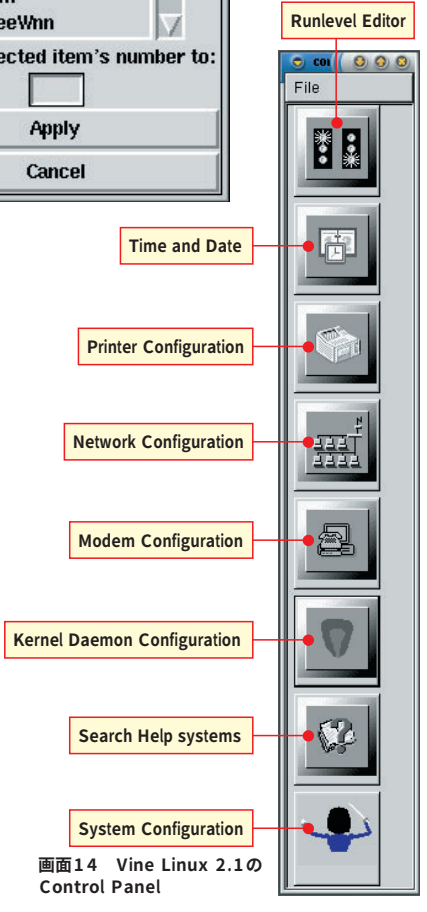
なお、メッセージウィンドウでも警告されるが、日付の変更を行った場合は、日付に依存するプログラムが誤動作をしないように、一度システムそのものを再起動したほうが無難だろう。




画面12　プログラムの起動停止順の制御




画面13　Time Machine




画面14　Vine Linux 2.1のControl Panel

## Red Hat Linux 7.1以外のControl Panel

Red Hat Linux 7.1以外のControl Panelでは、今まで説明してきた4つ以外の機能が含まれていることが多い。実際のところは、Red Hat Linux 7.1においてツールが削られたのだ。

その理由には、Red Hat Linux 7.1が新機能を搭載したため、Control Panelのツールが対応できなくなったり、Red Hat Linux独自の設定ツールが作成されたりしたため、Control Panelのツールが必要なくなったという経緯がある。

ここでは、ほかのディストリビューションを使っている読者のために、そのほかのツールについても解説しよう。以後の解説についてはVine Linux 2.1のControl Panelを元に行う（**画面14**）。なお、VineのControl Panelは、日本語に強いVine Linuxの定評どおり、日本語化が進んでおり、わかりやすいものになっている。

### ■Printer Configuration

［Printer Configuration］を使うと、簡単にプリンタの設定ができる。Linuxも、現在では数多くのプリンタに対応しており、いままで面倒だったプリンタドライバの選択と設定といった作業をかなり楽に行うことができるだろう。

プリンタを何も設定していない状態で［Printer Configuration］を起動すると、まず空のリストを持ったプリンタシステムマネージャが表示される。そこで、［追加］を選ぶとプリンタの種類を設定するためのウィンドウが表示される。もちろん、自分の使いたい種類のプリンタを選ぶわけだが、今回は［ローカルプリンタ］を選択しよう。すると自動検出が行われ、**画面15**のようなウィンドウが表示される。［入力フィルタ］の［選択］をクリックして、リストから自分の使っているプリンタを選んで［了解］をクリックしよう。さらに設定編集画面でも［了解］をクリックすればプリンタの設定は完了だ。

最後に、［テスト］メニューから好きなテストを選んでプリンタの設定を確認しよう。

設定された内容は、/etc/printcapファイルへ書き込まれる。

## Kernel Daemon Configuration

Linuxではさまざまな機能をカーネルのモジュールで実現している。そして、その機能を使うためにはモジュールの組み込みを設定しなければならないのだが、どのモジュールがどの機能に対応しているのかがわかりにくく面倒だ。［Kernel Daemon Configuration］は、この面倒な作業を助けてくれるツールだ（**画面17**）。

モジュールの追加は、［追加］をクリックして、［モジュールの種類］、［モジュールの定義］といった質問に答えていくだけだ。追加が完了したら［kerneldをリスタートする］をクリックすれば、モジュールは有効になる。

このツールが実際に行うのは、/etc/modules.confファイルへの設定の書き込みだ。

## Search Help systems

設定したドキュメントの中から、文字列を検索してくれるのが［Search Help systems］だ。

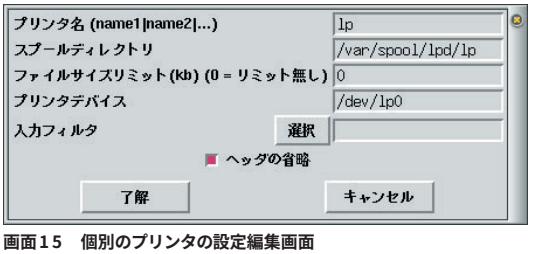
Linuxには、オンラインヘルプとしてmanコマンドやinfoコマンドも用意されているが、このツールでは一度にmanやinfoを含む数々の文書から情報を検索してくれる（**画面18**）。もちろん、検索結果のリストをクリックすると文書が表示される。

manファイルがなくinfoによるドキュメントしか用意されていないプログラムも多いので、こういったツールは非常に有用だろう。

## System Configuration

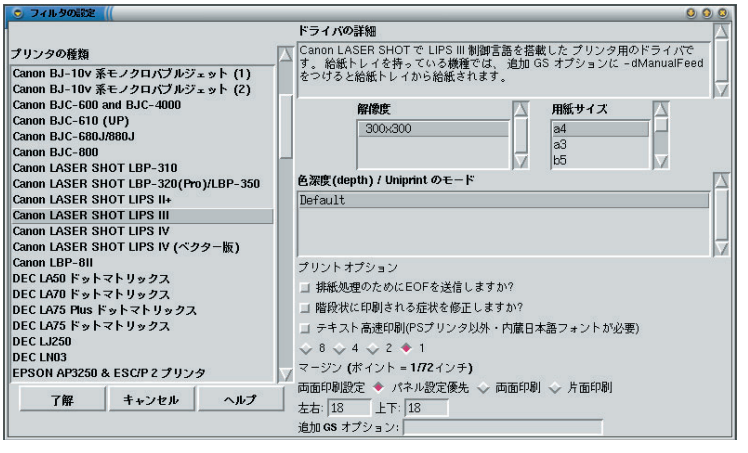
このアイコンをクリックすると、単にLinuxconfが呼び出される。直接Linuxconfを起動した場合と動作は変らないので、Linuxconfのページを参考にしてほしい。

ここまでで紹介した2つのツールを使うことで、Linuxのほとんどの設定を行うことができる。いつもは設定ファイルを直接変更しているユーザーも、簡単に設定できるGUIツールを使ってみてはいかがだろうか。



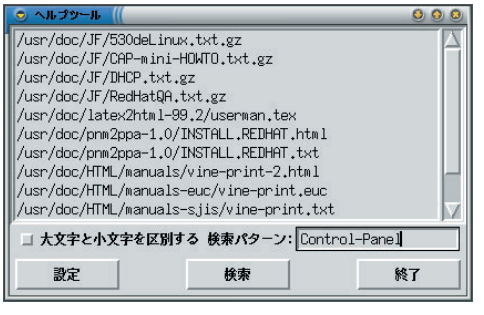

画面15　個別のプリンタの設定編集画面

画面16　充実したプリンタサポート





画面17　Kernel Daemon Configuration




画面18　Search Help systemsで「Control-Panel」を検索

# GNOMEコントロールセンターでデスクトップ環境を変える

文：編集部
Text : Linux magazine

最初のパートで解説したLinuxconfは、あらゆるシステム設定を行うことができる優れたツールだが、デスクトップ環境の設定を変更することはできない。クライアント用途など、X Window Systemを十分に使いこなすためには、デスクトップ環境のカスタマイズ方法を知っておきたいところだ。ここでは、GNOMEデスクトップの設定ツール「GNOMEコントロールセンター」について解説する。

## 起動方法

Linuxconfと同じく、Red Hat Linux 7.1をベースに説明していく。Red Hat 7.1では、GNOME 1.2（gnome-coreのバージョンは1.2.4）が採用されており、GNOMEコントロールセンター（以下、コントロールセンター）のバージョンは1.2.2だ。GNOMEの最新バージョンは1.4だが、コントロールセンターの設定項目について大きな変更点はないようである。

コントロールセンターは、メインパネルにある道具箱（？）のアイコン（**画面1の左端**）をクリックして起動する。また、メインメニューの［プログラム］－［デスクトップ設定］－［GNOMEコントロールセンター］でも起動できる。root権限は必要なく、一般ユーザーでも実行可能だ。ちなみに、コントロールセンターの実行ファイル名は「gnomecc」である。

画面1　GNOMEコントロールセンターの起動アイコン

## 初期画面と基本操作

起動直後の初期画面は**画面2**のようになる。ウィンドウの左側のペインに設定項目のグループがツリー表示され、選択されたグループの設定ダイアログが右側に表示される（**画面3**）。ダイアログには、［試行］、［戻す］、［OK］、［キャンセル］の4つのボタンがある。特に使い方がわからないものはないと思うが、念のため各ボタンについて簡単にまとめておこう。

- **［試行］ボタン**

ダイアログで行った設定を試すためのボタン。デスクトップの背景など、すぐに効果がわかる設定項目の場合に有効。クリックすると設定が反映されるが、保存はされない

- **［戻す］ボタン**

ダイアログで行った設定を変更前の状態に戻すためのボタン。元の設定がわからなくなった場合などに有効

- **［OK］ボタン**

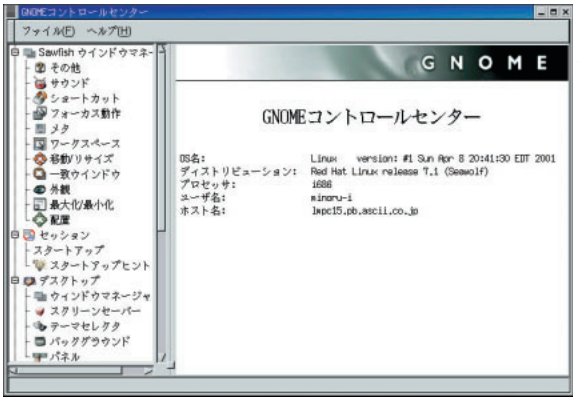
ダイアログで行った設定を反映・保存する。一度クリックすると［戻す］ボタンも［キャンセル］ボタンも利かなくなるので注意

- **［キャンセル］ボタン**

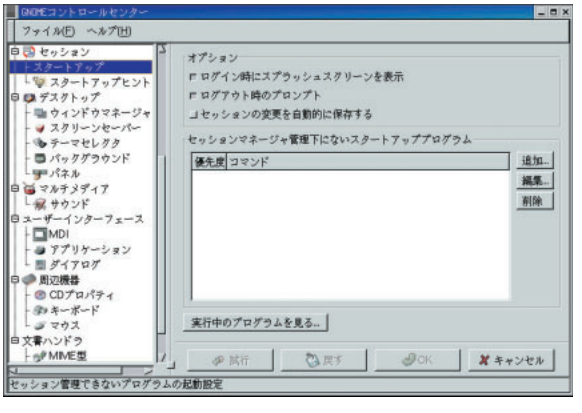
ダイアログで行った設定変更を取り消す。設定を変更して［OK］ボタンを押す前の状態のときのみ有効

特に便利なのが［試行］ボタンだ。デスクトップの外観を変更する際などに、効果を確認しながら設定でき、［戻る］ボタンで元の状態に戻すこともできる。あれこれ試しながら、好みの設定を探っていくといいだろう。

なお、コントロールパネルでの設定内容は、設定を行ったユーザーのホームディレクトリに保存される。つまり、ユーザーごとに個別のデスクトップ環




画面2　起動直後のコントロールセンター
初期状態では右側のペインにユーザー名などの情報が表示される。




画面3　設定ダイアログの例
設定グループを選択すると右側にダイアログが表示される。

境を好みに合わせて構築できるわけだ。

以下ではコントロールセンターの各セクションごとに、設定項目について解説していく。項目数が非常に多いため、すべてを詳細に説明することはできないが、重要と思われる項目を中心になるべく網羅的に取り上げる。

## セッションの設定

［セッション］セクションには、［スタートアップ］と［スタートアップヒント］の2つの設定グループがある。

### ■［スタートアップ］

［スタートアップ］ダイアログでは、起動・終了時のオプションを設定する。

［オプション］にある3つの設定項目はログイン・ログアウト時のオプションだ。ログイン時のスプラッシュスクリーンとログアウトプロンプトの表示／非表示、ログアウト時のセッションの自動保存のオン／オフを切り替えることができる。

［セッションマネージャ管理下にないアプリケーションのスタートアッププログラム］は、ログイン時に通常のアプリケーションを自動起動するための設定。［追加］ボタンを押すと、**画面4**のウィンドウが表示されるので、自動起動したいプログラムの実行ファイル名を指定しよう。メーラやスケジューラなど、ログイン時にチェックしておきたいアプリケーションを指定しておくと便利かもしれない。

### ■［スタートアップヒント］

ログイン時に表示されるGNOMEの使い方に関するヒントメッセージ（**画面5**）をオン／オフする。

通常のヒントの代わりに、フォーチュンメッセージ（日めくりに印刷してある格言のようなもの）や任意のテキストファイルの内容を表示するよう設定することも可能だ。

## デスクトップの外観とパネルの設定

［デスクトップ］セクションの各設定グループでは、デスクトップの外観、スクリーンセーバー、パネルなどを設定する。

### ■［ウィンドウマネージャ］

GNOMEは特定の専用ウィンドウマネージャを持たない統合デスクトップ環境である。GNOME開発プロジェクトが決めたガイドラインに沿ったウィンドウマネージャであれば、任意のものを使用することができる。

ただ、Red Hat 7.1を始めとする多くのディストリビューションで採用されているウィンドウマネージャ「Sawfish」は、GNOMEとの親和性が高く、独自のカスタマイズ機能を備えており、使い勝手が良い。特にこだわりがないのなら、デフォルトのままSawfishを使うことをお勧めする。

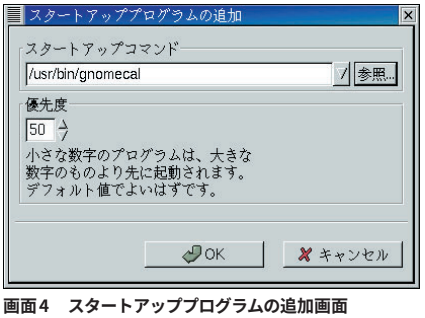
### ■［スクリーンセーバー］

スクリーンセーバーの種類や起動のタイミングを設定する。Red Hat 7.1では、デフォルトで「ランダムスクリーンセーバー」が20分後に起動するように設定されている。

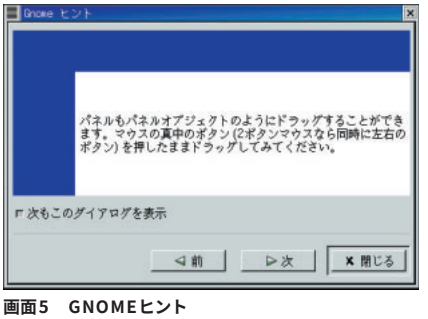
リストからスクリーンセーバーを選択すると、右側にデモが表示されるので、気に入ったものを設定しよう（**画面6**）。Xの設定によっては3Dスクリーンセーバーのデモが表示されないことがある。デモが表示されないものは設定できても動かないので気をつけてほしい。

［パスワードを要求する］をチェックすると、スクリーンセーバーからの復帰時にログインパスワードの入力が必要になる。環境によって、セキュリティを考慮したい場合などには、この設定を有効にしておくほうがよい。

スクリーンセーバーの起動後、指定した時間が経過したあとにディスプレイをスリープ状態にするオプションも用意されている。ただし、このオプションを有効にするためには、APMがきちんと設定されていて、apmデーモンが起動している必要がある。




**画面4　スタートアッププログラムの追加画面**
［参照］ボタンを使ってプログラムの実行ファイル名を指定できる。




**画面5　GNOMEヒント**
［次もこのダイアログを表示］をオフにして、ここで無効にすることも可能。


**画面6　スクリーンセーバーの設定**
各スクリーンセーバー独自の設定は［～の設定］ボタンで行う。

■［テーマセレクタ］

テーマとは、ボタンやリストボックス、ウィンドウ枠といったGUI部品をひとつのセットとしてまとめたものだ。その名のとおり、何らかのテーマに沿った統一感のあるデザインのGUI部品が用意されている。テーマを切り替えることで、デスクトップの外観をガラリと変えることができるのだ（**画面7**）。

GNOMEのデスクトップ環境では、ウィンドウ枠やダイアログのテーマを2つのパートに分けて管理しており、ウィンドウ枠についてはSawfish（ウィンドウマネージャ）、ダイアログのボタンやフォントについてはGTK＋（GUIライブラリ）のテーマをそれぞれ別に設定できるようになっている。ここで設定するのは、GTK＋のテーマだ。

［使用可能なテーマ］のリストからテーマを選択すると、ダイアログの下段にある［試写］にそのテーマで使われるボタンなどの外観が表示される（**画面8**）。試写だけでは効果が掴みづらいときには、GIMPなどのGNOME（GTK+）アプリケーションを起動しておき、［試行］ボタンを押して実際に確認するという方法もある。

［新規テーマのインストール］ボタンを使ってテーマを追加することもできる。GTK＋のテーマはtar＋gzip形式で配布される。このボタンを押してテーマのtarボールを指定すれば、インストールは完了だ。テーマの入手先としては、themes.org（http://www.themes.org/）がお勧めである。

■［バックグラウンド］

デスクトップの背景（壁紙と色）を設定する。

Red Hat 7.1のデフォルトであるエンボス効果を付けたロゴの壁紙もなかなか渋いが、お気に入りの写真の画像ファイルを壁紙にすることも、もちろん可能だ。そのほか、小さな画像ファイルをタイル状に並べたり、画面サイズに拡大して表示することもできる。

［色］のセクションでは、背景色を指定する。単色で塗るか、2つの色を指定して垂直または水平方向にグラデーションをかけることができる。デフォルトの設定を参考にしながら、いろいろ試してみよう。

［背景の設定にはGNOMEを使う］は、EnlightenmentやWindow Makerのように背景の設定機能を持ったウィンドウマネージャとGNOMEを組み合わせている場合に、どちらの機能を優先するかを指定するオプション。Sawfishを使っている場合は、デフォルト設定（GNOMEを優先）のままにしておく。

■［パネル］

このダイアログは設定項目が多岐にわたるため、各項目の説明を**表1**にまとめておくので参考にしてほしい。パネルの設定に関しては、ここでの設定だけでなく、パネル自体のプロパティもかかわってくる。メインメニューの［パネル］－［プロパティ］で表示される項目もあわせて設定する必要があるので注意すること。

## サウンドと周辺機器の設定

［マルチメディア］セクションと［周辺機器］セクションでの設定をまとめて説明しよう。

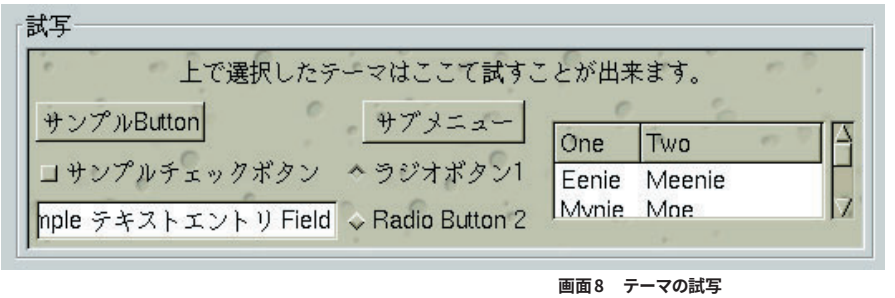
■［サウンド］

［マルチメディア］セクションの［サウンド］では、サウンドサーバとサウンドイベントの設定を行う。

［起動時に音サーバを使う］をチェックするとGNOMEの起動時に自動的にサウンドサーバデーモン（esd）が立ち上がる。ボリュームメータなど、サウンド関連のGNOMEアプレットの一部はesdのみサポートしているので、これらのアプレットを使う場合にはチェックを忘れずに。

［イベントに対して効果音を鳴らす］をチェックするとサウンドイベント機能が有効になる。メッセージの表示やメールの到着などに合わせて、指定したサウンドファイル（.wav）を効果音として再生できるようになる。使い方


画面7　テーマの変更例
GTK＋とSawfishのテーマを設定することでウィンドウの外観を変更できる。デフォルトの外観と比べてみよう。




画面8　テーマの試写
GTK＋のテーマを変更するダイアログでは試写が可能だ。

によっては、便利な機能なのでいろいろ工夫してみるのもおもしろいかもしれない。詳細な設定は［サウンドイベント］タブで行う。

### ■［CDプロパティ］

［周辺機器］セクションの［CDプロパティ］では、CDドライブへのメディアのセット時に実行する処理を設定する。データCDとオーディオCDの2つについて別々の設定ができるようになっている。

データCDについては、「自動マウント」、「autorunの実行」、「マウント時のファイルマネージャの自動実行」の3項目をオン／オフできる。

オーディオCDについては、セットされた際に実行するコマンドを指定できる。デフォルトでは、GNOME CDプレーヤ（gtcd）がセット時に自動的に起動し、再生がスタートするよう設定されている。gtcd以外の再生ソフトを使っている場合は、ふだん使っているソフトのコマンドオプションを調べて、設定を変更するとよいだろう。

### ■［キーボード］

このダイアログでは、キーボードに関する設定を行う。

［自動的にリピートする］では、キーを押しっぱなしにした際に、自動的にその入力を繰り返すかどうかを設定する。ただし、この設定はターミナルエミュレータに対しては有効にならないことに注意してほしい。

［リピート回数］は最大リピート数、［リピートまでの間隔］はリピートを開始するまでの遅延時間（ミリ秒単位）の設定だ。

［キーボードのクリック音］にある項目は、読んで字のごとくクリック音を鳴らすかどうかと、鳴らす場合の音量の設定だ。

［キーボードベル］では、キーボードエラーの通知に使われるビープ音の音量などを設定する。［ピッチ］でビープ音の高低、［長さ］でミリ秒単位でのビープ音の長さを設定できる。

| 設定項目 | 説明 |
|---|---|
| ［アニメーション］タブ | |
| アニメーションを有効にする | パネル操作時のアニメーション効果のオン／オフ |
| アニメーションは一定速度にする | アニメーションを描画する際に加速するかどうか |
| ［ボタン］タブ | |
| ボタンタイプ | 設定の対象となるボタンタイプを指定する |
| タイルを使用 | チェックすると指定した画像ファイルがタイルとして使用される |
| 高速だが低画質のボタン・アイコン | 解像度の低いアイコンを使用する。性能の低いPCなどで有効 |
| ［パネルオブジェクト］タブ | |
| デフォルトの移動方式 | パネルオブジェクトの移動方式を切り替える |
| 間隔 | パネルオブジェクトの最小間隔を設定する |
| ［メニュー］タブ | |
| 大きいアイコンを使う | メニューアイテムの左に表示されるアイコンを大きいものにする。1280×1024ピクセル以上の解像度で有効 |
| ［...］ボタンを表示 | ［...］ボタンの表示／非表示を切り替える。［...］ボタンを押すと、メニューアイテムをパネル上で操作する（ランチャの追加など）ためのポップアップメニューが表示される |
| メニューをメモリ内に保持 | チェックするとメモリにメニューを保持することでより高速に処理できるようになる |
| システムメニューを組み込む | 「システムメニュー」として設定されているアイテムをパネルのメニューに組み込むかどうかを切り替える |

表1　パネルの設定項目

### ■［マウス］

このダイアログでは、［マウスボタン］で右利き／左利きのボタン設定の切り替え、［マウスの動き］でマウスポインタの移動スピードを設定する。

ちょっとわかりづらい項目が［しきい値］だろう。これは［加速］に指定したスピードでマウスポインタが移動を始める「区切り」の値で、やはり速さが基準になっている。つまり、ここに指定した値を超えたスピードでマウスポインタが移動すると動きが加速していくようになる。

通常は［加速］、［しきい値］ともデフォルトの設定で問題ないと思われるが、画像ファイルの加工など細かなマウス操作が必要な場合など、必要に合わせて設定するようにしたい。

## MIMEの設定

［文書ハンドラ］セクションでは、MIME型、URLを取り扱うアプリケーション、標準エディタを設定する。

### ■［MIME型］

MIME型とは、たとえば「拡張子が.txtのプレーンテキストファイルを編集するアプリケーションはEmacs」であるとか、「拡張子が.jpgの画像ファイルはGIMPで処理する」といったように「ファイルの種類」と「処理を行うアプリケーション」とを関連づけたもの。ここでの設定は、ファイルマネージャ（GMC）やデスクトップに反映され、特定のファイルのアイコンをダブルクリックすると指定されたアプリケーションが自動的に起動し、そのファイルがオープンされるようになる。

**画面9**は「audio/x-mp3」のMIME型の編集ウィンドウを示している。画面からもわかるとおり、拡張子が「.mp3」のファイルを再生するアプリ

ケーションとして「mpg123」が指定されている（“%f”はファイル名をパラメータとしてコマンドに渡すための設定）。この設定によって、MP3ファイルのアイコンをダブルクリックした際に、そのファイルがmpg123で再生される。

MIME型をうまく設定することで、デスクトップの使い勝手を大幅に向上させられる。デフォルトの設定を見直して、使い慣れたアプリケーションが起動するように変更するだけで、ずいぶん違うはずだ。あれこれ工夫してみる価値はある。

### ■［URLハンドラ］

このダイアログでは、デスクトップアイコンに指定されているURLを処理するアプリケーションを設定する。また、manページやGNOMEヘルプなどを表示するアプリケーションも設定できる。

デフォルトではURLを処理するプログラムがNetscape Navigator、Man／GNOMEヘルプ／infoファイルを処理するプログラムがGNOMEヘルプブラウザとなっている。


画面9　MIME型の設定画面
［MIME型による動作］の各項目は、ファイルアイコンを右クリックしたときのポップアップメニューに対応している。

### ■［標準エディタ］

GNOMEデスクトップ環境の標準エディタを設定する。具体的には、「gnome-edit」コマンドで起動するエディタの指定である。デフォルトではEmacsが標準エディタとなっている。

## そのほかの設定項目

ここまでに触れていないセクションについて簡単に説明しておこう。

### ■［Sawfishウィンドウマネージャ］

Red Hat 7.1のGNOME環境の標準ウィンドウマネージャであるSawfishの設定を行うセクションだ。非常に項目数が多いため今回は詳細に解説できないが、いくつかの有用な設定項目を紹介しておくことにしよう。

［外観］ではSawfishのテーマ機能を設定する。［デフォルトフレームスタイル］でテーマを選択し、［試行］ボタンを押すとウィンドウ枠が変更される。ウィンドウ枠の色やウィンドウタイトルで使われるフォントなども設定できる。GNOME（GTK+）のテーマとマッチする設定を選ぶようにするとよいだろう。

［フォーカス動作］のデフォルト設定（Red Hat 7.1）は、アプリケーションウィンドウの領域をマウスクリックした時点で入力フォーカスを得るという設定になっている。

フォーカス動作を「enter-only」に設定し、［フォーカス時にウィンドウを前面に出す］をチェックすると、アプリケーションウィンドウ上にマウスポインタが移動することで、そのウィンドウが前面に表示され、入力フォーカスを得られるようになる。好みによって違いはあるだろうが、一度試してみてほしい。

［配置］では、アプリケーション起動時のウィンドウ表示位置を設定できる。デフォルトでは、デスクトップの左上に表示されるよう設定されており、アプリケーションが独自にウィンドウ表示の制御を行っている場合にはそれを優先するようになっている。とても使いやすい設定とはいえないので、［ウィンドウの配置方法］を「best-fit」または［interactively］にして［プログラムの指定するウィンドウの位置を無視］をチェックするなど、少し設定を工夫したいところだ。

「best-fit」を指定した場合、Sawfishが「なるべく使いやすそうな」位置にウインドウを配置してくれる。「interactively」の場合は、ウィンドウの枠のみが表示され、マウス操作で任意の位置にウィンドウを配置できる。少々面倒だが、好きな位置でアプリケーションを開始できるので結構便利である。

### ■［ユーザーインターフェイス］

GNOMEアプリケーション（geditやGnumericなど）のインターフェイスを


画面10　Sawfishの各種テーマ
これらは標準で用意されているものの一部だ。このほかにも、豊富なテーマが用意されている。

設定するセクション。

アプリケーションの側で独自に設定できる項目と重複する点などもあるようで、十分に整理されていない感じを受けた。特に気にしなければならない項目もないので、ほとんどの場合、デフォルトの設定で問題ないはずだ。

## メニューのカスタマイズ

今回検証に使用したRed Hat 7.1に限らず、標準で用意されているGNOMEのメニューは、非常に冗長なものだ。もちろん、いろいろなアプリケーションがメニューにあらかじめ登録されているのは、ありがたいことではある。「大は小を兼ねる」ということもあるのだろう。

ただ、常用するアプリケーションはユーザーによってほとんど決まっているだろうから、それらのプログラムをすみやかに起動できるようにしたいところだ。パネルにランチャアイコンを登録する方法もあるが、ここではメニューエディタを使ってメニュー自体を整理する方法を紹介しよう。

### ■メニュー階層を浅くする

標準のメニューは、項目が多く、階層も深い。目的のメニュー項目を見つけるのも、ひと苦労だ。

表示されるメニュー項目を整理し、メニュー階層を浅くすることで、使い勝手が向上するはず。ただし、せっかく登録されているメニューアイテムを削除してしまうのは考えものである。あとで必要になるかもしれないからだ。

さらに言えば、メニュー項目を移動したり削除したりするのは、かなり大変な作業になる（メニューエディタの操作性も問題ありだ）。最も手軽で便利なのが「お気に入り」を利用した方法である。

まずメニューエディタを開いて（メインメニューの［プログラム］－［デスクトップ設定］－［メニューエディタ］）、［お気に入り（ユーザーメニュー）］を選択し、ツールバーの［新規項目］ボタンを押す。「無題」というメニュー項目が追加されるので、**画面11**のように設定しよう（各設定項目については、標準で登録済みのメニュー項目を参考にするとよい）。［保存］ボタンを押せば、メニュー項目が追加される。

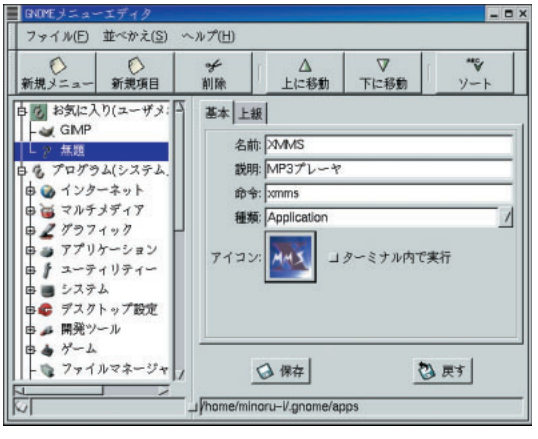
この手順を繰り返して、よく使うアプリケーションを「お気に入り」にすべて登録する。そして、GNOMEコントロールセンターの［デスクトップ］セクションにある［パネル］の［メニュー］タブ（**画面12**）で、「お気に入り」メニューの［メニューに］をチェックする。これで**画面13**に示すように、メインメニューにお気に入りのメニュー項目が表示されるようになる。


**画面13　設定変更後のメニュー**
追加するメニュー項目が多い場合は、［お気に入り］をサブメニューにするという手もアリ。

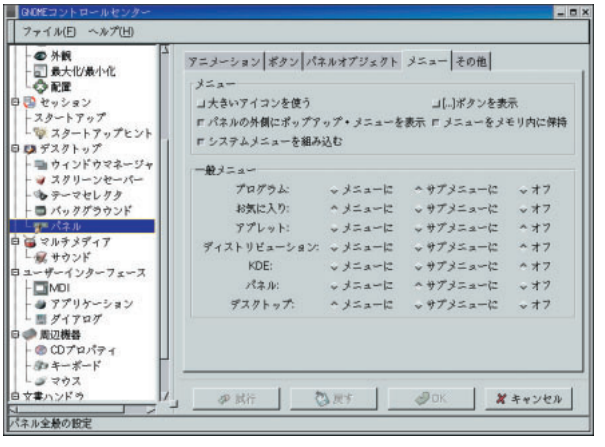
この方法であれば、既存のメニュー項目を削除することもなく、また通常のメニューを変更する場合と違い、一般ユーザーでも実行できる。標準のメニューが煩わしいと感じている人は、ぜひ一度試してみてほしい。

●

ディストリビューションで用意されているデフォルト環境は、GNOMEデスクトップのひとつの例にすぎない。コントロールセンターを駆使して自分に合った快適なデスクトップを構築しようではないか。




**画面11　メニュー項目の追加**
［説明］に設定したテキストはツールチップとして表示される。




**画面12　［メニュー］タブでの設定**
［システムメニューを組み込む］をオフにして表示されるメニュー項目を減らすことも可能。

知られざるホームコンピューティングの裏側

# パソコンの騒音問題を考える

パーソナルコンピュータ騒音対策委員会

# PCはウルサイ!

21世紀を迎えた現在、個人で充実したPC環境を整えるのはたやすい。いい時代になったものだと素直に喜びたいところではあるが、家庭でPCを使う際に無視できない問題が生じつつある。それは、PCの発する騒音問題だ。CPUやグラフィックチップ、チップセットの高機能化、高速化により、各パーツの発する熱量が増大した。そのため、PC内部を冷却するファンの数も増え、耐え難い騒音をまき散らすようになったのだ。

住宅事情の悪い日本において、部屋の中の騒音源は家庭不和の原因ともなりかねない。そこで今回は、静音化パーツやちょっと怪しげなグッズを秋葉原で購入し、その効果を検証してみた。とはいっても、音の静かさを測定して誌面で伝えるのは難しい。音の大きさだけではなく、音の種類や実際に聞く人の主観が混じるからだ。そのため、ちょっといい加減な気もするが、3名のレビュアーによる一口レビューも掲載することにしよう。

また、ソフトウェア面からのアプローチとして、1CD Linuxを紹介しよう。これは、CD-ROMからブートして利用するLinuxで、ハードディスクなしでも使えるものだ。PCルータを作る場合などに便利だろう。

## ターゲットにするマシン

静音化の対象にするマシンとして、**表1**に示したスペックのものを用意した。現在ではミドルクラスのマシンといったところだろうか。ケースは、本誌6月号の特集「4万円で作るLinuxマシン」で使用したマウス込みで5980円というものを流用した（**写真1**）。6月号では、低発熱CPUのVIA CyrixIIIを搭載していたが、今回はAthlon 1GHzを使うため、電源の容量が十分かどうかを確認する。この電源、“MAX 300W”などと書いてあるが、妙に軽いし、なんだかうさんくさい。スペックシートをよく見ると、実質150Wくらいの電源のようだ。そこで、Seventeamの300W電源を購入し、換装することにした（**写真2**）。計ってみたら、Seventeamのほうが700グラムも重い。こうして、ターゲットマシンは完成した。

いざ電源を投入してみると、見事にウルサイ。それだけではない。BIOS画面で見ると、CPUの温度がみるみる上がっていく。大丈夫だろうか。CPUに負荷がかかると、温度はさらに上昇するだろう。Linuxもインストールせずに、じっとBIOS画面を見ているわけにもいかないので、**写真3**の温度計を使ってCPUヒートシンクの温度を見ながら作業を進めることにしよう。小さすぎるケースは風通しが悪くなるのだ。気を付けよう。

| | |
|---|---|
| CPU | AMD Athlon 1GHz（FSB 266MHz） |
| マザーボード | ASUS A7M266 |
| メモリ | 256Mバイト（PC2100 DDR）×1 |
| 電源 | Seventeam ST-300BLV（300W） |
| ハードディスク | Seagate Barracuda ATA III 30Gバイト 7200rpm（ST330620A） |

表1　対象としたPCのスペック

写真1　4万円PCプロジェクトで購入したケース
高さはわずかに35cmと、ATX電源を内蔵するケースとしてはきわめて小さい。これがアダとなり、音だけではなく、熱の対策にもアタマを痛めることになる。

写真2　電源ユニット
左は5980円のケースに付属していた電源。右はソフマップにて6999円で購入したSeventeam ST-300BLVという300W電源。これで、取り出せる電力が倍増する。


写真3　FILCOの温度計
ケーブルの先にサーミスタが付いており、これをテープで貼り付けて使用する。-40℃〜100℃の範囲を0.1℃単位で測定可能。高速電脳で1980円。

# 騒音の元となるファン、ハードディスクを黙らせる！

PCを静かにしたいのはヤマヤマだが（そういう企画趣旨だし）、熱で壊れてしまっては元も子もない。騒音源となりうるデバイスを追加するのはツラいのだが、涙をのんでケース前面に冷却ファンを追加しよう。最近では、静かさをウリ文句にしたファンがちまたで話題になっている。**写真1**、**2**がそれだ。PAPSTのファンは、小さな金属片を使ってバランス調整をするという職人芸的な方法で静かさを向上させている。一方、松下のファンは、最近のハードディスクでも使われている流体軸受けを採用している。これは、特殊なオイルを封入した軸受けで、ボールベアリングに代わる最新技術だ。

## ケース前面に吸気ファンを取り付ける

まず、松下の2950rpm（標準？）のものを取り付ける。豪快に風が吹き込む。BIOS画面で見ると、回転数は3200rpmくらいだ。室温25℃の環境でCPUに負荷をかけ、CPUヒートシンクの温度を計る。するとどうだろう。ファンを付ける前にはあっという間に70℃を超えていたヒートシンクの温度が65℃くらいで安定した。しめしめといいたいところなのだが、非常にウルサイ。こんなマシンは決して自室に置きたくないくらいだ。ファンの回転音ではなく、ケース前面の隙間から吸気する際に発生する音が元凶のようだ。風量が多いとこういった音が大きくなるのだ。

次に松下の1900rpm（静音）ファンを試した。ケース前面からの音はかなり小さくなったが、CPUに負荷をかけるとやっぱり70℃をオーバーしてしまう。真夏の閉め切った部屋などでは室温がもっと高くなることを考えると少々不安だ。このCPUは、動作温度が90℃までとなっているが、できればヒートシンクの温度を70℃以下に抑えたい。BIOS画面で見ると、CPUコアの温度はヒートシンクよりも3〜5℃高く表示されている。

今度は、PAPSTの1500rpm（超静音）ファンに交換する。松下の1900rpmよりも回転数が低いのでちょっと心配だ。ところがビックリ、ヒートシンク温度が69℃で安定した。そのうえ、きわめて静かだ。ファンの回転音はほとんど聞こえず、吸気音もごくわずかにしか聞こえない。う〜む、ドイツのクラフトマンシップに脱帽。ケース前面の吸気ファンはPAPSTの超静音ファンに決定！

## 電源の静音化改造をする

やっとのことでスタートラインにたどり着いた。ここから静かPCへの道が始まる。騒音の元となるのは可動部分のあるデバイスだ。要するに、ファンとドライブということになる。これら

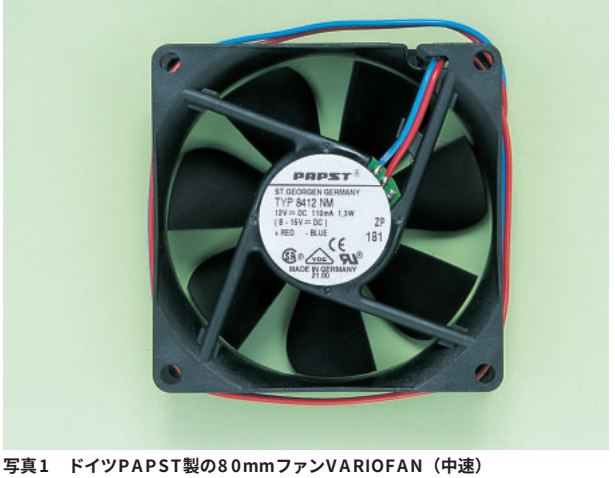



写真1　ドイツPAPST製の80mmファンVARIOFAN（中速）
回転ムラをなくすために、小さな錘でバランス取りを行っている。パルスセンサなしの2ピンタイプなので、BIOSで回転数を検知することはできない。今回は、1500rpm（超静音）、2600rpm（中速）を購入した。高速電脳で各2980円。


写真2　松下電器産業製の80mmファンPanaflo（2950rpm）
ボールベアリングよりも摩擦、騒音が少ないとされる流体軸受けを採用した冷却ファン。パルスセンサ内蔵の3ピンタイプ。今回は、1900rpm（静音）、2950rpmのものを購入した。高速電脳で各2800円。

を黙らせれば静かになるはずだ。このPCは、電源ユニットからウォーンとうなるような音がしているので、今度は電源ファンを交換しよう（**写真3**）。

電源ユニットを取り外し、元から付いていたファンを松下の2950rpmのものに交換する。BIOS画面によれば、元の電源ファンは1700rpmくらいで回っていたので、排気量とともに冷却効果のアップが期待できる。しかし、松下の2950rpmはケース前面に付けると非常にうるさかったファンなので、音が問題だ。覚悟を決め、電源を入れてぶったまげた。

静かだ。

電源ユニットからのうなりが見事に消えている。何がどうなっているのかとBIOS画面でファンの回転数を見ると、1670rpmだと表示される。どうやら、この電源は、ユニット内のファンに12Vではなく、6Vくらいの電圧を供給するらしい。冷却効果のアップはできなかったが、この静かさはすばらしい。元々付いていたファンがよほどうるさいものだったようだ。

PAPSTの2600rpmファンも試したところ、こちらも静かだが、松下とは違う種類の音がする。回転数がわからないことと、風量がちょっと少ないことから採用は見合わせる。電源ファンは松下の2950rpmに決定。

## ファンガードも見逃すな!

高級なケースや電源は風切り音の少ない針金タイプのファンガードを採用している。これはマネするしかない、早速改造だ。金ノコとヤスリでファンガードを切り取り、針金タイプを取り付ける（**写真4**）。前面のケースファン取り付け部分もギコギコと切る（**写真**

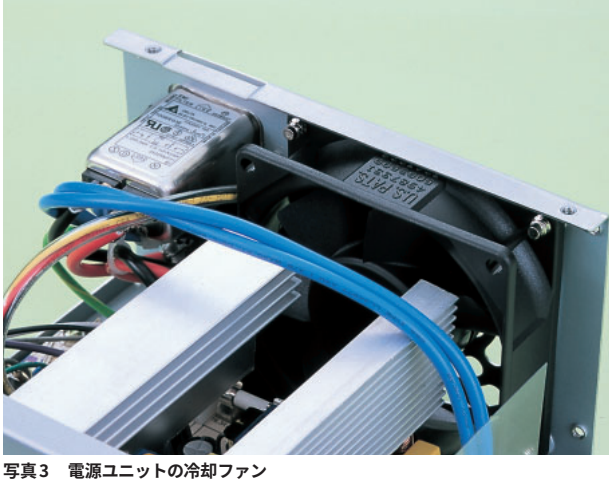


写真3　電源ユニットの冷却ファン
通常のATX電源には、8cm角のファンが付いている。電源のファンを交換する場合には、感電しないよう慎重に作業しないと危険だ。また、しっかり放熱できないと、火災の原因にもなるので、万人にお勧めできる改造ではない。

写真4　ファンガードを装着した電源ユニット
電源ファンの風切り音を軽減すべく、鉄板打ち出しのファンガードを金ノコで切り取り、針金タイプのファンガードを装着した（ソフマップで390円）。指を突っ込む可能性がなければ、ファンガードを付けなくてもよいかもしれない。

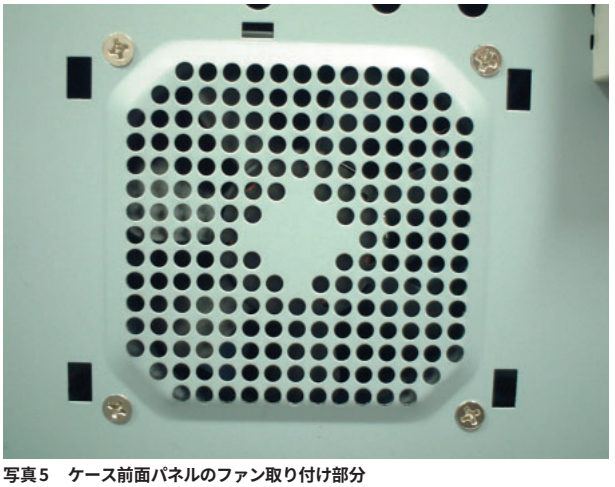


写真5　ケース前面パネルのファン取り付け部分
ケース前面の冷却ファン取り付け部分を見てみると、鉄板に小さな丸い穴がたくさんあけられている。この構造だと、風切り音が発生しやすく、吸気効率もよろしくないと思われる。どうせ前面パネルで隠れる部分だ、切り取ろう。

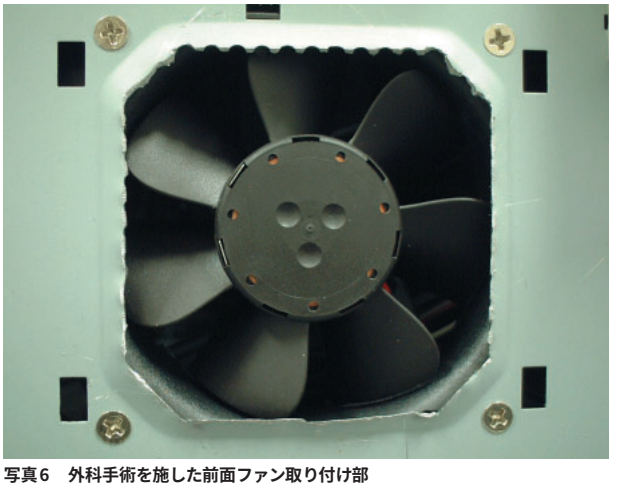


写真6　外科手術を施した前面ファン取り付け部
ラジオペンチ、金ノコ、ヤスリを駆使して切開した。切断面が汚いのは、我々の根性が不足しているため。みなさんが作業する際にはきれいに仕上げてほしい。写っているファンはPAPST製。中央左側にバランス取りの錘が見える。

5、6)。この結果、PCは一段と静かになったが、作業時には爆音がするので、夜間の実行はお勧めしない。

## ファンはすべて変えてみる

電源ファン交換に味を占め、残るファンも交換してみることにしよう。この時点でもっとも騒がしいのがCPUクーラーだ(**写真7**)。これには、60mm角、25mm厚のファンが使用されているので、同サイズの流体軸受けファンを用意した(**写真8**)。期待に胸を躍らせながら取り替えてみたが、ちっとも静かにならない。がっくり。回転数は元のファンよりわずかに速い程度(4300rpm→4500rpm)なので、その影響はわずかだろう。残念だが、元のファンに戻す。静音化には何ら関係しないのだが、松下のファンはトルクが非常に強かったことを申し添えておく。風量が多いのかもしれない。

残るファンはチップセットファンだけだ。今回使用したマザーボードには、DDR(Double Data Rate)メモリをサポートするAMD 761というノースブリッジが搭載されており、これが結構熱を持つのだ(**写真9**)。すぐ横にあるCPUファンが騒がしいこともあり、チップセットファンによる騒音はほとんど気にならないのだが、電源ファンの例もあるので念のために取り替えてみる。

残念ながら、PAPSTや松下電器産業の流体軸受けファンで40mm角のものを見つけられなかったので、ごく一般的なものを購入(**写真10**)。交換して電源を投入するとビビるような音がする。これは失敗だ。たまたまハズレの個体に当たっただけかもしれないが、元のファンよりもずっとうるさくなってしまった。半ばやけになって、ファ

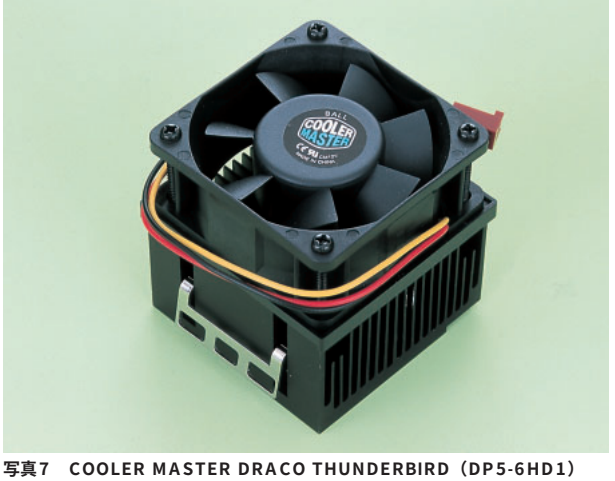



写真7　COOLER MASTER DRACO THUNDERBIRD(DP5-6HD1)
1.2GHzまでのAthlon、Pentium IIIに対応する。60mm角、25mmのファンでCPUを強力に冷却するが、その高さは65mmにもなるので、取り付けスペースがあるかどうかをよく調べよう。高速電脳にて3500円で購入。音はかなりうるさい。


写真8　松下電器産業製、Panafloの60mmファン
流体軸受けを採用し、回転数は4200rpm。CPUクーラーのファンを交換するために高速電脳で購入。パルスセンサ付きの3ピンタイプでBIOS画面などで回転数をモニターできる。流体軸受けの静音効果やいかに。価格は2500円。

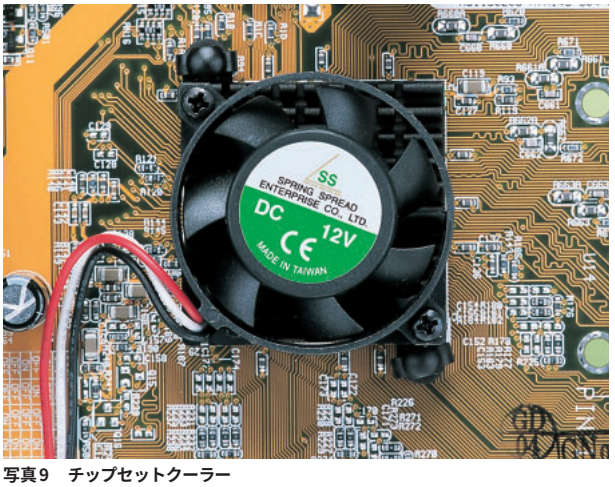



写真9　チップセットクーラー
DDR(Double Data Rate)メモリをサポートするノースブリッジのAMD 761には、40mm角のヒートシンクとファンが付いている。より放熱性能を高めるため、ヒートシンクとチップの間シリコングリスを塗布した。


写真10　Justyブランドの40mmファン
チップセットクーラーのファンを交換するために購入した。軸受けは、一般的なボールベアリング。元から付いているファンより静かになるかはわからない。回転数は5000rpmだ。T-ZONEで1580円だった。

ンを外してみると、ヒートシンクの温度がどんどん上昇していく。70℃の手前で恐くなり、電源を切る。結局は元のファンを取り付けることにする。

## 今度はハードディスク

ハードディスクの記録密度が高まったことで、データ転送速度も向上した。そのため、比較的低い回転速度でも十分実用になるようになり、静粛性と発熱の少なさを特徴としたドライブも市販されている。静かなPCを作るためには、このようなモデルを購入するという解決方法もあるが、今回はあえて7200rpmという高回転のドライブをチョイスしてみた。SeagateのBarracuda ATA IIIがそれだ。実際に動かしてみると、7200rpmの割に回転音が小さい。ヘッドをシークさせる際の「カカコ」という音のほうが回転音よりもずっと大きく聞こえる。

ハードディスクの音を軽減するアイテムとして、**写真11**のSMART DRIVEを試す。見た目はアルミの弁当箱みたいだ（もちろん、さまざまなノウハウに基づいて作られているのだろうが）。中には、吸音材が張ってありドライブをすっぽりと包み込む（**写真12**）。

SMART DRIVEを5インチベイに取り付けて音を聞いてみた。なんと！ドライブの回転音がまったくしなくなった。これはすごい。シーク音もごくわずかにしか聞こえない。PCから少し離れると聞き取れなくなるほどだ。「カカコ」と耳につく音だったのが、くぐもった感じになったのも静かに感じる原因になっているかもしれない。

## 回転系デバイス交換は利く!

以上のように、冷却ファンを静かなものに交換したり、ハードディスクを静音化すると、見違えるように（聞き違えるように？）PCがおとなしくなる。ファンガード回りの改造もそれなりに有効だ。ただし、静かだからといって、安易に高回転型のファンを低回転型に置き換えてはならない。冷却性能が低下するとさまざまなデバイスの寿命が短くなったり、火災の原因になるからだ。ケース内のエアーフローを総合的に見ながら静音化を進めてほしい。


写真11　ハードディスクの静音化アイテムSMART DRIVE
3.5インチのハードディスクを収納するアルミのケース。リボンケーブルと電源ケーブルは写真手前にあるシリコンゴムの隙間から外部に引き出す。ソフマップにて、6479円で購入した。ちょっと高い？

写真12　SMART DRIVEにハードディスクを入れたところ
SMART DRIVEの内部には、ハードディスクをピッタリと覆うように吸音材が張られている。フタを閉めたら、ケースの5インチベイに取り付ける。アルミでできているため、ドライブの放熱にも効果があるという。

### クロスレビュー

| | |
|---|---|
| **Tux Heaven** | 実験機の初期状態は壊れかけの加湿器のような凄い音がしていた。ところが前後のファンを変えていくとだんだん静かになり、さらにSMART DRIVEを利用することで、ほとんど気にならないレベルまで静音化でき、ちょっと感動した。ただ、1カ所改善すると、他の音源が気になったりするので、静音化するなら一気にやるのが正解だ。 |
| **のりぞう** | 最も騒がしいパーツから静音化すると効果的だ。まずは耳を駆使して騒音源を突き止めよう。SMART DRIVEも威力を発揮した。ハードディスクがうるさいと感じるなら、強い味方になるはずだ。このほか、ファンガード改造は、手間はかかるが費用がかからないのでぜひ試してほしい。ファン交換の際には冷却能力の低下に注意しよう。 |
| **にゃ〜** | あらためて聞くとPCは非常にうるさいということを再確認した。電源ファンの交換やファンガードの交換は効果的だが、個人的にはめんどくさいのでやだ。SMART DRIVEは、取り付けが簡単で見た目もイカしているのでグッド（フタをすると見えないのではあるが）。ただし、ドライブだけ静かになっても効果が低い点には要注意。 |

# 防音グッズをアレコレ試してみました

「防音グッズ」とは、いったい何ぞや？　と疑問に思われる向きもあろうかと思う。事実、ここで紹介する各種グッズも、今回検証してみるまでは、海のものとも山のものともしれぬ「あやしい」存在だったんである。

## ダイポルギー

のっけから、あやしさ炸裂で申し訳ない。「ダイポルギー」……なんというネーミングセンスだろうか。思わず「ポキ○ル」とか「ボラギノ○ル」などのシモ関連商品を連想しそうである。さて、ネーミングはともかく、その正体と効果を検証してみよう。

今回試したのは、「ダイポルギー制振シート」（**写真1**）と「ダイポルギー吸音フィルム」（**写真2**）の2つの製品である。パッケージの説明書きによると、ダイポルギーとは、

高分子化学と電気工学のテクノロジーを融合することによって開発された画期的な新素材

であるようだ。そして「騒音、振動、衝撃などのエネルギーを大幅に低減する」という特性があるらしい。これが本当であれば、かなりの期待が持てそうだ（でも何だかチトあやしい）。

### ■ ダイポルギー制振シート

まずは、筐体のマザーボード取り付け部分の裏側に制振シートを貼り付けてみることにした（**写真3**）。この部分はマザーボードから伝わる振動（CPUファンが主な原因と推測される）が顕著に感じられた部分だ。

「ものは試し」ということで、まったく期待していなかったのだが、これが意外にも効果を発揮したのであった。ケースに触ってみると、外側に伝わる振動が確実に減っているのだ（！）。これにより、騒音も取り付け前より抑えられている感じだ。

貼り付けただけで振動を抑えるとは！　すごいぞ、ダイポルギー！

### ■ ダイポルギー吸音フィルム

調子に乗ったところで、吸音フィルムも試してみた。こちらは、筐体のマザーボード取り付け側にある空間を塞ぐ形で両面テープを使って固定してみた（**写真4**）。

説明書きを信ずるなら、「膜振動により騒音を吸収する」ことで音の漏れを防げるはず。制振シートの威力からして、大いに期待が膨らんだのだったが……。

全然効かない

これ以上の解説は不要だ。


写真1　ダイポルギー制振シート
超エネルギー変換新素材「ダイポルギー」を採用した振動吸収グッズだ。大きさは150×190ミリで、厚さが0.8ミリ。見た目はボール紙のようだが、意外に柔らかい。加工しやすく、裏側がシールになっているので、いろいろな用途がありそうだ。開発・発売元はシーシーアイ、価格は1枚1280円。


写真2　ダイポルギー吸音フィルム
同じくダイポルギーを使用した防音グッズ。ちなみにダイポルギーは世界33カ国で特許出願中だそうだ。制振シートよりも薄く（0.2ミリ、大きさは同じ）、ゴムのような触感である。「膜振動により音を吸収する」という謳い文句だったのだが……。発売元と価格は制振シートと同じ。

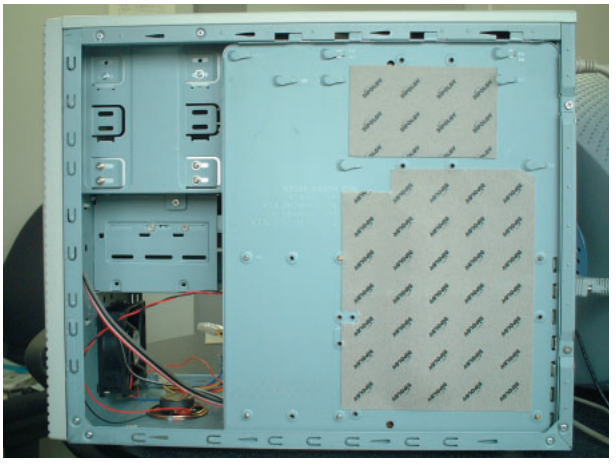


写真3　制振シートを筐体に貼付してみた
ケース全体の剛性にもよるのだろうが、マザーボードの裏側にあたる部分はかなり振動が大きい。実験当初は効果を期待していなかったこともあり、マザーのほぼ全面にくるよう豪快に貼り付けた。

写真4　吸音フィルムはケースのすき間に説明書きに「振動しやすい状態に」してほしいとあったので、このようにフィルムの端のみテープで固定した。残念ながら今回は効果を発揮しなかったが、取り付け方を変えればあるいは……。

## 振動吸収ネジ

次に登場するのは、パーツの取り付けに使うだけで「余分な振動を見事に吸収!!!」する（ビックリマーク3つは、説明書きの原文ママ）という触れ込みのネジである（**写真5**）。ネジと同じ「ABSORBER」シリーズのワッシャも同時に試してみることにした。

本来はハードディスクやマザーボードの取り付けに使うものなのだろうが、すでにハードディスクはSMART DRIVEで対策済みだし、マザーボードを取り付けるには本数が足らない（実はマザーボードを外すのが面倒だったという噂もある）ので、実験的に電源の取り付けに使ってみた。

結論から言うと、効果は不明である。ファンを取り替えてかなり静かになったとはいえ、電源は依然として振動しており、騒音も発生しているはずなのだが、はっきりとした効果は認められなかった。この段階までくると、最も耳につくCPUファンのものらしき騒音以外は（少なくとも我々の耳には）聞こえてこなかったのである。

残念ではあるが、評価は保留といったところか。

## 純鉛テープ

鉛の大きな比重（水：鉛＝1：11.6であるそうだ）と柔らかさにより、防振・防音効果がある、それが「防振・防音 純鉛テープ」であるらしい（**写真6**）。能書きとしては、先ほど紹介した「偉大なる」ダイポルギー制振シートに似ている。ただ、比重はともかくとして、「柔らかさ」がどのように効果を発揮するのかは、イマイチ不明であるといえよう。

とりあえず、ケースのサイドパネルの裏側に15cmほどの長さに切って貼り付けてみたが、効果はなし。比較のためにダイポルギー制振シートを同じ大きさに切って、同じ箇所に貼ってみたところ、こちらもはっきりした効果は認められなかった。

やはり純鉛テープの効果を引き出すには、最大の武器である「重さ」を生かすべきだろうということで、今度はパネルの裏側全面に貼り付けてみることにした（**写真7**）。

写真5　振動吸収ネジ＆ワッシャ
製品名は「ABSORBER SCREW」と「ABSORBER RING」。どちらも、「M2052」という特殊な素材から作られているそうである。それゆえか、1セット4つ入りで、SCREWが1800円、RINGが980円とかなり高価である。開発元はセイシン、販売元は親和産業。

写真6　防振・防音 純鉛テープ
「鉛色」という表現があるが、これはどうみても「銀色」である。純度の高い鉛はこういう色なのだろう。0.3ミリと薄く、ハサミやカッターで簡単に加工できる。また、裏面がシールになっていている。製造・販売元はワイドワーク、100×1000ミリで1400円。

写真7　ずっしりと貼り付けられた純鉛シール
全面に隙間なくびっしり貼り付けるという案もあったのだが、めんどくさかったので、こんな感じに。これでもかなりの重さである。フルタワーケースではどれくらいの重さになるのやら。

この状態でマシンを起動すると、確かに振動は軽減されているようだ。さらには、これまで微かに聞こえていた低音の「うなり」が低減されている。ダイポルギーほどのインパクトはないが、これはこれでかなり効果的かもしれない。

ちなみに、もう1つの特徴である「柔らかさ」については、振動を抑える効果はともかく、加工がしやすいという利点があるということを報告しておこう。

## V-Chip(ナゾ)

「V-Chip」という商品名からは、どのような製品かわからないだろう。実のところ、製品を見てもよくわからないのだ（**写真8**）。かろうじてV-Chipの効果を推測させるのは、その「プニョプニョ感」である。プニョプニョのゲルによって振動を吸収するのだと思われる。

製品セットの内容からいって、使用箇所はケースの「足」の部分以外にないとの結論に達したので（というか、それ以外には思いつかなかったのだが）、さっそく取り付けてみた（**写真9**）。ケースを軽く揺すると、ゲルのプニョプニョ感が手に伝わってきて変な感触だ。

マシンを起動すると、これまでよりさらに振動が減ったように感じられた。全体的に振動を抑えているようだが、特に机に伝わる振動が激減している。プニョプニョ・ゲルは確実に効いているのだ（すばらしい！）。

自宅で床にPCを直置きしている場合など、設置面への振動の伝わりを軽減できるので、設置場所によっては大いに役立つはず。

なお、よりプニョプニョ感の高い姉妹品「U-Chip」もある。両者の違いは負荷重量の制限。V-Chipは20キログラム、U-Chipは5キログラムまでの負荷に耐えられるらしい。PCの足として使う場合には、V-Chipのほうでないと制限重量を超えてしまう。試してみようかなという方は、購入時に十分注意するようにしてほしい。


写真8　振動吸収ゲルチップ
アルミ板に振動を吸収する特殊ゲルを貼り付けたもので、ゲルの柔らかさが違う「V-Chip」と「U-Chip」がある（写真はU-Chip）。どちらも、25ミリ四方で厚さ7ミリ、同じ大きさのアルミ板（厚さ2ミリ）と円形のウレタン（厚さ1ミリ）が同梱されていて、価格は1980円。発売元はJapanValue。

## 総まとめ

ファンやハードディスクケースも含めて、これまでの成果をもとに、電源、ケースなどの改造による騒音対策をテストマシンにすべて施してみた。特に効果の高かったダイポルギー制振シートを振動の大きいファンの取り付け部分に多用することに決定。写真10のようにネジ穴のある四隅に貼付した。

最終的な全改造点は以下のとおりである。

**・ケース／筐体**

Papstの超低音ファンを装着。ファンの吸気口の金網を切除し、筐体とファンの接触部分にダイポルギー制振シートを挟み込むように貼付。さらに、ケースの両サイドパネルに純鉛テープを、筐体のマザーボード裏の部分にダイポルギー制振シートを貼付し、とどめに

写真9　V-Chipで作った「足」
V-Chipに同梱のウレタンを貼り付けてケースの足替わりにしてみた。シールが付いているので取り付けは簡単。現在、これ以外の使い方を随時メールで受付中（ウソ）。

# パソコンの騒音問題を考える


写真10　CPUファンに取り付けた制振シート
ご覧のように接触面は少ないものの効果は「アリ」。完全に音がなくなるわけではないが、耳障りな雑音がカットされる。ケースファンと電源ファンにも同じ処置を施した。

写真11　ケーブルマウンター
ケーブルをまとめるバンドとシールで固定できる3センチ角ほどのプラスチック製の「マウンター」がセットになっている。マウンターの穴にバンドを通して使用する。6セット入りで価格は290円、発売元はAINEX。

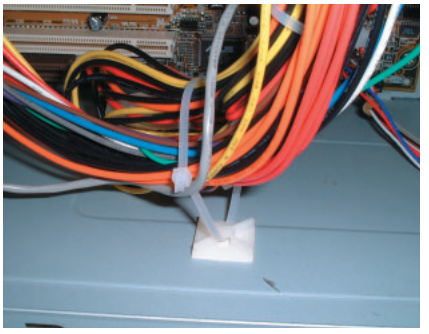


写真12　取りまとめた電源ケーブル
このようにバンドで取りまとめたケーブルをマウンターで固定する。ちょっとしたアイデアだが、なかなか効果的。バンドの長さが足らない場合は、2つのバンドをつなげるとよい。

足の部分をゲルチップに交換

**・電源**

ファンを松下電器の流体軸受けタイプに変更し、電源本体とファン、PC筐体との接触部分にダイポルギー制振シートを挟み込むように貼付。電源の取り付けには、振動吸収ネジを使用。また、排気口の金網を切除し、専用のファンガードに交換

**・CPUファン**

取り付け部分にダイポルギー制振シートを貼付

**・ハードディスクドライブ**

制音・冷却ケース「SMART DRIVE」に収納して取り付け

このほか、熱対策として**写真11**の「ケーブルマウンター」を使って、ケース内の電源ケーブルの取り回しを整理した（**写真12**）。

写真13　ケーブルを整理したケース内部
ケーブルをまとめることで空気の流れがよくなり、ファンの効果が高まる。実際に温度を計測したところ、負荷をかけた状態でヒートシンクの温度が64.9℃までに抑えられた。

| 製品 | 価格 |
|---|---|
| 8センチファン（ケース） | 2980円 |
| 8センチファン（電源） | 2800円 |
| ファンガード | 299円 |
| SMART DRIVE | 6479円 |
| 制振シート（×2） | 2560円 |
| 吸音フィルム | 1280円 |
| 振動吸収ネジ | 1800円 |
| 防振ワッシャー | 980円 |
| 純鉛シール（×2） | 2800円 |
| V-Chip | 1980円 |
| ケーブルマウンター | 290円 |
| 合計 | 2万4158円 |

表1　今回使用した製品の価格一覧

## ■ 絶大なる効果とそれなりの費用

かくして、低騒音PCは完成をみた。最初の状態からすると、驚くべき静かさである。ファンを取り替えるなどの音を小さくする対策も効果的だが、今回の実験から、振動を抑えることが重要であることがわかった。振動を抑えることで、耳障りな雑音が除去されるのだ。

費用の面にも目を向けてみよう。ここまでに紹介した改造に要した費用はいかほどなのか？　**表1**にその概算をまとめたので参照してほしい。

う〜ん。万全な騒音対策のためには、やはり相応の出費を覚悟しなければならないようである。予算と相談しながら、効果的な対策を目指していこう。

## クロスレビュー

| | |
|---|---|
| **Tux Heaven** | ダイポルギー、鉛板、防振ねじ、振動除去チップ。もう怪しさ爆発なアイテム群だ。しかし、実はこれらのアイテムがけっこう効果があるということがわかって驚いた。小さくて静かで、それでいてパワフル（1GHz）なマシンを作りたければ、これらのアイテムに加え、ケーブルを整理して風通しを良くするといった工夫が必要だ。 |
| **のりぞう** | 予想通り（！）、効果がわからないものもあった。そんな中で、ダイポルギー制振シートと、振動除去チップはしっかりとその力を発揮した。局所的で細かい振動を軽減するならダイポルギー、PC全体の振動が机や床に伝わらなくしたいなら振動除去チップを使おう。非常に高価な防振ネジの効果が確認できなかったのは残念である :-p。 |
| **にゃ〜** | 個人的には、振動除去チップを「大プッシュ」したい。皆さんにも、ぜひあのプニョプニョ感を味わってほしいものである。推測ではあるが地震対策にもなると思われる（たぶんウソ）。また、その効果が最も疑われていたダイポルギーが大健闘したのは見事だった。「振動を制する者は騒音を制す」という警句をもってまとめとしたい。 |

# 韓国からきた驚異のファンレス冷却システム

前のセクションまでで、本誌謹製サイレントPCは一応の完成を見た。その静かさはメーカー製PCと比べても遜色のないレベルだと自負している。ところが、韓国にはPCをファンレスで動作させるというすごいケースがあったのだ！　**写真1**がそのケース「CALM System」である。一見するとちょっと変なデザインのケースにしか見えないが、その内部には驚異の冷却システムが秘められている。内部をのぞくと、なにやら怪しげな部品が目につく。四角い金属部品から、サイドパネルに向けて2本の樹脂製チューブが延びているのがわかるだろう（**写真2**）。この部品には、エバポレータ（蒸発器）という名前が付けられており、CPUやグラフィックチップに取り付けてチップを冷却する役割を果たす。

チューブの行き先を確認すると、サイドパネルに固定された部品につながっている。形状はエバポレータに似ているが、ひとまわり大きい。これにはコンデンサ（復水器）という名前が付けられており、エバポレータから伝えられる熱をサイドパネルに逃がす（**写真3**）。エバポレータやコンデンサの中には、無色透明の液体が封じ込められている。

冷却システムの原理は次の通り。チップの発する熱でエバポレータが暖められると、その内部にある液体が蒸発する。液体が気体になる際には、気化熱を奪うのでエバポレータは冷却される。蒸発した気体はチューブを通って、コンデンサへ行き、そこで冷やされて液体となる。気体が液体になるときには凝結潜熱を放出し、コンデンサが暖められる。液体はチューブを通ってエバポレータへと戻る。この結果、チップで生じた熱は、エバポレータ→コンデンサというルートをたどってケース左側のサイドパネルへと運ばれる（**写真4〜6**）。ノートPC等で使われているヒートパイプと同じ原理だ

## その実力は?

開発元のCool & Silent Systemによると、現行のCALM Systemは、動作クロックが1GHz未満のCPUに対応するという。そこで、FSBを133MHzから100MHzに落とし、Athlonを750MHzで駆動してテストした。室温30℃の環境で、CPUをアイドル状態にすると、エバポレータの温度は52℃で安定した。このとき、BIOS画面でCPUの温度を見ると、73℃であった。正直にいって冷却性能はあまり高くない。しかし、すごいのは静粛性だ。ハードディスク以外は何も音を発しない。前述のSMART DRIVEを併用すると、PCが動作しているかどうかさえわからない（ホント）。

今回モデルにしたPCは、発熱量が大きいパーツを多用している。DDR


写真1　CALM System
韓国のCool & Silent Systemが開発した完全ファンレスのPCケース（http://www.cnssystem.com/）。日本で購入する場合は、送料込みで159.5USドル。支払い方法は、USドルの銀行振込のみ（振込手数料はなんと約5000円！）。

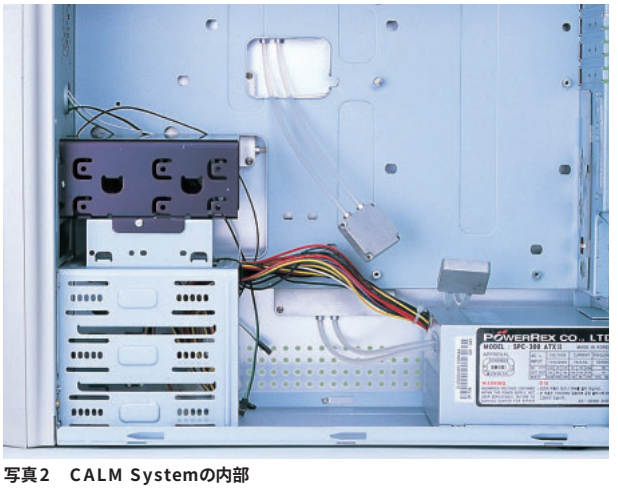



写真2　CALM Systemの内部
2本のチューブが付いているのがエバポレータ。これをCPUやグラフィックチップに固定する。電源内部にもエバポレータが取り付けられている。黒い3.5インチベイはアルミニウム製で放熱に効果を発揮するそうだ。

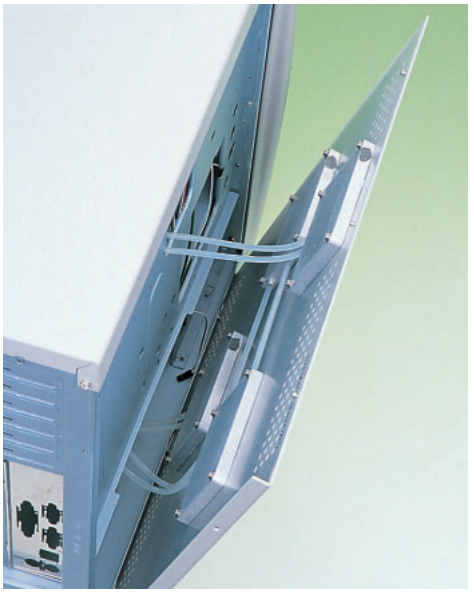


写真3　CALM System左側のサイドパネル
エバポレータからのチューブは、サイドパネルに固定された四角い金属部品につなげられている。これは、コンデンサと呼ばれるもので、アルミニウム製のサイドパネルに熱を逃がすためのものだ。

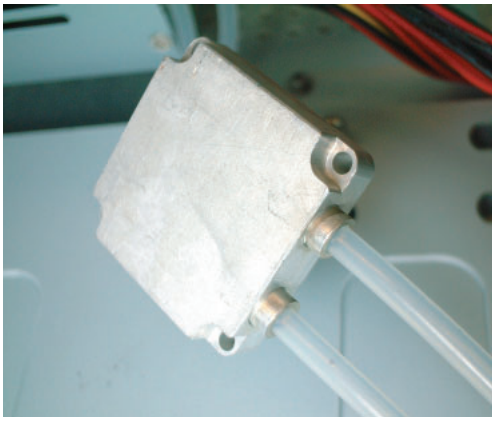


写真4　エバポレータ
チップや電源の熱はこのエバポレータに伝わり、中の液体を蒸発させる。その際の気化熱でチップなどの熱が奪われるのだ。生じた蒸気はチューブを通ってコンデンサに行き、そこで冷やされて液体になってエバポレータに戻る。

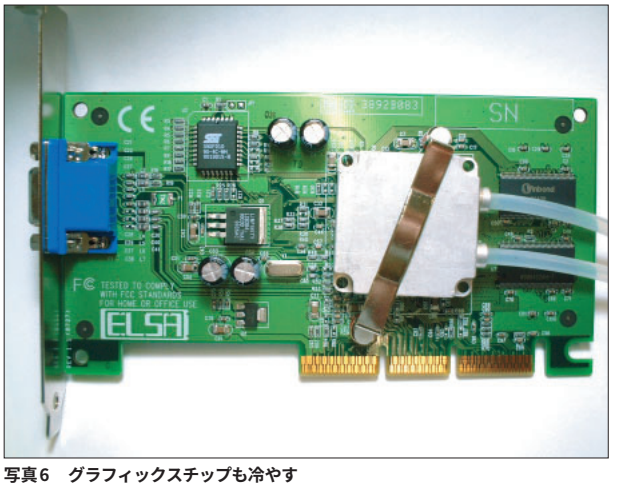



写真6　グラフィックスチップも冷やす
グラフィックスチップ冷却用のエバポレータはこのように装着する……のだが、チューブが届かず、カードをAGPスロットに挿せないことが発覚した。マザーボードやビデオカードの部品レイアウトにシビアなシステムだといえる。

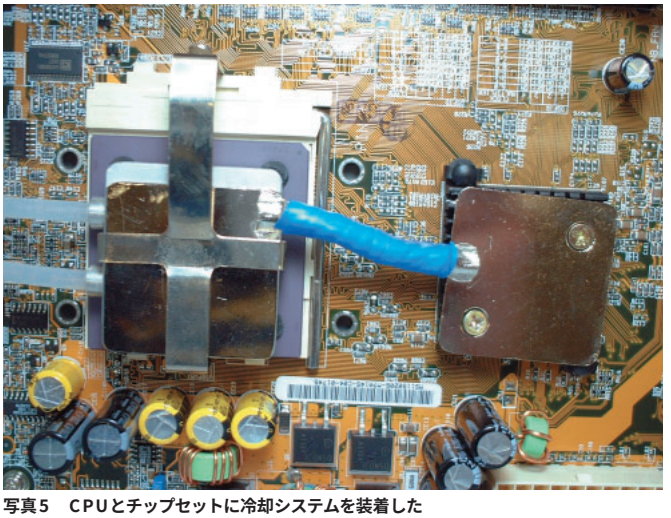


写真5　CPUとチップセットに冷却システムを装着した
CPUにエバポレータを装着（左側）。また、CPU上のエバポレータとチップセットのノースブリッジの間を金属板をつないだパーツでつなぎ、ノースブリッジの熱をエバポレータに伝える。マザーボードのレイアウトによっては使えない。

写真7　メモリクーラー
2枚のアルミ板でDIMMを挟むヒートシンク。メモリチップに接触する部分には熱伝導性のゲルシートが張られている。今回は、高速電脳で1480円で購入した。効果のほどはわからないが、音を発しない（当たり前？）ので試してみた。

SDRAMもかなり発熱するので、メモリクーラーを装着してみた（**写真7**）。さらに、システムの冷却性能を上げるため、ヒートシンクとして働く左側のサイドパネルを外部から扇風機で空冷したところ、30分後には10℃の温度低下をみた。サイドパネルに放熱フィンを取り付けるのもよいかもしれない。

同クロックのAthlonよりも発熱が少ないとされるPentium Ⅲなどで、動作クロックを500MHzくらいに抑えれば、ほとんど無音のPCを安定稼働させられるだろう。1GHz以上のCPUに対応するCALM Systemも開発しているようなので、今後に期待しよう。

マザーボードの部品レイアウトによってはチューブが届かないことがあるので、購入する際には注意が必要だ。

# 1CDで実現する静音ルータ

パソコンの騒音問題を考えるにつけ、音を出さないデバイスはないものかと思案していると意外に身近なところにあった。そう、メモリだ。

読者もご存じのとおり、メモリには駆動部分も振動部分もないので、常に無音だから、OSをメモリにインストールできれば、いままでハードディスクが出していた騒音から解放されるだろう。さいわい、LinuxにはRAMDISKといって、メモリの中にファイルシステムを作る機能があるので、Linuxをメモリにインストールすることは可能だ。

そこで、すでにルータとして稼動しているLinuxを利用して、メモリへインストールできる専用のLinux CD-Rを作ってみることにする。

## まずはファイルを丸ごとコピー

筆者のマシンは、Celeron 300MHzと256Mバイトのメモリを搭載しており、Red Hat Linux 7.1が最小構成でインストールされている。

このLinuxのパーティションの構成は、以下のとおりだ。

```
/dev/hda9       /
/dev/hda13      /cdroot
```

容量はどちらも2Gバイトで、/cdrootはCD-R作成用のパーティションだ。

/cdroot以下のファイルを1つのイメージファイルにしてCD-Rへ焼き付けるので、CD-Rの作成に必要なディスクの空き容量は、/cdrootの容量の2倍程度になる。まず、稼動中のLinuxのファイルを作業用の/cdrootにすべてコピーしよう。

```
# rsync -ax / /cdroot/
```

/cdroot以下の総容量が650Mバイト以下であればすべてのファイルがCD-Rに収まるので問題ないが、650Mバイトを超えた場合は、インストールされているドキュメントなどを削除して、/cdrootが650Mバイト以下になるようにシェイプアップしよう。

筆者の環境では、/cdrootの総容量は320Mバイト程度だったので、ファイルを削らずに、/cdroot以下のファイルをすべてCD-Rに焼くことにした。

```
$ su -
# mke2fs /dev/ram0
mke2fs 1.19, 13-Jul-2000 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
        …
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
# mount /dev/ram0 /mnt/
# touch /mnt/test
# ls -l /mnt/
合計 12
drwxr-xr-x    2 root     root        12288  6月 23 12:53 lost+found
-rw-r--r--    1 root     root            0  6月 23 12:53 test
# umount /mnt
```

画面1　RAMDISKの作り方

## RAMDISKとは?

CD-Rを作成する前に、Linuxをメモリにインストールする仕組みについて少し触れておこう。

まず、メモリの中にファイルシステム（RAMDISK）を作るといってもピンとこない人がいると思うので、実際にメモリの中にファイルを作成してみる（**画面1**）。/mntにマウントしたメモリ内のファイルシステムに、サイズゼロのファイルtestが作成されたのがわかるだろう。

さて、**画面1**の手順でRAMDISKを作成できない場合は、使用しているカーネルがRAMDISKをサポートしていない可能性がある。その場合は、カーネルの再構築が必要だが、最近のディストリビューションであれば、たいていRAMDISKはサポートされているはずだ。

## initrdとは?

RAMDISKにLinuxをインストールする際に使うinitrdというファイルについてもざっと説明しておこう。

まず、initrdの正体はどのようなものだろうか。たいていのディストリビューションにはinitrdのイメージファイ

ルを作成できるスクリプトが収録されているので、それを使ってinitrdファイルを作成してみよう。

```
# cd /tmp
# uname -r
2.4.2-2
# mkinitrd initrd.gz 2.4.2-2
```

mkinirdコマンドは、initrdファイルの名前と、「uname -r」で表示されるカーネルのバージョンを指定して実行する。

カレントディレクトリに、gzipで圧縮されたinitrdファイルinitrd.gzが作成されるので、

```
# gzip -dc initrd.gz > initrd
```

とgzipで伸長したあとに、

```
# losetup /dev/loop0 initrd
# mkdir /loop0
# mount /dev/loop0 /loop0
```

initrdをループバックデバイスとしてマウントし、

```
# ls -l /loop0
```

とすれば、initrdファイルの中身は、小さなLinuxファイルシステムで構成されていることがわかるだろう。

## Linuxのブートプロセス

initrdファイルの正体がわかったところで、initrdファイルがどのように使われるのかを、Linuxの起動プロセスに沿って見ていこう。

マシンを起動すると、まずLILOやGRUBといったブートローダが、Linuxカーネルとinitrdファイルをメモリに読み込み、直後に圧縮されたカーネルが展開される（**図1**）。次に、ブートローダに指定されたサイズのRAMDISKがメモリの中へ作成される（**図2**）。

このあと、RAMDISKの中にinitrdファイル内のLinuxファイルシステムが展開される（**図3**）。

さらに、initrdファイル内のファイルシステムがルートファイルシステムとしてマウントされ、initrdファイルを保存していた領域が解放される（**図4**）。

最後に、initrdファイル内のlinuxrcというスクリプトが実行されて、そのあとにハードディスク内のファイルシステムがルートにマウントされるのが一般的だ。

initrdファイルにはLinuxの起動に必要な最小限のファイルしか含まれていないので、Linuxの運用に必要なファイルは、CD-Rに焼いたものをRAMDISKにコピーして使うことにする（**図5**）。また、CD-Rに含まれるファイルをRAMDISKコピーしきれない場合は、CD-Rにリンクを張って対処する。

## CD-R作成に必要なツールの確認

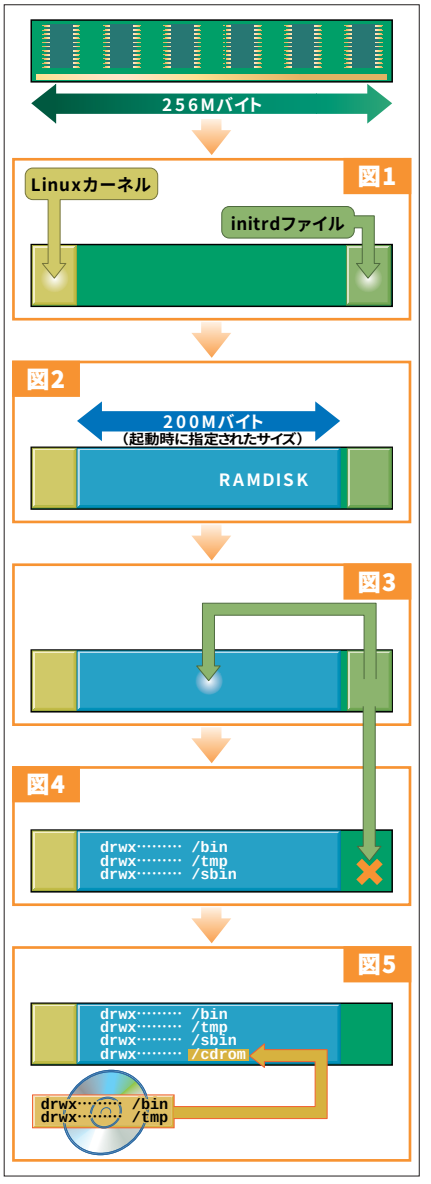
起動可能なCD-Rの作成にはisolinuxとmkisofsという2つのツールが必要だ。

isolinuxはCD-ROM用のブートローダで、syslinux 1.60以降に収録されている。また、mkisofsは、ISO9660形式のイメージファイルを作成するためのツールで、ほとんどのディストリビューションに収録されている。

```
# rpm -q syslinux
# mkisofs --version
```

とすればインストールされているツールのバージョンが表示されるので、syslinuxのバージョンが1.5.x以前のものだったり、mkisofsのバージョンが1.12以前のものである場合は、**表1**のURLからダウンロードしてインストールしよう。

このほかにも、RAMDISK内のLinuxがCD-Rからファイルをコピーできるように、基本的なコマンド（bash、rsync、ln、mount、umount）を、ライブラリのない環境でも使えるようにコンパイルしておく。必要なソースを本誌付録のCD-ROMに収録しておいた



ので、以下のようにコンパイルしよう。

```
# mount /mnt/cdrom
# cd /mnt/cdrom
# cp ./Linuxmag/SilentPC/* /tmp/
# sh /tmp/make.sh
```

コンパイルされたコマンド群は、/tmp/binary以下に格納される。

## initrdファイルの作成

それでは、CD Linuxの作成のキモとなるinitrdファイルを作成しよう。

### ■initrdの基本部分を作る

まずは、initrdファイルの容器となる空のイメージファイルを作成する。

```
# cd /tmp
# dd if=/dev/zero of=image bs=1024k
count=192
```

この例では、「1024kバイト×192ブロック」で、192Mバイトのイメージファイルimageが/tmpに作成される。

imageのサイズは、搭載しているメモリの7〜8割程度のサイズにしておこう。このあとは、imageをループバックデバイスとして使えるようにセットして、/dev/loop1にマウントされたimageをext2でフォーマットして、imageにアクセスできるようにマウントする。

```
# losetup /dev/loop1 image
# mke2fs /dev/loop1
# mkdir /loop1
```

▼リスト1　fstabの編集

```
/dev/ram0       /           ext2    defaults        1 1
none            /proc       proc    defaults        0 0
none            /dev/pts    devpts  gid=5,mode=620  0 0
```

```
# mount /dev/loop1 /loop1
```

あとは、

```
# cp -a /loop0/* /loop1/
```

initrd内のファイルをimageの中へコピーすれば、読者オリジナルinitrdファイルの基本部分は完成だ。

### ■initrdファイルのカスタマイズ

まずは、コンパイルした基本ツール群と、RAMDISK内のLinuxがCD-ROMドライブをマウントできるように、ドライバモジュールをコピーする。

```
# mkdir -p /loop1/tmp
# cp /tmp/binary/* /loop1/tmp/
# cd /lib/modules/2.4.2-2/kernel/
drivers/
# cp cdrom/cdrom.o /loop1/tmp/
# cp ide/ide-cd.o /loop1/tmp/
```

さらに、メモリ内のLinuxに必要となる最小限のデバイスファイルを作成しておく。

```
# cp -a /dev/cdrom /loop1/dev
# cp -a /dev/ram0 /loop1/dev
# cp -a /dev/hdc /loop1/dev
```

最後の「/dev/hdc」は、実際にCD-ROMドライブが接続されている場所だ。

Linuxのファイルシステムに必要なディレクトリも作成しておく。

```
# cd /loop1/
# mkdir cdrom var usr mnt proc root
opt
# rm -rf /loop1/lib
```

さて、CD Linuxは、RAMDISKを「/」としてマウントするので、

```
# cp /etc/fstab /loop1/etc/
```

とfstabをコピーしたあとに、**リスト1**のようにRAMDISKが「/」としてマウントされるように編集する。

最後に、initrdがRAMDISKに読み込まれた直後に実行されるスクリプトlinuxrcを**リスト2**のように編集して、実行属性をつけておけば、initrdファイルのカスタマイズは終わりだ。なお、linuxrcは（http://linux.ascii24.com/linux/linuxmag/update/）からダウンロードできる。

ループバックマウントしたイメージファイルを解除したあとに、gzipで圧縮しておこう。

```
# chmod +x /loop1/linuxrc
# cd /tmp/
# umount /loop0
# umount /loop1
# losetup -d /dev/loop0
# losetup -d /dev/loop1
# gzip -c9 image > image.gz
```

| ツール名 | URL |
|---|---|
| isolinux | http://www.kernel.org/pub/linux/utils/boot/syslinux/RPMS/ |
| mkisofs | http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html |

表1　CD Linuxの作成に必要なツール
isolinuxはsyslinuxというパッケージに収録される。mkisofsは、mkisofsかcdrecordというパッケージに収録されている。

## CDイメージファイルの作成

まず、syslinux 1.60以降のパッケージに含まれているisolinux.binというブートローダを、isolinux用のディレクトリにコピーする。

```
# mkdir -p /cdroot/isolinux
# cp /usr/lib/syslinux/isolinux.bin
/cdroot/isolinux/
```

次に、/cdroot/isolinux/の中に、カーネルとinitrdファイルをコピーする。

```
# cp /boot/vmlinuz-2.4.2-2 /cdroot/
isolinux/
# cp /tmp/image.gz /cdroot/isolinux/
```

あとは、/cdroot/isolinux/に設定ファイルisolinux.cfg（**リスト3**）を作成すればよい。

最後にmkisofsコマンドでISO9660形式のイメージファイルを作成する（**画面2**）。作成されたcdrlinux.isoファイルをCD-Rに焼き付ければ、CD Linuxは完成だ。

## 更新されたファイルはどうする?

さて、今回作成したCD Linuxは、更新されるファイルのことをまったく考えていないが、産業技術総合研究所の須崎氏が、更新されるファイルも管理できるシステム構築例「Full RAM DISK Linux」を公開されているので、興味のある読者は須崎氏のWebページを参考にしていただきたい（**表2**）。

**▼リスト2　RAMDISK Linuxの起動スクリプトlinuxrc**

```
#!/tmp/bash
echo "+ Starting linuxrc....."

/bin/insmod /tmp/cdrom.o && /bin/insmod /tmp/ide-cd.o &&
echo "+ Modules for CDROM are installed."

/tmp/mount -t proc proc /proc && echo "+ proc filesystem is mounted."

echo 0x100 > /proc/sys/kernel/real-root-dev &&
echo "+ /dev/ram0 is mounted as root filesystem."

/tmp/mount -t iso9660 /dev/cdrom /cdrom  &&
echo "+ /dev/cdrom is mounted as /cdrom."

/tmp/ln -fs /cdrom/lib /  &&
/tmp/ln -fs /cdrom/usr/lib /usr/  &&
/tmp/ln -fs /cdrom/usr/share  /usr  &&
/tmp/rsync -avx /cdrom/etc  --exclude "fstab"  /  &&
/tmp/rsync -avx /cdrom/sbin /  && /tmp/rsync -avx /cdrom/bin    /  &&
/tmp/rsync -avx /cdrom/var  /  && /tmp/rsync -avx /cdrom/dev    /  &&
/tmp/rsync -avx /cdrom/home /  && /tmp/rsync -avx /cdrom/mnt    /  &&
/tmp/rsync -avx /cdrom/boot /  && /tmp/rsync -avx /cdrom/opt    /  &&
/tmp/rsync -avx /cdrom/root /  && /tmp/rsync -avx /cdrom/tmp    /  &&
echo "+ Files are synced to RAM DISK."

echo "+ linuxrc is ended."
```

**▼リスト3　isolinuxの設定ファイルisolinux.cfg**

```
serial 0 9600
prompt 1
default multi
label single
        kernel vmlinuz-2.4.2-2
        append initrd=image.gz ramdisk=200000 single
label multi
        kernel vmlinuz-2.4.2-2
        append initrd=image.gz ramdisk=200000
```

カーネルのファイル名
圧縮前のinitrdファイルimageよりも大きな値にする
initrdのファイル名

```
# cd /cdroot
# mkisofs -R -J -no-emul-boot -boot-load-size 4 -boot-info-table -b isolinux/isolinux.bin -c isolinux/
boot.cat -o /tmp/cdlinux.iso /cdroot
```

CDのルートディレクトリ
ブートローダ
CDのイメージファイル名

画面2　ISO9660形式のイメージファイルを作成

| | | |
|---|---|---|
| Full RAM DISK Linux | http://www.etl.go.jp/~suzaki/RAMDISK_Linux/index.html | 更新されたファイルの管理にも対応したRAMDISK Linux |
| initrd.txt | linux/Documentation/initrd.txt | LILOの作者によって書かれたinitrdの解説 |
| ramdisk.txt | linux/Documentation/ramdisk.txt | カーネルパラメータについて詳しく書かれたRAMDISKの解説 |
| The Linux Bootdisk HOWTO | http://www.linuxdoc.org/HOWTO/Bootdisk-HOWTO/index.html | CDを含めたブートディスクの作成方法の詳細な解説 |
| SuperRescue CD | http://www.kernel.org/pub/dist/superrescue/v2/ | isolinuxの作者によって作られたレスキュー用のCD Linux |
| Gibraltar | http://gibraltar.vianova.at/ | Debianをベースとしたルータ用のCD Linux |
| CD-Based Linux Distros | http://plug.twuug.org/articles/cddistro.html | CD Linuxが数多く紹介されているページ |

表2　CD Linuxの作成に参考になる文書やプロジェクト
ゼロからCD Linuxを作るのはちょっと敷居が高いという読者は、完成した状態で配布されているSuperRescue CDやGibraltarをおすすめする。

新世代64ビットCPUに迫れ

# Itaniumそこが知りたい

## Pentiumとはここが違う！　キーワードを元にItaniumを理解する


文：Linux magazine編集部
*Text : Linux magazine*
*Photo : Junko Kitade (Pacia)*

Itanium

# Itaniumの性能を分析する

Ready

Itaniumの正式デビューから1カ月がたち、インターネット上の情報サイトに、Itaniumに関する記事、レビューなどが多く見られるようになってきた。

以前は発表できなかった、性能の評価結果も見受けられる。それらを見ると、

「たいしたことない」

という意見と、

「すごく速い」

という両極端の意見があるように思われる。なぜこのように評価が分かれるのか。本当のところは、どちらなのだろうか。

現状では、Itanium搭載機は非常に高価（最低でも200万円前後）であり、「ちょっと試してみたい」レベルのユーザーが買えるものではない。また、x86 PCのように、秋葉原のショップでCPUやマザーボードが単品で売られることもない（少なくとも6月半ばの段階では、そのような話は聞かれていない）。そのため大多数のユーザーは、自分でItaniumの性能を確認することができないまま、上記のように相反する評価結果を見ることになる。

## 遅い理由

最新のCPUなのに「遅い」という評価になってしまう理由は、

- **32ビットコードエミュレーション**
- **コンパイラの性能**

の2点だ。

### ■32ビットコードエミュレーション

Itaniumは、既存のx86とはまったく異なったコード体系のCPUではあるが、x86の32ビットコードも動作させることができる。i386以降Pentium 4まで、バイナリ互換性を保ちながら性能を上げ続けたことで、x86市場に君臨してきたインテルなら、当然の判断だろう。

ただx86コードのItanium上での実行は、エミュレーションであり、フルに性能が発揮されるわけではない。x86コードのプログラムなら、同じ時期に製造されている最新のx86 CPUで動作させるほうが速いのだ。

またItaniumのx86エミュレーションは、それほど効率が高いものではないらしい。一説によるとソフトウェアでx86コードのエミュレーションを行っているTransmeta社のCrusoeよりも、効率が低いという調査結果もある。

Itaniumでx86コードを実行するのは、x86 PC用の拡張カード上のBIOSを動作させる場合など、ごく限られた状況だけと考えるべきであろう。

### ■コンパイラ

Itaniumが採用しているEPIC（Explicitly Parallel Instruction Computing）アーキテクチャは、コンパイル時に同時実行可能な命令群を抽出することで、1クロックあたりの実行命令数を高めて性能を上げている。

RISC（Reduced Instruction Set Computer）で用いられることの多い「スーパースカラー」が、命令実行時に同時実行可能な命令を選び出すのとは対照的だ。ハードウェアで行っていた処理をソフトウェア（コンパイラ）に行わせるのがEPICだと思えばよい。

このためItaniumは、RISC CPU以上に、コンパイラの能力によって性能が上下する。いかに多くの同時実行可能な組み合わせを抽出できるかがポイントだ。

Itanium用のコンパイラは、インテルからC++、Fortranコンパイラが販売されているほか、SGIからもC、C++、Fortran90コンパイラがGPLに基づいて提供されている。

またGCC（GNU Compiler Collection）もItaniumに対応したコードを出力できるが、現状ではEPICに合わせた最適化が進んでおらず、他のコンパイラよりも性能の低いバイナリになってしまうようだ。今後は、インテルやSGIから情報が提供されることによって、GCCの性能も上がってくることが期待される。

## 速い理由

Itaniumがその性能を発揮できるのは、上述の条件と反対の場合、すなわち高性能なコンパイラを用いて作成した、ネイティブコードを実行したときだ。

**グラフ1**は、セキュリティを考慮したWebトランザクションを1秒間にいくつ実行できるかを比較した結果である。これは、インターネット上の買物サイトでクレジットカード番号などを扱うのと同様の処理であり、SSL

（Secure Socket Layer）を用いて暗号化を行っている。

Itaniumは、サン・マイクロシステムズのUltraSPARC IIと比較すると、クロック比を大幅に上回るほどの性能差を示す。SSLは負荷の高い処理であり、必要に応じてハードウェアアクセラレータが用いられるが、Itaniumのソフトウェア処理による結果は、アクセラレータを組み合わせたUltraSPARC IIもさらに上回っている。

SSLの処理には、掛け算と足し算を繰り返す処理（積和演算）が大量に含まれているが、Itaniumはこの処理を1命令で行えるため、これほどまでの大差がついたと思われる。一般にSSLのハードウェアアクセラレータは、非常に高価なため、この用途ではItaniumマシンの競争力は非常に高いといえるだろう。

## 全般的な処理速度の比較

SSLの処理は極端な例だが、全般的にItaniumは浮動小数点演算に強いCPUであると言える。

**グラフ2**に、SPEC（Standard Performance Evaluation Corporation）のSPEC CPU2000というベンチマークの結果を示した。SPECは、ベンチマークの標準化を目指した非営利団体である（**画面1**）。SPECint2000は整数演算、SPECfp2000は浮動小数点演算の指標である。

整数演算ではUltraSPARC IIIに遅れをとるが、浮動小数点演算では大差をつけていることがわかるだろう。

## まず技術計算分野に普及か

Itaniumは、大企業の基幹部門でデータベースアプリケーションを動かすようなビジネス用途が中心と考えられていたが、浮動小数点演算性能の高さから大学や研究所などの科学技術計算分野でも需要が高まっているようだ。

この分野のユーザーは、性能が高ければ特にOSやアーキテクチャにこだわらない性向があるため、まったく新しいCPUでも抵抗なく受け入れられている。また、必要なプログラムはユーザー自身が作成するため、新規プラットフォームにありがちなアプリケーション不足も問題にはならない。

Itaniumネイティブのアプリケーションが揃うのを待って、徐々に導入されているビジネス分野より先に、科学技術計算の分野でItaniumが普及していくと思われる。

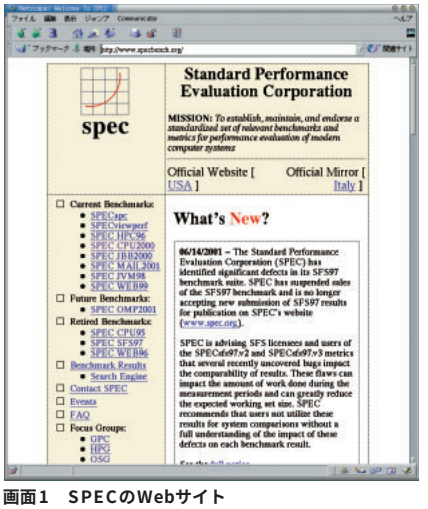
## Alpha技術の統合

締め切り過ぎの6月25日、Itaniumにも影響のある大きなニュースが飛び込んできた。コンパックが、Alphaプロセッサアーキテクチャのプロセッサ／コンパイラ技術をインテルに移管し、自社の64ビットプラットフォームを、2004年までにItaniumに統一することを発表したのだ。

Alphaは元DECが開発した高性能64ビットCPUで、DECがコンパックに買収されたため、コンパックの所有となっていた。

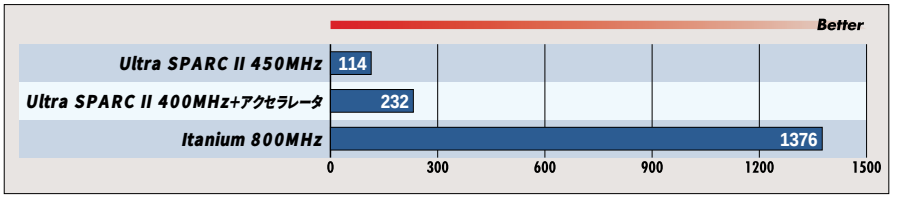
コンパックにとって、Alphaとx86／Itaniumという複数のプラットフォームを発展させ続けるのは、同社にとって負担が大きかったのであろう。

インテルにとっては、64ビットCPU市場でのライバルが減ったのと同時に、優秀な技術者を雇用できる大きなチャンスだ。Itaniumの後継機種開発もさらに加速することであろう。

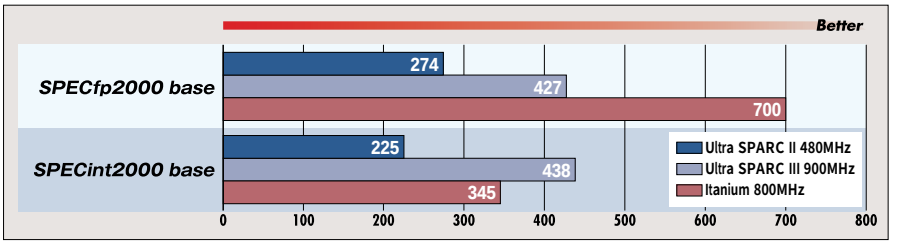



画面1　SPECのWebサイト
http://www.specbench.org/

グラフ1　セキュリティ保護されたWebトランザクション数（1秒あたり）




グラフ2　SPECint2000／SPECfp2000



# Itaniumを理解するためのキーワード

「Itaniumは64ビットCPUだ」という言葉を聞くけど、本当のところ32ビットのx86よりどれくらい優れているのか？ 64÷32＝2だから2倍？（違います）

今さら人に聞けないItaniumに関する基礎知識を、キーワードを元に押さえておこう。

## IA-64

インテルとヒューレットパッカード（HP）が共同で開発した64ビットCPUのアーキテクチャのこと。何をもってCPUのビット数とするかはいつも議論になる点だが、IA-64は128本の64ビット整数レジスタと、64ビットのアドレス空間を持った本物の64ビットCPUである。

32ビットCPUがフラットにアクセスできるアドレス空間は、4Gバイト（2の32乗）だが、64ビットでは約180億Gバイト（2の64乗）となり、当分は無制限とみなしてよい広大なアドレス空間をフラットに利用できる。

また、64ビットの整数レジスタと82ビットの倍精度浮動小数点レジスタを持ち、大きな数字や高精度のデータを扱う際にも、特別な操作は必要ない。

## Itanium

IA-64に基づいた最初の製品である。今年の5月29日に正式リリースされた。「アイテニアム」（テにアクセント）と読まなければならない。カートリッジ型の形状をしており、内部に1〜3次キャッシュまでを統合している。

733MHz／2Mバイト 3次キャッシュ
800MHz／4Mバイト 3次キャッシュ

の2グレードがある。

## McKinley（マッキンリー）

次世代のItaniumのコードネーム。今年末にサンプル出荷の予定。インテルのロードマップを見ると、マッキンリーの次世代として、

Madison（マディソン）
Deerfield（ディアフィールド）

というコードネームが上がっている。これら以外にも未発表のIA-64系プロセッサが開発中であると言われている。

## VLIW

CPUを高速化するには、次の2通りのアプローチがある。

**（1）命令の処理時間を短くする**
**（2）多くの命令を同時に処理する**

（1）は、クロックを上げることである。現状ではCPUのクロック上昇にメモリなど周辺装置の速度が追いつかないため、CPUが待たされることになり、クロックを上げても実効性能はそれほど上がらなくなっている。

そこで高クロック化と同時に、（2）の複数命令の同時処理（並列処理）が一般的になっている。

VLIW（ Very Long Instruction

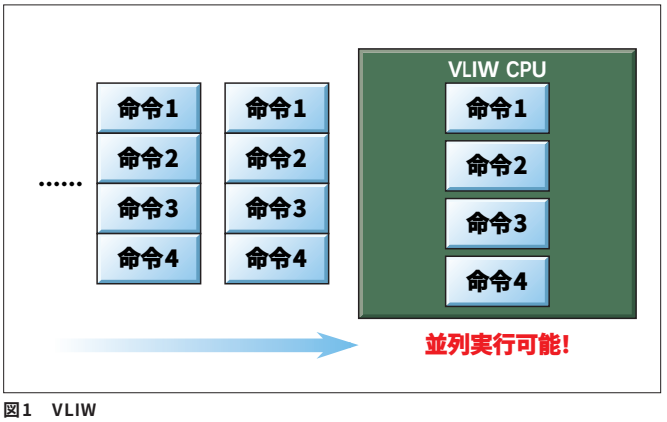



図1　VLIW
ひとまとめにされた命令群は、同時実行可能だ。

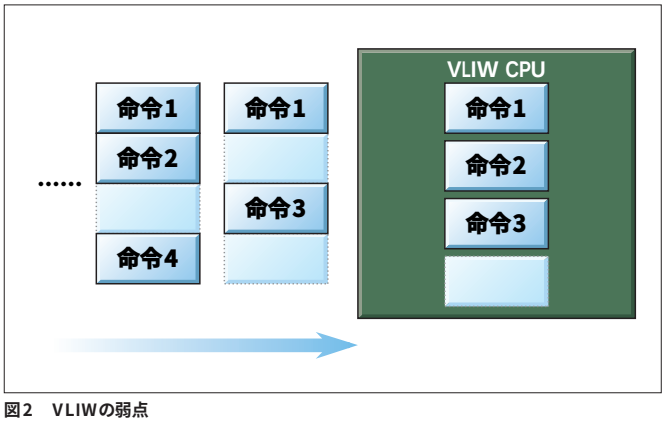



図2　VLIWの弱点
コンパイラの性能が低いと、性能が上がらないうえに、プログラムサイズが大きくなってしまう。

Word）は、コンパイル時に並列処理可能な命令群を抽出し、とても長い（Very Long）命令にまとめるという手法だ。まとめられた命令群は、並列実行可能であることがわかっているので、CPUは個別の命令に分解した後、命令実行ユニットに送ることで並列処理が行える。

VLIWは、あらかじめ並列実行可能な命令群を処理するので、CPUのハードウェアは単純なものになる（**図1**）。

しかし一方で、コンパイラの性能が低いと、**図2**のように命令で満たされていない長い命令が読み込まれることになる。その場合、空いたところには、「何もしない」命令が書き込まれるため、性能が上がらないばかりか、プログラムのサイズが不必要に大きくなってしまうことになる。

またハードウェアが進歩して、同時に実行できる命令数が増えた場合、コードの互換性がなくなってしまうという問題点もある。

VLIWは以前から研究されていた手法ではあるが、上述の問題点があるため、今までVLIW方式のCPUが主流になることはなかった。

## EPIC

IA-64で採用されているアーキテクチャで、Explicitly Parallel Instruction Computingの略であり、直訳すれば「明示的並列命令コンピューティング」となる。VLIWをベースに、その欠点を克服するための工夫が施されている。

IA-64の命令は、6種類のタイプに分類されており、それぞれ処理される実行ユニットが決められている（**表1**）。たとえば命令タイプAに分類される、単純な加減算の命令は、IユニットまたはMユニットで実行可能だ。

各命令は、個別に扱われるのではなく、3個ずつ「バンドル」と呼ばれる「長い命令」にまとめられている（**図3**）。命令の長さは41ビットであるのに対し、バンドルは128ビットの長さを持つ。残りの5ビットは「テンプレート」と呼ばれ、バンドルに含まれている3つの命令の、

**・実行ユニット**

**・同時実行の可否**

が記述されている。図4のバンドルを例にとると、命令0はMユニット、命令1、2はIユニットで処理すればよいことがわかる。また命令2の後ろに二重線が描かれているが、これは後続のバンドル内の命令とは同時実行ができないことを表している。この同時実行が可能かどうか示す境界は、「ストップ」と呼ばれる。

このようにEPICでは、並列実行できる命令群を「長い命令」の内部に限らず、ストップで「明示」することにしたため、VLIWのような不必要なコードの膨張や、CPU世代間の非互換といった問題を解決している。また、各命令をどの実行ユニットで処理するかは、テンプレートを見るだけで簡単に判別できるため、命令の解釈（デコード）も短時間で済むようになっている。

| 命令タイプ | 説明 | 実行ユニットのタイプ |
|---|---|---|
| A | 比較的簡単な整数演算 | IユニットまたはMユニット |
| I | 整数演算 | Iユニット |
| M | メモリ処理 | Mユニット |
| F | 浮動小数点演算 | Fユニット |
| B | 分岐処理 | Bユニット |
| L+X | 拡張演算 | IユニットまたはBユニット |

表1　命令タイプと実行ユニットの関係

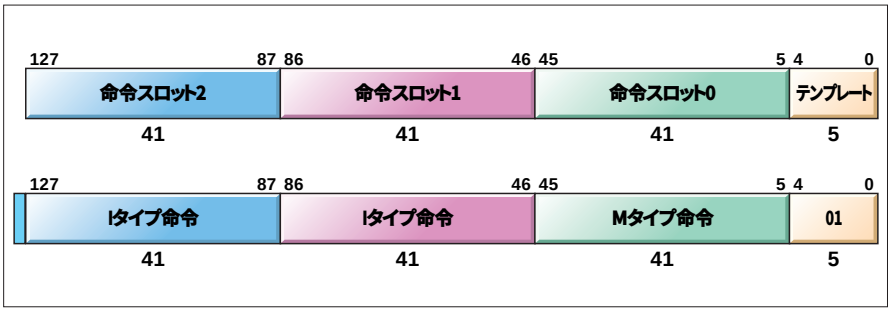



図3　バンドル
128ビットのバンドル内には、3個の命令と、テンプレートが含まれる。

## スーパースカラー

CPUは高速化のために、命令の実行を複数のステップに分割して行っている。このステップ全体のことを、「パイプライン」という。多くのRISC（Reduced Instruction Set Computer）CPUやPentium以降のx86 CPUでは、このパイプラインを複数持つことで、1クロックサイクルに複数の命令を実行する。この方式をスーパースカラーという。

スーパースカラーは、並列処理可能な命令群をプログラム実行時に抽出するが、そのための回路が加わることで、CPUが大きく複雑になるという問題が生じる。また、そのわりに1クロックあたりの平均実行命令数は2程度にとどまっている。

# Itaniumの事業戦略

**21世紀突入と同時にインテルが新たに投入する64ビットCPU、Itaniumは既存のx86 CPUとはまったく別ラインの製品だ。現在でも好調な同社のx86 CPUシリーズとは別に、Itaniumをリリースするその真意は？ Itaniumの必要性とその事業戦略をインテルエンタープライズプラットフォームマーケティングマネージャの平野浩介氏に聞いた。**

Interview

## Itaniumの必然性

**――x86 CPUも順調に進歩しているように見える今、なぜItaniumが必要なのでしょう？**

**平野：**Itaniumが必要な理由は、主に以下の3つです。

まず、x86の性能向上がアーキテクチャ的にかなり厳しいことです。IA-32はスーパースカラー、アウトオブオーダーの命令実行など、RISC CPUと同等以上の技術が投入されています。しかし動的なスケジューリングでこれ以上に命令の並列度を上げるのは、困難です。

しかし、ItaniumのEPICアーキテクチャならば、コンパイラの性能にもよりますが、理論上はかなり並列度を向上させることが可能です。

2つめに64ビット化によって、非常に大きなメモリ空間が利用できることです。企業の基幹部門などで用いられるデータベースなど、巨大なデータサイズのアプリケーションが増えています。データサイズが、32ビットCPUでフラットに扱える、4Gバイトをオーバーするのが当たり前になりつつあるのです。

もちろん、XeonとWindows Datacenter Serverや Advanced Serverを組み合わせれば、4Gバイトを越えるメモリを扱えますが、ページングなど余分なテクニックを使用するため、メモリを増やしても性能がリニアに向上しないのです。

64ビットのItaniumなら、このような巨大なデータを扱いたいという要求を満たすことが可能です。

そして3つめは、より精度の高い計算を高速に行いたいという要求があることです。たとえば金融業界では、倍精度の浮動小数点データを用いて2000×2000の行列演算を行い、瞬時に答えを出してそれを元に売買の判断をする必要があります。

32ビットマシンでも64ビットのデータを扱うことは可能ですが、データの分割やメモリに展開するなどの作業が必要になります。

Itaniumなら特別な作業なしで64ビットのデータを扱えるため、32ビットマシンと比べてはるかに強力な処理能力を提供できます。

## 事業戦略

**――Itanium搭載機種の具体的な事業戦略をお聞かせください。**

**平野：**まず、膨大な処理能力が要求される、バックエンドのデータベースマシンに、早期に導入されていくことを期待しています。SQL Server、Oracle、DB2など主要なソフトウェアの移植が進行中で、ここ1、2年で導入が進むでしょう。

ミドルレンジのアプリケーションサーバについては、負荷が高い部分、または負荷の変動率が大きいために余裕を持たせたい部分から導入されると考えています。

フロントエンドについては、当分32ビット機が主流と考えています。大規模なWebサーバのフロントエンドは、Webサーバ、ファイアウォール、ロードバランサ、キャッシュサーバなどのアプライアンスマシンの組み合わせです。

その中で、セキュリティ処理を行っているマシンについては、ハードウェアアクセラレータよりもItaniumのソフトウェア処理のほうが高速です。ハードウェアアクセラレータは高価ですから、置き換える価値があるでしょう。

また、大量のメモリを搭載して、多くのセッションをメモリに置く必要があるキャッシュサーバでもItanium機が活かせます。人気のあるWebサイトは、ユーザーの期待も高いため、4秒も待ってくれないと言われていますから。

**――現在x86のシステムを利用しているユーザーに対して、どのように移行をサポートされる予定ですか？**

**平野：**導入事例をみると今までx86を使っていたから、すぐにItaniumに移行するという例は少ないようです。お客様それぞれに資産もありますし。

大手のお客様では、新しいサービスとともに、導入するマシンの台数が増えています。ですから移行ではなく、新しく導入する際にItaniumを選択していただくケースが多いようです。データについては、x86のシステムで作成したものがそのまま利用できますので、OSやアプリケーションプログラム、ユーティリティなどを64ビット化すれば、移行が可能です。

**――Itaniumの後継機種については、どのような展開を予定していますか？**

**平野：**前回のIDF（Intel Developer Forum）で、テクノロジーデモとして1U筐体に入ったMcKinley（Itaniumの次世代バージョンのコードネーム）のデュアルマシンを公開しました。McKinleyやその次の世代なら、1U筐体に2個搭載することも十分可能で、サーバアプライアンスも登場するでしょう。

## Linuxについて

**――Itanium搭載機にはWindows XPの64ビット版がバンドルされますが、これはまだベータ版であり、ユーザーがLinuxを選択することも多いと思われます。インテルとLinuxコミュニティとの関係をお聞かせください。**

**平野：**LinuxのItanium対応については、インテルが全面的に協力を行っています。まだItaniumに関する情報が開示されていなかった時期に、VA Linux社を中心に、Trillianプロジェクトを開始しました。その後、コミュニティと協議の結果、カーネル2.4のソースツリーにItanium用のコードを統合することになりました。

McKinleyに関してはまだ何も情報公開されていませんが、McKinley対応Linuxも開発が進められています。

そして適切な時期に、Itanium用のコードと同じようにLinuxカーネルのソースツリーに統合してもらえればと考えています。

ただ、企業の基幹部門で使われることを考えますと、「安定している」だけでは足りません。たとえば、バックアップツールにしてみても、Solaris、HP-UX、AIXといった商用Linuxと比較すると明らかに見劣りします。Linuxは安定して動作しますが、ミッションクリティカルな分野に安心して採用できるに足るユーティリティなどが不足しているのです。

そこで、Computer Associates、IBM、インテルが主幹事となり、日本のメーカーがメジャースポンサーとなって、OSDN（Open Source Developer Network）を作りました。OSDNの主目的は、ミッションクリティカルな用途に利用できるエンタープライズ系Linuxの開発です。

すでに米国オレゴンでラボが稼動しています。そこでは16ウェイのItaniumマシンが利用できるように公開されています。そして日本にも、今年12月までに2番めのラボを開設します。

エンタープライズ環境で使用されている大規模なマシンをオープンソースコミュニティのメンバーに公開することで、Linuxに貢献していきます。

## EPICアーキテクチャとコンパイラ

**――EPICアーキテクチャは、コンパイラの性能が重要になるとのことですが。**

**平野：**率直に言って、現在のGCCはあまり性能が良くありません。GCCの出力したコードで、Itaniumの性能を評価されるのは困りますね。

**――インテルのコンパイラをオープンソース化する予定は？**

**平野：**オープンソース化はしませんが、コンパイラ作成に必要な技術的なフィードバックは行います。

**――そのフィードバックを元に、GCCも改良されるでしょうか？**

**平野：**2年もたてば、EPICアーキテクチャ用にチューニングが進んだコンパイラがオープンソースで登場するでしょう。技術者にとっても、チャレンジしがいのある課題だと思います。特定の用途に関しては、インテル製のコンパイラをしのぐものが出てくるでしょう。

**――コミュニティに対して何か希望することはありますか？**

**平野：**現状ではLinuxが最も進んだItanium用のOSです。これを機に、ミドルレンジやバックエンドにもLinuxが利用されるようになる可能性があると思います。

必要な情報は、Webなどを利用して提供しますし、OSDNのラボも日本に作りますので、協力をお願いします。

（聞き手：編集部　飯岡）

平野 浩介氏
インテル株式会社
エンタープライズプラットフォーム
マーケティングマネージャ

# Itaniumマシンを徹底解剖する

今年5月末の正式発表以来、各メーカーからItanium搭載機種の発表が相次いでいる。一部のハイエンド機を除くと、インテルの460GXチップセットを利用したマザーボードを採用し、2個までのItaniumをサポートするという構成のマシンが多いようだ。

Itanium搭載機の第一世代ということもあり、これらのマシンはハードウェア的には差がないように見える。筐体の形状が似ていたり、マザーボード上の部品配置がほぼ同じという製品が多かった。

そのため、現状では差異化のポイントは、ライブラリの提供などソフトウェアに関する点と、サポート体制などのサービスが中心のようだ。

各メーカーが独自設計のハードウェアを投入してくるのは、次世代版のItaniumであるMcKinley（コードネーム）が使われるようになってからと思われる。

それは、現行のItaniumのデビューが遅れたため、McKinleyの登場まであまり間がなくなってしまったためだ。結果として、今のItaniumは、強力な浮動小数点演算能力を活かした実際の業務に使われる以外に、IA-64プラットフォームの開発・テスト環境として利用されることが多くなっている。

## ハードウェア構成の特徴

今回、三菱電機インフォメーションテクノロジーと日本SGIのItaniumマシンをテストすることができた。これらのマシンを通して、ハードウェア構成の特徴を探ってみよう。

写真1は、三菱電機インフォメーションテクノロジーのFT8000 20Iaの内部を撮影したものだ。通常のPCとは上下が逆、つまりCPUが下部に、拡張スロットが上部にあることが見てとれる。Itaniumは巨大なヒートシンクを装備するため、とても重い。そのうえ、CPUに電力を供給するパワーポッドを隣に配置する必要がある。このパワーポッドもCPU同様に重く、これらをケース内の上部に配置してしまっては、マザーボードがたわんだり、最悪の場合破損してしまうおそれがある。もちろん、重心が上にくるので、安定性の面でも望ましくない。そのため、これらをマザーボード上の下部に配置するようになっている。

またSGIのSilicon Graphics 750では、マザーボードの最下部に垂直にCPU用のドーターボードを取り付けるような構造になっており、CPUとパワーポッドは、筐体内部の底部に位置している。これも同様の配慮であろう。

## CPUの冷却

Itaniumは、その高性能の代償として非常に消費電力が高いCPUだ。最近では、ハイエンドを中心にx86系CPUでも消費電力の高い製品が増えているが、それらを大幅に上回る大食らいである。

消費電力の高さは、発熱量の増大に直結する。そのためItaniumマシンは、CPUをきちんと冷やしてやる必要がある。そこで、1Uのサーバで見られるような、筐体の前後にファンを置いて強制的に空気の流れを作るような方式がとられている。

写真1の下半分を見ると、CPUを覆うように透明のアクリル製カバーがかけられており、ダクトのようになっていることがわかるだろう。さらに写真2のように、このダクトの前後に2基ずつのファンが設置され、筐体の前から後ろに空気の流れを作るようになっている。動作中に、このダクトを外すのは無謀だ。

SGIのSilicon Graphics 750も同様に、筐体の前後にファンを備え、発泡スチロール製のカバーでCPUを覆ってダクトを形成していた。見た目はFT8000 20Iaのほうがいいが、冷却効果は同等であろう。

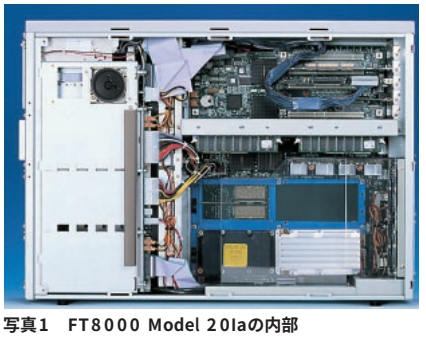


写真1　FT8000 Model 20Iaの内部
重いCPUやパワーポッドが下に配置され、透明のダクトに覆われている。

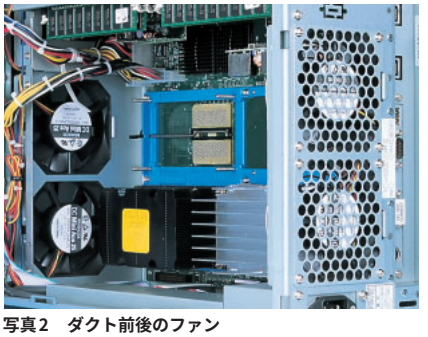


写真2　ダクト前後のファン
合計4個のファンで強制的に空気の流れを作っている。相当うるさい。

## パワーポッド

Itanium本体は、小型のハードディスクのような外観をしている（**写真3**）。底面に418本のピンがあり、マザーボード上のソケットに配置する。x86系CPUでは、電力も底面のピンから供給されるが、Itaniumに必要な電力は膨大であり、ピンからでは供給できない。

そこで代わりに、エッジタイプのコネクタを側面に持ち、Itaniumのすぐ横に設置されているパワーポッドから供給するようになっている（**写真4**）。ItaniumとパワーポッドはエッジコネクタをCPUとパワーポッドの消費電力は非常に大きく、1セットあたり100Wを超えると言われている。

## メモリ用ドーターボード

今回試用した2機種はどちらも、メモリを専用のドーターボード（**写真5**）上に増設するようになっている。Silicon Graphics 750については、ドーターボードを2つ持ち、インターリーブアクセスをすることによって、メモリ帯域を広げている。

マザーボードに直接メモリスロットを設ける代わりにドーターボードを用いることで、マザーボードのサイズに関係なく、メモリを増設することが可能になる。また、メモリ増設の作業自体も簡単になり、メンテナンス性が向上する。

メモリについては、ECC付きのPC100またはPC133 SDRAMが利用できる。

## 対応OS

Itaniumに対応、もしくは対応予定のOSとしては、Linux、Windows、そして各メーカーのUNIXがある。UNIXについては、ヒューレット・パッカードのhp-ux、IBMのAIXなどが移植される予定だ。サン・マイクロシステムズのSolarisは、当初移植作業が行われていたが、現在は中断されている。

Windowsは、Windows 2000の後継であるWindows XPを64ビット化したものである。x86用のWindows XP自体がまだ発売されていないOSであり、5月29日のItaniumの正式リリース時に間に合わないと言われていた。

しかし、直前になって急遽サーバ版（Windows Advanced Server Limited Edition）またはワークステーション版（Windows XP 64-bit Edition）のWindowsがItanium搭載機にバンドルされることが決定した。

これらはどちらもベータ版という扱いだが、マイクロソフトによるサポートが提供されるうえに、正式版のリリース時には無償でアップグレードできることになっている。

Linuxカーネルについては、比較的早い時期からTrillianプロジェクトにおいて、Itaniumへの移植が進められていた。そしてカーネル2.4のソースツリーにItanium用のコードが統合されることになった。

Trillianはその後、IA-64 Linux Projectに名前を変え、現在も活動が続けられている（**画面1**）。この成果をもとに、Caldera、Red Hat、SuSE、Turbolinuxの各社からItaniumに対応したディストリビューションがリリースされている。


画面1　IA-64 Linux Project
多くのディストリビュータ、ハードウェアベンダが加わっている。

写真3　Itanium
3次キャッシュを統合するため、小型のハードディスクのような外観を持つ。

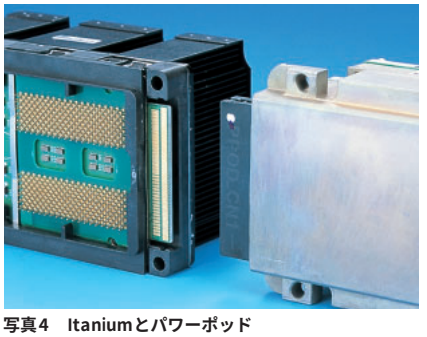


写真4　Itaniumとパワーポッド
パワーポッドはエッジコネクタを介してItaniumに接続され、電力を供給する。

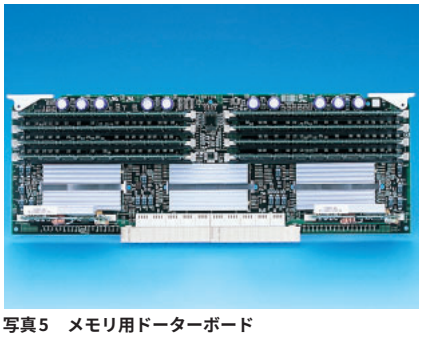


写真5　メモリ用ドーターボード
ドーターボード上にメモリスロットを設けることで、メンテナンス性を向上させている。

三菱電機インフォメーションテクノロジー株式会社
http://www.mdit.co.jp/

# FT8000 Model 20Ia

250万円～

FT8000 Model 20Ia（**写真1**は、三菱電機インフォメーションテクノロジーが製造・販売するItanium搭載のサーバ機である。

本機は大きめのタワーケースだが、高さ4Uで19インチラックマウントタイプのModel 25Iaもある。並列コンピューティング、負荷分散システムなど複数台のマシンを利用する際には、25Iaのほうが効率良く設置できる。

チップセットはインテルの460GXを採用し、733MHz／2Mバイト 3次キャッシュ、800MHz／4Mバイト 3次キャッシュの両グレードのItaniumをサポートしている。デュアルプロセッサにも対応する。メモリはECC付きPC100 SDRAMを標準で512Mバイト搭載しており、8Gバイトまで拡張が可能。

64ビットのPCIスロットを5本備え、うち3本は66MHzにも対応している。またホットプラグにも対応しているため、PCI拡張カードのホットプラグの評価／検証が行える。

Ultra160 SCSIに対応した64ビットPCI用のアダプテック29160 SCSIアダプタ（**写真2**）を装備しており、SCSIハードディスクは2台まで内蔵可能だ。3台以上に増設する場合は、オプションで用意されている拡張ディスクモジュールを用いる。

内蔵用ドライブベイは、SCAコネクタによるホットプラグに対応しており、筐体前面から着脱可能になっている。そのため、通常運用する環境と開発／テスト用の環境を分け、評価／検証が容易に行えるようになっている。

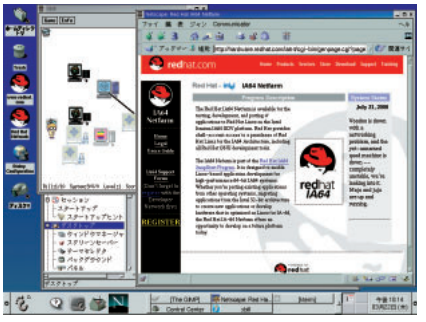
本機はAGPスロットを持たないので、グラフィックスカードはPCI接続のATI Rage 128GL（**写真3**）を用いている。また、ネットワークカードは、インテルのPRO/100＋を搭載している。

ホットプラグ対応の電源ユニットは、Itanium 1個につき1台必要（デュアルCPU時は2台必要）で、冗長構成用にもう1台増設できる。

Red Hat Linux（**画面1**）、Turbolinux、Windows Advanced Server（ベータ版）の各OSで動作確認がとられている。

Itanium版のLinuxは、起動時にLILOを用いない点などをのぞけば、x86版とほとんど差がない。特にX Window Systemのデスクトップが起動してしまえば、まったく同じように扱える。何か操作を行っても、もたつくことはない。

写真1　FT8000 Model 20Ia
奥行き方向に長い大型のタワーケースを採用している。ラックマウントタイプもある。



画面1　Itanium対応Red Hat Linux
デスクトップを起動させれば、x86環境とまったく同じように利用できる。

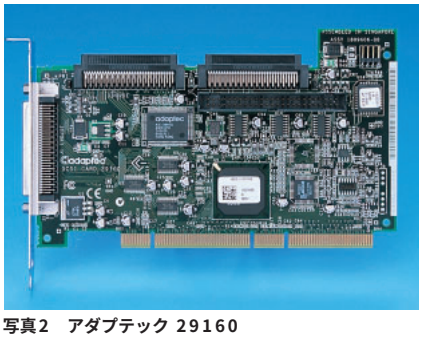


写真2　アダプテック 29160
Ultra160 SCSIに対応したSCSIアダプタ。64ビットPCIスロットが利用できる。

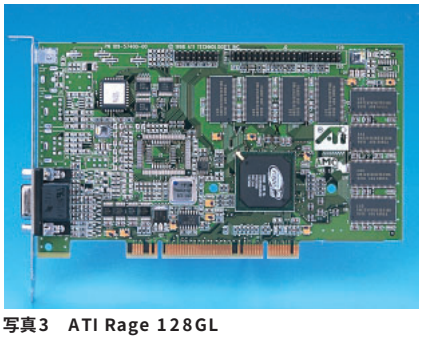


写真3　ATI Rage 128GL
本機はAGPスロットを持たないので、PCIタイプのグラフィックスカードを採用している。

日本SGI株式会社
http://www.sgi.co.jp/

# Silicon Graphics 750

近日中に発表予定

Silicon Graphics 750（**写真4**）は、日本SGIがリリースしたItanium搭載機である。

チップセットはインテルの460GXを採用し、733MHz／2Mバイト 3次キャッシュ、800MHz／4Mバイト 3次キャッシュの両グレードのItaniumをサポートしている。デュアルプロセッサにも対応する。

PC100 SDRAMを採用し、標準で1～4Gバイトのメモリを搭載している。最大16Gバイトまで拡張が可能。2枚搭載されているメモリボードをインターリーブアクセスすることで最大4.2Gバイト／秒のメモリ帯域を持つ。

64ビットのPCIスロットを7本備え、うち5本は66MHzにも対応している。

Ultra160 SCSIに対応したqlogic QLA12160アダプタ（**写真5**）を装備しており、標準で18GバイトのSCSIハードディスクを1台内蔵している。最大5台までハードディスクドライブを内蔵できる。

グラフィックスカードは、AGP ProスロットにATI Xpert 2000 Pro（**写真6**）を装備しており、科学技術計算の結果をそのまま可視化（2D）することができる。また、ネットワークインターフェイスはマザーボード上に装備されている。

Silicon Graphics 750の標準OSは、64ビット版のLinuxである。また、SGIが開発したSCSL（Scientific Computing Software Libray）がバンドルされている。SCSLは、Silicon Graphics 750に最適化された、最新の科学数学演算ライブラリである。ブロック化、マルチスレッド化、最適化されたサブルーチンを用いており、アプリケーションの性能と効率を向上させる。

SCSLは、共有メモリ並列化、SGIによってサポートされるすべてのプラットフォームにおけるコード移植の簡易化のための同じAPI、標準のBLAS（Basic Linear Algebra Subprograms）とLAPACK（Linear Algebra Package）の機能を持ち、コード開発の工数を削減する。また、64ビット整数の引数をサポートし、古いプラットフォームからの移植も容易になっている。

高速ネットワークのMyrinet2000（**写真7**）を利用することによって、Beowulf型のクラスタとして利用することも可能で、その場合は8、16、32プロセッサ構成に対応する。


写真4　Silicon Graphics 750
曲線で構成されたフロントパネルを持つ、ワークステーションである。

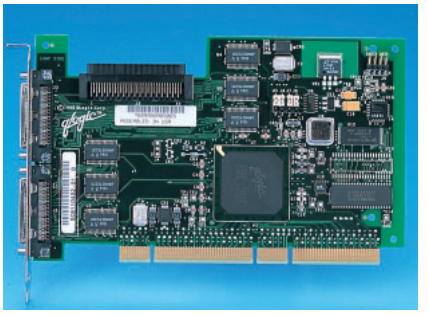


写真5　qlogic QLA12160
Ultra160 SCSIに対応したSCSIアダプタ。64ビットPCIスロットが利用できる。

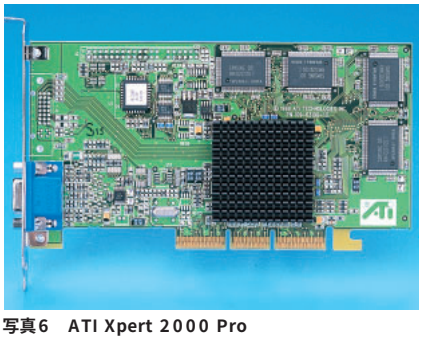


写真6　ATI Xpert 2000 Pro
現状では、3Dアクセラレーション機能はサポートされていない。

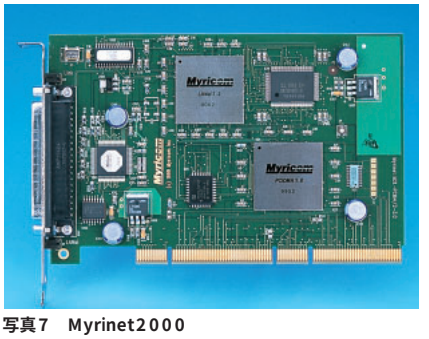


写真7　Myrinet2000
Beowulf型のクラスタとして利用する際の高速ネットワークを構成するカードだ。

## 各社のリリース状況

Itanium搭載機は、各社から発表／発売されている。性能や目的別に分類すると、2個までItaniumを搭載可能なワークステーション、4個までItaniumを搭載できるサーバ、そして8～16ウェイに対応したハイエンドサーバの3種がある。

今回紹介する7機種以外にも、東芝、富士通、ノーザンライツ、米IBM、米Gatewayなどが搭載機種のリリースを予定している。つまり大手のPCベンダーは、ほとんどすべてがItanium機の市場に参入することになる。

なかでも米コンパックは、最も強力にItaniumにコミットするようだ。同社はPCサーバシリーズのProLiantラインに、最大4個までのItaniumを搭載可能なモデルをリリースする予定だ。

そのほか、現在Alphaプロセッサを用いている64ビットサーバ（AlphaServer）と、MIPSのRISCプロセッサを用いているNonStopシリーズを、数年かけてItanium搭載機種で置き換えることにしている。AlphaServerやNonStopシリーズはいずれも同社が買収した企業（DEC、Tandem）が所有していた製品である。

なお、Alphaの開発に携わっていた技術者は、インテルへの移籍が予定されており、おそらくはItaniumのプロセッサやコンパイラ開発に携わることになっている。

コンパックは、サーバ製品のCPUをItaniumに統一して、ハードウェアベンダーからソフトウェア／サービスプロバイダーへの転身を狙うと思われる。

日本ヒューレット・パッカード株式会社
http://www.jpn.hp.com/

### hp workstation i2000

191万円～

hp workstation i2000は、日本ヒューレット・パッカードから発売されたItanium搭載のワークステーションだ。最大2個のItaniumを搭載可能で、PC133 SDRAMを用いて4Gバイトまでメモリを拡張できる。

OSは、Windows XP のベータ版がプリインストールされるが、Red Hat Linux 7.1、hp-ux 11i Version 1.5のCD-ROMも付属している。

hp-ux 11iを利用する場合は、「Object Code Translation」機能により、PA-RISC用のバイナリを再コンパイルすることなく実行できる。

NVIDIAのQuadro2 Proを搭載したグラフィックスカードを採用しており、強力な3D描画機能を持つ。この機能は、NVIDIAのWebサイトからドライバを入手することで、Red Hat Linuxでも利用可能だ。

デルコンピュータ株式会社
http://www.dell.com/jp/

### PowerEdge 7150

価格未定

PowerEdge 7150は、デルコンピュータが今年第3四半期以降に出荷を予定している、エンタープライズ向けサーバである。最大4個のItaniumを搭載可能で、ECC付きのSDRAMを用いて、64Gバイトまでメモリを拡張できる。チップセットは、インテルの460GXを用いている。

OSは、Red Hat Linuxおよび、Windowsに対応する予定。

10本の64ビットPCIスロットを持ち、そのうち8本は、電源が入った状態での抜き差し（ホットプラグ）が可能になっている。筐体内に4つ用意されているハードディスクのドライブベイもホットプラグに対応している。

外部記憶装置として、同社の「PowerVault」シリーズのRAIDシステム、テープバックアップシステムなど、さまざまな製品が利用可能だ。

ぷらっとホーム株式会社
http://www.plathome.co.jp/

## PrizeX 9600M

195万円〜

PrizeX 9600は、ぷらっとホームから発売されているItanium搭載のワークステーションだ。最大2個のItaniumを搭載可能で、ECC付きのPC100 SDRAMを用いて8Gバイトまでメモリを拡張できる。チップセットは、インテルの460GXを用いている。

Turbolinux、Red Hat Linuxのほか、Windows XP、 Windows Advanced Server（ともにベータ版）にも対応する。

標準構成では、グラフィックスカードにATI Xpert128、NICにインテルPRO/100＋、SCSIカードにアダプテック29160（64ビットPCI）を備える。ハードディスクは、Ultra160m/SCSIに対応した18.2Gバイトのものを1個搭載している（最大2個内蔵可能）。高さ4Uのラックマウントケースに、ホットスワップ対応の電源を2台内蔵している。

株式会社日立製作所
http://www.hitachi.co.jp/

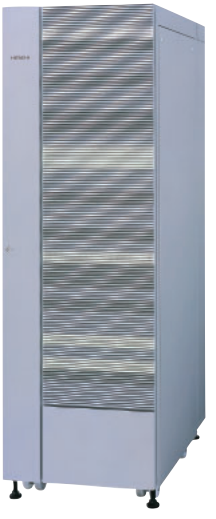
## アドバンストサーバHA8000-ex/880

1230万円〜

HA8000-ex/880は、Itaniumを最大8個まで搭載できるハイエンドサーバである。独自開発のチップセットを採用し、最大64Gバイトのメモリを搭載可能。さらに最大4個のCPUを搭載可能な「ノード」ごとに64Mバイトのノードキャッシュを搭載し、高速化を図っている。当初はRed Hat LinuxまたはTurbolinuxがインストールされるが、Windows Advanced Server、 hp-ux 11i Version 1.5にも対応予定だ。

1台のex/880を論理的に多重化し、複数のサーバのように利用するMLPF（ Multiple Logical Processor Feature）を備え、複数のOSを同時に稼動させられる。

また、最大128台のex/880を高速ネットワーク「Myrinet2000」で接続することで並列クラスタを構成し、大規模な技術計算を高速に実行できる。



日本電気株式会社
http://www.nec.co.jp/

## Express5800/1160Xa

1548万円〜

Express5800/1160Xaは、Itaniumを最大16個まで搭載可能な、ハイエンドサーバである。最大64Gバイトのメモリ、128のPCIスロットをサポートする。対応OSは、 Windows Advanced ServerとLinuxだ。

NECは以前より、Itaniumを搭載したサーバのプロトタイプ「AzusA」を公開していた。この「AzusA」という名前は、1160Xaの同型機「TX7/AzusA」に受け継がれている。TX7/AzusAは、対応OSがhp-ux 11i Version 1.5であり、主に科学技術分野をターゲットとしている。

1160Xaは、16個のCPUを4個単位（セル）で分割し、最大4つのサブシステムとして独立して運用ができる。各サブシステムは個別に起動、再起動が行える。セルはCPUのほか、メモリ、独自開発のチップセットからなり、セル間は、高速バックプレーンで接続されている。

# 家庭内IPv6ネットワークへの道

前回（6月号）でUSAGIスタックも無事にインストールできました。今回は、IPv6ネットワークの利用に必要な基礎知識を解説したあと、構築したローカルのIPv6ネットワークをインターネットに接続します。

文：國武功一　USAGIプロジェクト／グルーオン・パートナーズ（株）
*Text : Koichi Kunitake*

## IPv6を取り巻くエトセトラ

いきなり懺悔です。前回の記事でFreeNet6へのトンネルの張り方を書いたのですが、あのとおりやっても、おそらくつながりません。サイトからダウンロードしたスクリプトをそのまま実行すればよいと書いてしまいましたが、多くの場合は若干の修正が必要です。スクリプトを適当なエディタで開いてみてください。

```
system(`ifconfig eth0 add 3ffe:b00:c18:1fff:0:0:0:XXXX`);
```

という箇所があると思います。この“eth0”を使用している環境に合わせ、“ppp0”のように編集してやる必要があります。たまたまeth0を持っていたら、そのままでも動く可能性があります。確認が足らず、申し訳ありませんでした。

で、懺悔はそれぐらいにして、皆さんは先日幕張メッセで開催されたNetworld+Interop（N+I）へ足を運ばれたでしょうか？　N+Iでは、毎年各社が持ち寄る最新ネットワーク機材を使ってShowNetと呼ばれるデモンストレーション・ネットワークを構築します。これには、最新機材が一同に会して相互接続性などを試すたいへん貴重な機会という側面もあるわけです。

そのN+Iで、去年に引続きIPv6 ShowCaseが開催されました。その会場はメイン会場とは若干離れたイベントホールだったにもかかわらず、去年とは比べものにならないほどの大盛況となりました。去年、各社が出品していた機材は、ルータやサーバといった一般受けしそうにないものが多かったように思えます（ネットワークゲームの展示があったので、それは一般受けしたかも）。しかし今回は、今までIPv6の開発者が「IPv6によって、将来はこんなものがネットワークにつながるようになるんだよ」と語っていた将来像が、家電系の品々やセンサー類、あるいはインターネット・カーという姿をとって、目に見える形で出て来ました。今まで夢としてしか語られていなかったものが、着実に現実のものになろうとしているようです。

IPv6の話をするとよく、「IPv4からの移行なんてありえない！」という反応をされる方もありますが、個人的にはそれでかまわないと思っています。IPv6のネットワークは、前述した家電や組み込み系の機器たち、はたまたインターネット・カーといった、新しいデバイス群によって成長していくことでしょう。そして、そのネットワークに蓄積された資源にアクセスしたくな

ったユーザーが、IPv6を利用するようになるのではないでしょうか。新規にネットワークを構築するのであれば、将来性を見据え、最初からIPv6を導入するというのもひとつの手でしょう。

さて、インターネットの黎明期、IPv4は学術利用が先行し、その後、商用利用が広まっていきました。しかし、現在では状況が大きく変わっていて、IPv6はいきなり商用利用が期待されています。そういった状況から、前回紹介したTAHIプロジェクトによるIPv6スタックの品質管理という取り組みは、非常に重要な意味を持ちます。地道な作業だけに頭が下がる思いです。

IPv4では、こういった技術に対するブラッシュアップを担っていたのは米国だったと言っていいでしょう。私達は日々、インターネットを利用することによってその恩恵を受けているわけです。IPv6の開発と運用については、現在のところ、日本やヨーロッパ諸国が一歩先んじているといってよいでしょう。今度は私やあなたがインターネットの発展に寄与できる千載一遇のチャンスかもしれません。

「いやぁ～俺はそんなのいいや。使わなくちゃいけなくなってからで」というのでもいいかもしれません。

しかし、LinuxにおけるIPv6はまだまだ成長段階にあります。IPv4の世界でLinuxが活躍したのと同様に、IPv6の世界でもLinuxが活躍できるよう、ぜひとも開発に協力していただきたいと思っています。プログラムを書かなくても、実際に使って不具合をどんどん開発者へ報告すれば、それだけLinuxのIPv6スタックは鍛えられます。それは、今までLinuxが歩んで来た道そのものではないでしょうか。

さてさて、前置きが長くなってしまいました。では、前回省略したIPv6の基礎知識について補足してから、実際のネットワーク構築方法を説明したいと思います。

## IPv6の基礎知識

### IPv6用語の基礎知識

まず最初に、IPv6に特有の使い方をする用語から説明しましょう。

- **ノード（node）**
  IPv6が実装されている機器。
- **ルータ（router）**
  自分宛て以外のIPv6パケットを他へ転送するノード。
- **ホスト（host）**
  ルータではないノード。
- **上位層（upper layer）**
  IPv6の直上のプロトコル層。TCPやUDPなどがそれに当たる。
- **リンク（link）**
  IPv6の直下の層。イーサネットやPPPリンクなどがそれに当たる。IPv6 over IPv4といったトンネルもリンクに相当する。
- **近隣（neighbors）**
  同じリンクに接続しているノード。
- **インターフェイス（interface）**
  ノードがリンクへ接続するためのアタッチメント。
- **アドレス（address）**
  1つまたは複数のインターフェイスに割り当てられた、IPv6層の識別子。
- **パケット（packet）**
  IPv6ヘッダがついたペイロードを指す。IPv6ヘッダに含まれているペイロード長は、IPv4とは違い、IPv6ヘッダを含まないペイロードのみの長さを示していることに注意（ただし拡張ヘッダはペイロードに含まれる）。
- **リンクMTU（link MTU：Maximum Transmission Unit）**
  そのリンクで伝送可能な最大パケット長。
- **パスMTU（path MTU）**
  接続元から接続先までのあいだに経由するリンクのなかで、最も小さなリンクMTU。

実は前回の記事では、ノード、ホスト、ルータといった用語を使い分けていたつもりですが、大丈夫だったかな？　実際のところ、ホストとルータとでは、パケットの処理のしかたで挙動に違いがありますので、開発に携わろうという方は特に注意が必要です。

### IPv6ヘッダ

IPv6ヘッダを**図1**に示します。IPv4ヘッダと比較することはここではしませんが、IPv4ヘッダをご存じの方は、ヘッダがいくぶん簡略になっていることに気づかれることと思います。これは、ヘッダを簡略化してヘッダサイズを小さくしようとしていることはもちろんのこと（IPv4ヘッダより大きくなってますが、努力はしているんです;-）、それによってヘッダ処理を簡略化し、ルータ機器への負担を軽くすることに主眼がおかれています。

IPv4ヘッダ内のオプションに相当するのが、ヘッダの後ろの拡張ヘッダです。必要な機能だけを必要なだけ数珠つなぎに並べます（**図2**）。拡張ヘッダには、**表1**に挙げるものがあります。

拡張ヘッダの並びは、まったく無秩序というわけではなく、表1に挙げた順序で配置することが推奨されています[*1]。

## IPv6のアドレス体系

RFC2373を読め！

以上。

というのが一番正確な知識が得られてベストなんですが、それをやっちゃうと怒られちゃうので、以下、IPv6のアドレス体系について触れたいと思います。さて、車ひとつとってみても、分類方法によって、同じ車をセダンと言ってみたり、FF（前輪駆動）と言ったりします。それと同様に、IPv6アドレスにもいくつかの分類方法があります。前回の記事では、

- **ユニキャストアドレス**
- **エニーキャストアドレス**
- **マルチキャストアドレス**

の3種類に分ける分類を説明しました。これはパケットの振舞い方を示すものなので、一番基本といえるものでしょう。このほかに、アドレスがどこまで有効かを示す、適用範囲（スコープ）による分類方法もあります。これは、

- **グローバルアドレス**

IPv4でいうところのグローバルアドレスと同義。

- **サイトローカルアドレス**

IPv4でいうところのプライベートアドレスのようなもの。ルータは、このアドレスを始点アドレスとするパケットを、他組織へは送りません。

- **リンクローカルアドレス**

ノードが直接つながっているリンク（セグメント）内でのみ使用可能なアドレス。リンク上にルータがなく、ルータから通知されるメッセージ（RA）によるグローバルアドレスの自動設定ができない場合などに使用されます。

の3種類です。また特殊なアドレスとして、以下のものがあります。

- **未指定アドレス（The Unspecified Address）**

アドレスがつけられていないことを示します。“ ： ： ”と表記されます。このアドレスは、システムの初期化中

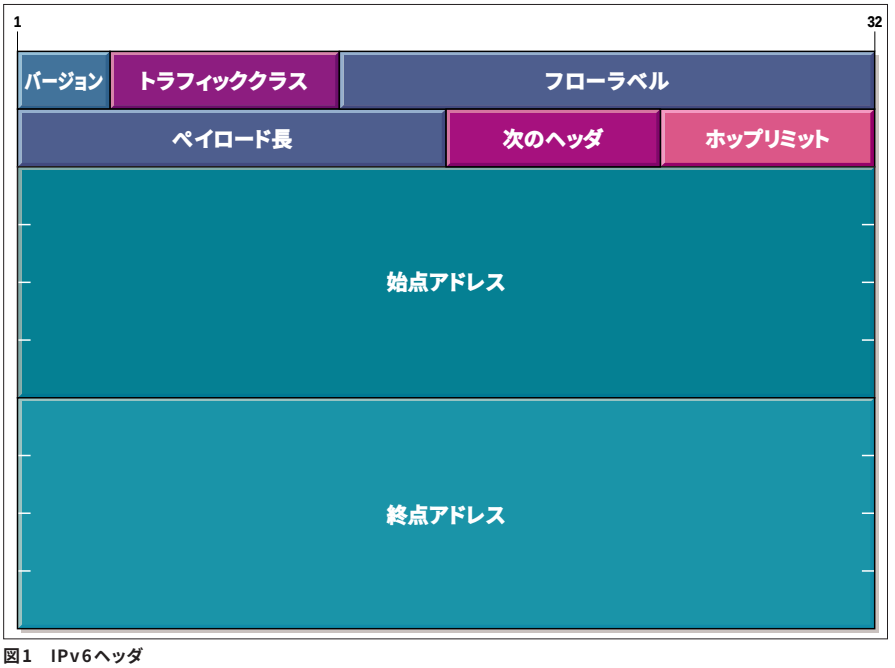



図1　IPv6ヘッダ

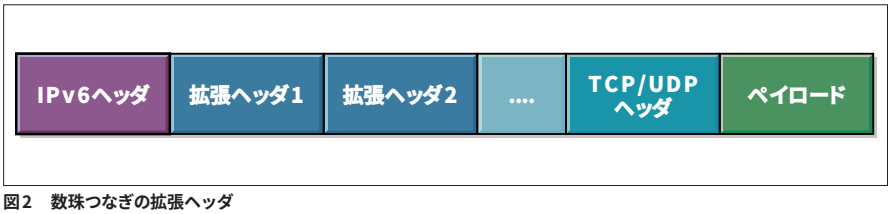



図2　数珠つなぎの拡張ヘッダ

| 拡張ヘッダ | 説　明 |
|---|---|
| 中継点オプションヘッダ（Hop-by-Hop Options header） | パケットの配達経路上のすべてのノードに評価される情報が含まれています |
| 終点オプションヘッダ（Destination Options header） | 終点ノードでのみ評価されるオプションが含まれています |
| 経路制御ヘッダ（Routing header） | 始点から終点へとパケットが伝送される際に、訪れるべき中継ノードの一覧が含まれています |
| 断片ヘッダ（Fragment header） | 送信したパケットがパスMTUよりも大きい場合に、パケットがフラグメント化（分割）されていることを示します。IPv6はIPv4とは違って、途中のルータなどでパケットが断片化されることはなく、始点でパケットの分割を行います。そのため、ルータへの負荷が小さくなることや、転送速度の向上が期待されています |
| 認証ヘッダ（Authentication header）<br>暗号ペイロードヘッダ（Encapsulating Security Payload header） | IPsec使用時に使われるヘッダとなっています |

表1　拡張ヘッダ

でアドレスがまだついてないホストが、パケットを投げるときに始点アドレスとして使ったりします。

**・ループバックアドレス（The Loop back Address）**

自分自身を指すアドレス。IPv4では127.0.0.1として知られているものです。IPv6では::1が使用されます。IPv4と同様、自分自身にパケットを送信する場合などに使用します。

**・IPv4互換アドレス（IPv4-compatible IPv6 address）**

IPv6ノード同士がIPv4ネットワーク経由で通信する際に使用するアドレス。自動トンネリングという機能で使用されます（**図3**）。

**・IPv4射影アドレス（IPv4-mapped IPv6 address）**

IPv6ノードがIPv4しかサポートしていないノードと通信する際に使用します（**図4**）。

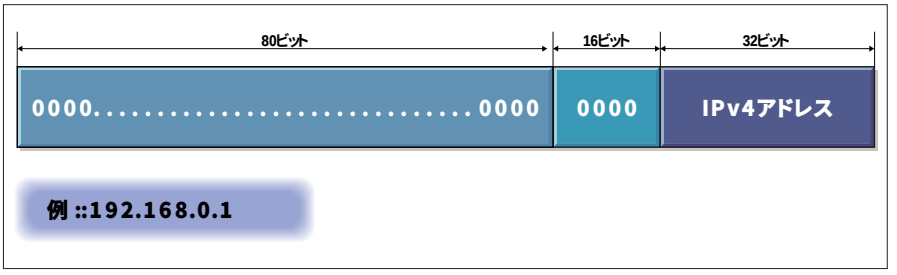



**図3　IPv4互換アドレス**

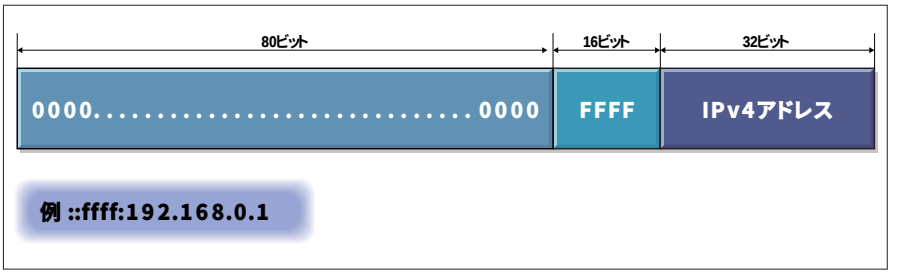



**図4　IPv4射影アドレス**

最後の2つのアドレスは、説明からもわかるとおり、IPv4からの移行をサポートするために用意されたものです。特に後者の射影アドレス（マップド・アドレスと呼ばれることが多い）が曲者なんですが、それについては後述します。一見してわかるように、このアドレス表記はIPv6のアドレス表記に従っていません。利便性を優先して、IPv6のプレフィクスにIPv4アドレスをそのままくっつけてあるのです。

## 家庭内IPv6ネットワーク

さてさて、実際にNLA（Next Level Aggregator）組織相当のアドレスをもらって、そのネットワークを運用していくことを考えましょう。この/48のネットワークアドレスは、固定のIPv4アドレスさえ用意できれば、個人でも大手のプロバイダから取得することが可能です。持っているIPv4アドレスを使って、トンネルという形態で対岸のIPv6ネットワークへつなごうというわけです。

ダイヤルアップの環境しかないよ……という方でも大丈夫。前回紹介したFreenet6が/48の配布を開始したようですので、一度試してみてはいかがでしょうか？

今回は、IPv4の固定アドレスが用意できたものとして話を進めます。固定アドレスというと、従来は企業などの専用線ユーザーにしか縁のないものでしたが、最近は固定アドレスを比較的安価に配布してもらえるADSLサービスなどの普及によって、ちょっと手を伸ばせば届くようになりました。サーバを自分で運用してみたい人にとっては、嬉しい環境が整ってきています。

ところで、現時点ではIPv6アドレスの多くは実験という名目で配付されていますので、この場合には突然接続が切れたり、配付されたアドレスを返さなくてはならなくなることも了解しておく必要があります。日本で個人にアドレスを配布しているISPですが、

```
http://6bone-jp.v6.wide.ad.jp/nla.html
```

の6bone-jpのNLAリストを参考に探してみてください。すでに商用サービスを開始しているISPもありますので、実際に各社のWebを確認してみるとよいでしょう。

### 実際のネットワーク構成

この記事で想定するネットワーク構成を**図5**に挙げます。使用するLinux

Glossary

＊1　一番最後に、再度終点オプションヘッダが来ることもあります。

Boxには、あらかじめUSAGIのSnap Kitのカーネルとコマンド群をインストールしておきます（6月号参照）。また、割り当てられたプレフィクスは、以下のとおりです。

```
3ffe:XXXX:YYYY::/48
```

ここでは、ISP側にあるRouterAとユーザー側が用意するRouterB（Linux君です）との間をトンネルでつなぎます。ルータ間の経路制御には、RIPngというルーティングプロトコルを使用しましょう。RouterAは、6月号で説明したトンネルブローカーに当たります。ISPとユーザー側でやりとりするIPv6パケットはIPv4のヘッダにくるまれ、ルータ間のIPv4ネットワーク上を流れることになります。

何はともあれ、トンネルを張りましょう。その前に、Linuxのsitデバイスについて少し説明しておきます。sitは、トンネルを張るときに使う仮想デバイスです。Linux Boxに複数のイーサネットカードを装着すると、eth0、eth1、eth2とインターフェイス名の末尾の数字が増えていきますよね？

sitデバイスも同様にトンネルを複数張っていくと、sit1、sit2のようになります。しかし、そのなかでもsit0だけはちょっと特殊です。sit0に対してトンネルの設定を行うと、実際のIPv6通信に使われるsitデバイスが別途に作成されるのです。この説明だけではわかりにくいですね。具体的な例を挙げて見てみましょう。たとえば、

```
RouterX 192.168.1.1/24
RouterY 192.168.2.1/24
RouterZ 192.168.3.1/24
```

の複数の接続先とトンネルを張ることを考えましょう。

**画面1**のように、トンネルの設定はすべてsit0に対して行います。たとえば、RouterXの先につながっているIPv6ネットワークとはsit1を通して通信し、RouterYの先につながっているIPv6ネットワークとはsit2を通して通信することになります。こうしてトンネルをいくつも張っていくと、どのインターフェイスがどのトンネルポイントとつながっているのかわからなくなってきます。そこで、iptunnelというコマンドを引数なしで実行すると、**画面2**の設定一覧を表示してくれます。

もちろん、このiptunnelを使ってトンネルを張ることも可能です。

```
# ifconfig sit0 up
# iptunnel add sit1 mode sit remote
192.168.2.1 local any ttl 64
# ifconfig sit1 up
```

```
# ifconfig sit0 up
# ifconfig sit0 tunnel ::192.168.1.1 …この瞬間にsit1が生成される
# ifconfig sit1 up
# ifconfig sit0 tunnel ::192.168.2.1 …この瞬間にsit2が生成される
# ifconfig sit2 up
# ifconfig sit0 tunnel ::192.168.3.1 …この瞬間にsit3が生成される
# ifconfig sit3 up
```

画面1　トンネルを生成する

```
# iptunnel
sit0: ipv6/ip  remote any  local any  ttl 64  nopmtudisc
sit1: ipv6/ip  remote 192.168.1.1  local any  ttl 64
sit2: ipv6/ip  remote 192.168.2.1  local any  ttl 64
sit3: ipv6/ip  remote 192.168.3.1  local any  ttl 64
```

画面2　トンネルの設定一覧を表示する

## Column

### DNSの不思議

IPv4からIPv6に移行しようとするとき、クライアントは当面デュアルスタックであることが想定されます。つまり通信相手がIPv6を話せるならIPv6で、逆にIPv4だけしか話せないならIPv4で、というわけです。では、いったいどうやって相手側がサポートしているIPプロトコルを知るのでしょうか？　実は、ここで活躍するのがおなじみのDNSサーバなのです。

たとえば、www.linux-ipv6.orgへアクセスすることを考えてみましょう。IPv4しかサポートしていない場合、ブラウザは、実際に接続する前に、そのFQDNをもとにしてIPv4アドレスが記録されたAレコードをDNSに問い合わせます。そしてその結果、“203.178.141.227”というアドレス情報を手に入れ、そのアドレスへ接続するわけです。

一般的に、IPv4／IPv6をサポートしたDNSサーバには、Aレコードのほかに AAAAレコードが登録されています。中身はIPv6アドレスです。つまり、デュアルスタックのクライアントは、まずDNSサーバへAAAAを問い合わせ、それが存在すればIPv6で接続し、Aレコードしかなければ IPv4を通じて接続します。

ではIPv6での接続が失敗した場合はどうなるのでしょうか？　当然ユーザーとしてはIPv6がダメならIPv4で接続したいわけですが、実はその挙動は、アプリケーションに依存します。前回IPv6対応ブラウザとして紹介したNetscape6ですが、残念ながらIPv4での再試行をしてくれません。ただし、こういった挙動のアプリケーションは減ってきているので、早くNetscape6も改善してほしいですね。

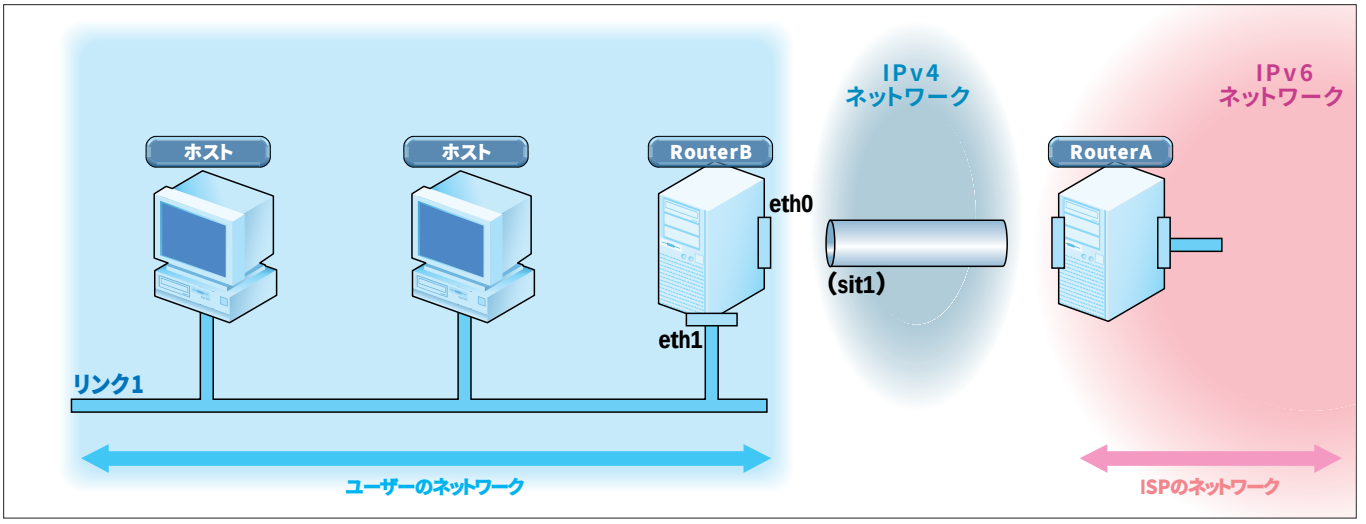



図5　実験用ネットワーク

不要となったトンネルは、

```
# iptunnel del sit1
```

を実行し、消してしまいましょう。

## Plug and Playの設定

これでトンネルの設定はOKですね？もちろんIPv6のネイティブ・サービスを導入したぜ！　という強者は、読み飛ばしてもかまわない部分でした。

さて、次にローカルネットワークのリンク1に接続したホストのために、RouterBにルータ通知メッセージ（Router Advertisement）デーモンの設定をしましょう。

この設定を適切に行っておくと、ホストはルータから通知されるRAを利用してアドレスを自動生成することができます。

実際にこれを実現するには、ルータ側でradvdなどのRA専用のデーモンを設定して起動する方法があります。以下は、そのradvdの設定ファイル/usr/local/v6/etc/radvd.confの例です。

```
interface eth1
{
        AdvSendAdvert on;
        prefix 3ffe:XXXX:YYYY::/48
        {
                AdvOnLink on;
                AdvAutonomous on;
        };
};
```

## Zebraでルーティング

さて、これでリンク1にぶら下がっているホストにはグローバルアドレスが自動生成されたことと思います。ですが、これだけでは外部のネットワークと通信することはできません。パケットの宛先を見て、次にどこへ転送するかを判断する、経路制御というものが必要となります。ふだんインターネットを通じてサーバなどにアクセスできるのも、ルータがパケットを正しい目的地へ転送するように経路制御をしてくれているおかげです。

経路制御の方法は大きく分けて、あらかじめ転送のルールを固定的に設定しておく静的経路制御と、ルータ同士で制御情報を送りあって常に最新のネットワーク状況を反映させる動的経路制御の2つがあります。大雑把に言ってしまえば、前者は小規模ネットワークに向き、後者は大規模ネットワークに向いている技術だといえます（ほんとうに大雑把な説明です）。

例に挙げた程度のネットワークであれば、ネットワークの安定性の良さから考えて静的経路制御を使用すべきでしょうが、何事も経験ということで、ここではRIPngによる動的経路制御を使ってみます。とはいうものの、ルーティングに興味はない、とりあえずIPv6のネットワークが欲しいだけ、という方のために、静的経路制御の例も以下に挙げておきます。その方法はとても簡単で、

```
# route -A inet6 add ::/0 gw
fe80::192.168.1.1 dev sit1
```

を実行すればOKです。もちろん“192.168.1.1”は、自分のトンネルの接続先となるIPv4アドレスと置き換えて読んでください[*2]。

うまくいきましたか？　実は、これはオリジナルのカーネルではうまくいきません。USAGI 版でもカーネルオプションの、

```
[*]     IPv6: allow default route
when forwarding is enabled
```

を有効にしてコンパイルしないと、IPv6のデフォルト経路を設定することができません。

この不可思議な仕様は、オリジナルカーネルのポリシーに起因します。それは、「IPv6ネットワークにおいて、ルータはその全経路を持つべきで、デフォルトルートというものは存在しない」というものです。このため、LinuxがIPv6ルータとして動作する場合、経路制御においてデフォルトルートは設定できないことになっています。

USAGI版カーネルでは上記オプションを用意して、その判断をユーザーに委ねています。どうしてもオリジナルのポリシーがいいという方は、以下のように疑似的にデフォルトルートを設定することも可能です。

```
# route -A inet6 add 3ffe::/16 gw
fe80::192.168.1.1 dev sit1
# route -A inet6 add 2000::/12 gw
fe80::192.168.1.1 dev sit1
```

ただしご覧のとおり、IPv6ネットワークで使用するプレフィクスが増えるたびに設定を見直す必要があります。

あとは、

```
# echo 1 > /proc/sys/net/ipv6/conf/
*/forwarding
```

としてやれば、Linuxはパケットをフォワードするようになります。

さて、次はいよいよRIPngを用いたルーティングについてです。ここではルーティングデーモンとしてZebraを使用します。ZebraはGPLで配布されているもので、もともとBGPというプロバイダ間で使用されるルーティングプロトコルをターゲットとして開発されました。しかし今では、OSPFやRIPなどのプロバイダ内部などで使用される代表的なプロトコルについても開発が進められ、現在に至っています。早くからIPv6への対応を進めていたのも、大きな特徴のひとつでしょう。

Zebraは面白い仕組みになっていて、動かしたいルーティングプロトコル別にデーモンが用意されています。

たとえばRIPngを動かしたければ、

```
# ripngd &
```

のように起動してやります。ただ、実際にパケットをフォワーディングさせるには、これだけではダメです。後述するRIPngの設定はもちろんですが、zebraというデーモンを立ち上げる必要があります。このプログラムを動かすと、RIPngによって学んだ経路ははじめてLinuxカーネルのルーティングテーブル（経路表）に反映されます。

```
# zebra &
```

実際の設定にはtelnetを使用する方法がお勧めです。

```
$ telnet ::1 ripngd
$ telnet ::1 zebra
```

設定方法は、IOSが搭載されているシスコ社のルータと変わりません。ここでは実際のコンフィグ情報のみを掲載しますので、コマンドラインの詳細についてはシスコルータの関連書籍を求めるか、Zebraについてくるマニュアルを参照してください。すぐれたCUIだと私は思います。さて実際のRIPngの設定についてですが、

```
access-list   local-only   permit
3ffe:XXXX:YYYY::/48
access-list local-only deny any
!
router ripng
 network sit1
 redistribute connected
 redistribute static
```

となります。いかがでしょうか？　設定を終えたら、RouterAへちゃんと経路が流れているか確認してもらいましょう。トンネルの先の上流ISPとは、ルーティングに静的経路制御を使うのか、あるいはRIPngを使用した経路制御を使うのかを事前に取り決めておく必要があります。

## 閑話休題

実は、前述したRAの設定にZebraを利用することも可能です。以下がその設定例です。

```
interface eth1
  ipv6   nd   prefix-advertisement
3ffe:XXXX:YYYY::/48
  ipv6 nd send-ra
```

“ipv6 nd send-ra”を打ち込むと、はじめてRAがリンク上に流れ始めます。

この記事を書いている6月現在、Zebraはradvdほど細かい設定はできません。これは、Zebraにradvdのすべての機能が実装されていないためですが、その解決にとりかかっている人がいますので、Zebraですべてが賄えるようになる日も近いでしょう。その際にはコマンドの文法が変更される可能性が大ですので、操作方法はマニュアルで確かめるようにしてください。

## 設定時のヒント

さて、これでIPv6ネットワークへ接続することができたはずです。意外に簡単でしたか？　骨の折れる作業だと嘆息されたかもしれません。ルータの設定は、ホストがケーブルを差しただけで通信できるようになるのとは、えらく違いがありますね。えてしてネットワーク管理者の苦労というものは、そんなものです。IPv6になって管理負担が減るようにと努力されてはいますが、IPv4のときと同様、すべてに手を抜けるほど簡単にはなりません。

あとは、普通にクライアントとして使うもよし、サーバを立ててみるもよし、お好きにどうぞ。……となるのですが、いくつか細かいTIPSについて触れておきたいと思います。まず、次のコマンドを実行してみてください。

```
$ netstat -A inet -a
```

せっかちな方は、

```
$ netstat -A inet -an
```

でしょうか？　ここでステータスが“LISTEN”となっているものは、皆さんのLinux君がIPv4でサービスしているサーバの一覧です[*3]。次にIPv6のサービス一覧を表示してみましょう。

```
$ netstat -A inet6 -a
```

どうでしょうか？　まったく表示されなかった人も多いと思います。USAGIプロジェクトのSnap Kitをインストールされた方は、ftpdが/usr/local/v6/sbinにインストールされていますので、試しにFTPサーバをIPv6でサービスしてみましょう。

その前にちょっぴり復習です。サーバとしてサービスを提供するには、デーモンとして常時立ち上げておく方法と、クライアントが接続してくるたびにinetd経由で該当プログラムを立ち上げる方法があります。後者はtcp_wrappersというセキュリティツールとともに使われることが多く、このツールでアクセス制限を設けることができます。ここではIPv6対応のinetdを使用し、そのinetd経由でftpdを立ち上げるまでをお見せします。IPv6対応のinetdとその設定ファイルはそれぞれ、

```
/usr/local/v6/sbin/inetd
/usr/local/v6/etc/inetd.conf
```

へインストールされています。サンプルファイルも同様に、

```
/usr/local/v6/etc/inetd.conf.sample
```

にあるので、そちらもご覧ください。

さて、**リスト1**を参照してください。この例では、IPv6用にUSAGIのftpdを、IPv4用にwu-ftpdを使っています。3列目の項目に注目してください。この項目には取り扱うTCPパケットを以下のように設定できます。

```
tcp  ：IPv4のみをLISTENする
tcp4：IPv4のみをLISTENする
tcp6：IPv6のみをLISTENする
tcp46：IPv4とIPv6の両方をLISTENする
```

このほか、UDPにも同様の設定があります。説明は正確さに欠ける気もしますが、実用上は問題ないでしょう。

次にinetdをIPv6対応版へと変更してみましょう。

```
# killall inetd
# /usr/local/v6/sbin/inetd
```

あとは、ちゃんとftpdがIPv4/IPv6の双方でLISTENしているか（クライアントからの接続を待っているか）、netstatを使って調べてみてください。どうでしょうか？　おそらく、多くの環境ではうまくいかなかったのではないでしょうか（オイオイ）。

/var/log/daemon.logあたりをチェックしていただければ、

```
Jun 18 16:38:16 linux inetd[1551]:
ftp/tcp: bind: Address already in
use
```

といったような一文が記録されていることと思います。これは何を意味して

**Glossary**

＊2　ここにある ::/0 は前述した未指定アドレスのことではありません。これはネットワークプレフィクスを表していて、全経路とマッチします。このためデフォルト経路の設定に用いられます。

＊3　意図せず動かしてしまっているサービスについてはちゃんと停止しておくことをお勧めします。特に会社でお使いで、NATの中だから大丈夫だと思っている方（これもNATの弊害のひとつといえるのかもしれません）、クラッキングの多くが内部からの仕業だという報告もあります。

▼リスト1　inetdを介してIPv6／IPv4用のFTPデーモンを起動する

```
/usr/local/v6/etc/inetd.conf
ftp     stream  tcp6    nowait  root    /usr/local/v6/sbin/tcpd
/usr/local/v6/sbin/in.ftpd
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/wu-ftpd -l
```

いるのでしょうか？　実は、Linuxでは、一度IPv4で開いたポートを再度IPv6から開くことができません。これは、上記のようにIPv4とIPv6とで別のプログラムを動作させることができないことを意味します。じゃあ、IPv6のFTPサーバを立ち上げたら、IPv4のサービスを提供できないのか？　と思ってしまいますが、それは違います。

実は、本来のLinuxではtcp6の設定はできないのです。許されているのはtcp46で、IPv6対応のサーバを立ち上げたときは、そのサーバが前述したIPv4射影アドレスを用いてIPv4もIPv6も面倒を見ることになっています。

ここにIPv4射影アドレスの落し穴が潜んでいるのですが、それはtcp_wrappersの説明のところで述べたいと思います。さて、USAGI版のカーネルでは、

```
[*]          IPv6:  allow  binding
ipv6/ipv4 sockets on the same port
```

を有効にしてカーネルをコンパイルしていれば、同じポートに対してtcp4とtcp6を設定することができます（今回の例でいえばFTP）。

このオプションが有効でなくとも、IPv4でサービスが立ち上がっていなければ、tcp6を設定することは可能です。これは、オリジナルカーネルにはない機能です（IPV6_V6ONLYソケットオプションの実装により実現されています）。

## tcp_wrappersでアクセス制限

さて、tcp_wrappersによるアクセス制御を実施するには、/etc/hosts.[allow｜deny]に制限のルールを記述します。その中身は、

```
ALL : LOCAL
ftp : 192.168.1.0/255.255.255.0
```

といったものになります。コントロールしたいサービス名と適応させたい条件（多くはアドレスやドメイン名）を“：”で区切って記述します。しかし“：”はすでにIPv6アドレスの表記方法として使用されていますから、USAGI Snap Kitの tcp_wrappersでは、次のように記述することになっています。

```
ALL : LOCAL
ftp  :  192.168.1.0/255.255.255.0
[3ffe:XXXX:YYYY::]/48
```

これで万事OK……ちょっと待ってください。もしこのftpdがtcp46で動作していたらどうなるのでしょうか？

IPv4射影アドレスの説明でもあったように、IPv6のノードはIPv4からのアクセスに対処するため、内部ではIPv4射影アドレスというIPv6アドレス体系に変換して取り扱っています。そのため、もし192.168.1.1からアクセスした場合には、IPv6アプリケーションにとっては::ffff:192.168.0.1からのアクセスに見えます。そのとき、上記のような条件にマッチするでしょうか？　それはアプリケーションの実装に依存します。もしかしたら、IPv4射影アドレスにも考慮してマッチするようになっているかもしれないし、そうではないかもしれません。そういう意味で、思わぬセキュリティホールになる可能性もありますので、IPv6のデーモンがIPv4／IPv6の両方をサービスしている際には注意が必要です。

さて、ではUSAGIについてくるtcp_wrappersはその点を考慮して動くでしょうか？　それとも別途IPv4射影アドレスにマッチする条件を追加してやる必要があるのでしょうか？　もちろん、ちゃんと考慮しています。

## おわりに

かなり駆け足になってしまったのではないかと危惧しているのですが、いかがでしたか？　今回は書きたいことが多かったのに、その割には十分な時間がとれず、悔いの残るものとなってしまいました。なんにせよ、IPv6を使ってはみたいけど、何から始めたものやら……という方に少しでも役立ったとしたら、非常にうれしく思います。

とりあえず、IPv6のお話はこれでひと段落です。IPv4と同じような使い方をするのであれば、IPv6はすでに十分その任をはたせるところまで来ていると思います。

もともと私はISPにいたのですが、IPv4アドレスの枯渇問題にはうんざりさせられます。アドレスを余剰に持っているところからの回収なんて夢のまた夢。アドレスが足りる、足りない、なんてくだらないことに頭を悩ませなければならないのが現状です。とりあえずIPv6の実用化が見えてきたということで、そういったことからネットワークエンジニアは開放されそうです。とはいえ、IPv6にはIPv6の悩みがあり、「IPv6なんて大嫌いだぁ～～」と、ぼやく日々がやってくるかもしれません。それでも、どうにもならないアドレス不足に悩むよりはよっぽど建設的な仕事となるでしょう。

これまでの説明でもわかるとおり、LinuxのIPv6スタックってまだまだです。とりあえず、先にゴールしつつあるKAMEに追い付かねば。この記事を読んでいるそこの方、USAGIと一緒に、あの丘まで登ってみませんか？

## Column

### アドレスのまとめ

IPv4アドレスの先頭の数ビットを見て、そのアドレスがクラスAのアドレスなのか、それともクラスCのアドレスなのかがわかったように、IPv6アドレスも、その先頭の数ビットを見ることによって、どのような種類のアドレスかがある程度判別できます。その代表的なものを表2にまとめておきます。

ifconfigを実行すると、アドレスの右側の表示でスコープがわかるようになっていますが（図6）、これにも先頭数ビットを使った判別方法が用いられています。

表2にあるように、001から始まるアドレスは前回に説明したTLAやNLAといった階層構造を持つ集約可能グローバルユニキャストアドレスです。

また、マルチキャストアドレス（図7）については表3に挙げるようなものを知っておくと便利でしょう。これからもわかるように、IPv4でのブロードキャストアドレスは、IPv6ではマルチキャストアドレスがその機能を内包しています。

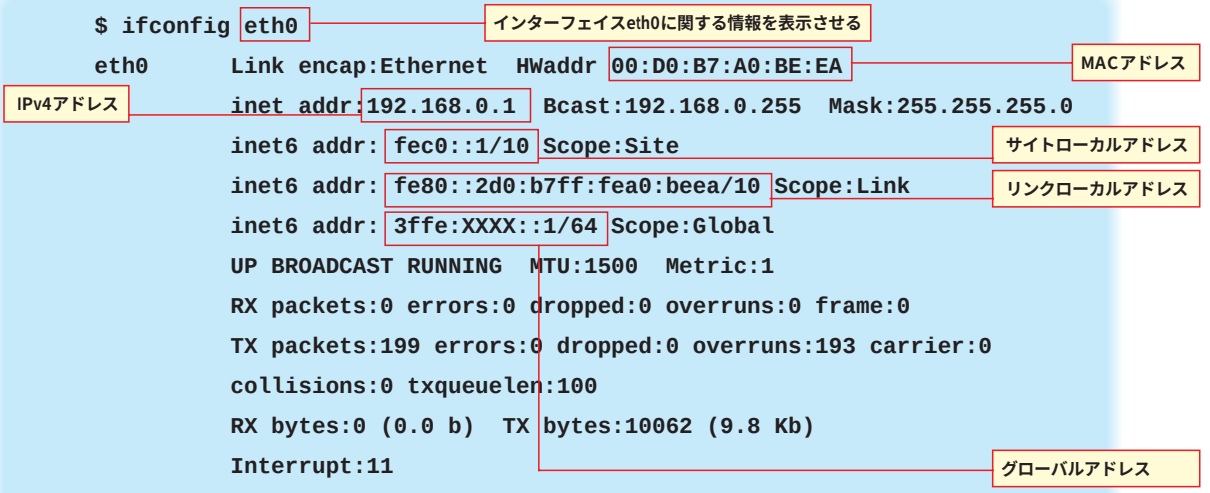



図6　ifconfigのアドレス表示

| 割り当て | プレフィクス 2進数 | 16進数 | 例 |
|---|---|---|---|
| 未指定、ループバックなどの特殊なアドレス | 0000 0000 | 00 | ::1、::ffff:192.168.1.2 |
| 集約可能グローバルユニキャストアドレス | 001 | 2〜3 | 2001:XXXX::1、3ffe:XXXX::1 |
| リンクローカルユニキャストアドレス | 1111 1110 10 | FE8〜（FEB） | fe80::2d0:b7ff:fea0:beea |
| サイトローカルユニキャストアドレス | 1111 1110 11 | FEC〜（FEF） | fec0:XXXX::1 |
| マルチキャストアドレス | 1111 1111 | FF | ff02::1 |

表2　アドレスの割り当て

| アドレス | 説明 |
|---|---|
| ff02::1 | リンク上に接続された全ノード宛て |
| ff02::2 | リンク上に接続された全ルータ宛て |
| ff02::3 | リンク上に接続された全ホスト宛て |

表3 よく使われるマルチキャストアドレス

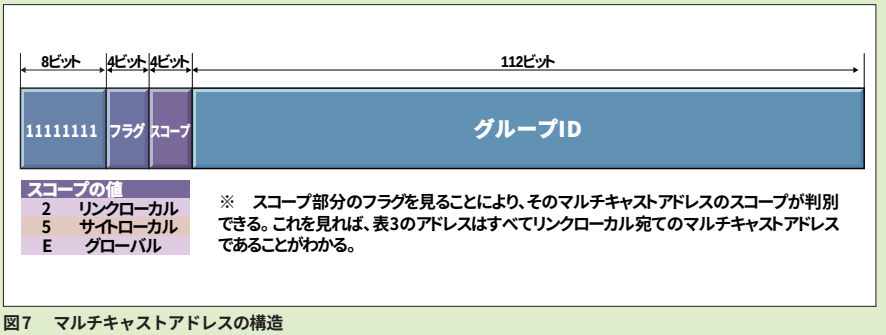



図7　マルチキャストアドレスの構造

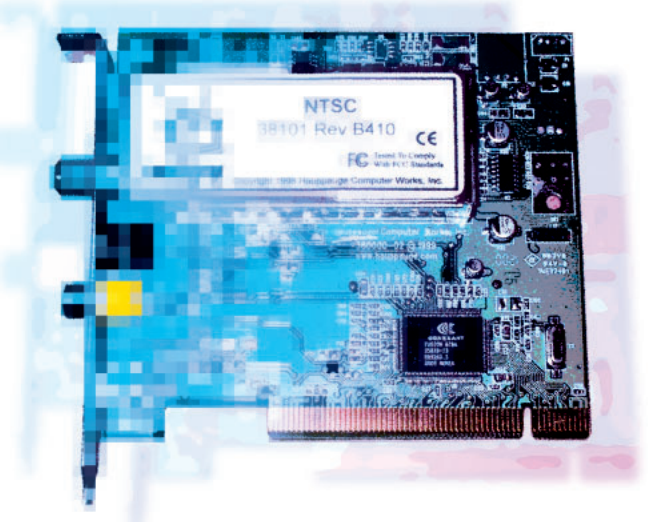
短期集中連載

## Linuxでテレビ録画に挑戦！

# 1万円で作るテレビ自動録画サーバ

## 最終回　MPEG-1形式でキャプチャする

**前回、前々回と立て続けに「公約」を果たすことができず、読者のみなさんの期待を裏切るようなことになってしまった。あらためてお詫びしたい。しかし、Linuxでのマルチメディア、とくにビデオ関連のデバイスドライバやアプリケーションは、さまざまな要素が関連するためとくに設定が難しいのは事実である。今回は、連載の最終回として、「mp1e」をインストールして、MPEG-1形式でのリアルタイムキャプチャを行う方法を紹介しよう。**




文：竹田善太郎
*Text : Zentaro Takeda*

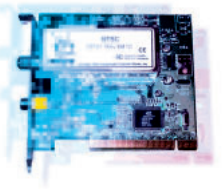
ビデオデータは、音声データや静止画像データと比べて、桁違いに大量の記憶領域を必要とする。このため、キャプチャカードなどを使って録画したデータを、なんの加工もせずにそのままの状態で保存しようとすると、たとえ100Gバイト単位の大容量ハードディスクがあったとしても、わずか2〜3時間分のデータしか保存できないということになってしまう。このため、何らかの方法でデータを圧縮しなければならない。

この連載でいままで紹介してきたテレビ録画システムでは、「Motion JPEG」（mjpeg）という形式で録画データを圧縮してきた。これは、xawtvに付属のstreamerコマンドが標準で対応している唯一の圧縮形式なのだが、動画の圧縮方式としてはかなり「原始的」なものなので、圧縮の程度を高くしてファイルサイズを小さくするのには限界がある。

ビデオデータの圧縮方法として、現在もっとも普及しているのが「MPEG」形式である。ここでMPEGについて詳しく説明することはできないが、現在利用されているMPEG形式には、「MPEG-1」、「MPEG-2」、「MPEG-4」の3種類がある。このうち、MPEG-1形式はビデオCDや一部のハードディスクレコーディング機器などに、MPEG-2はDVDビデオやデジタル衛星放送などで、MPEG-4はインターネットなどのネットワーク経由でのビデオ画像配信などに利用されている。

パソコンで使えるビデオキャプチャカードの中には、MPEG-1やMPEG-2形式のデータ圧縮を行う専用のハードウェアを内蔵しているものがある。このような製品を利用すれば、特別なソフトウェアなしでもビデオデータを圧縮できるはずなのだが、現実には、ハードウェア圧縮機能を利用するためには、専用のソフトウェアが必要になる。現在、ハードウェアMPEG圧縮機能をLinuxで利用できるようにする試みが、いくつかのグループでなされているようだが、MPEG圧縮ハードウェアの仕様は一般には公開されておらず、また内部仕様の利用にあたってはライセンス契約が必要になるものがほとんどなため、フリーで利用できるドライバ類が公開される可能性はかなり低い。現在、ごく一部のMPEG対応キャプチャカードについて、ベータ版のLinux用ドライバが無償公開されているが、正式版では有償で配布されることが前提となっているようだ。

一方、MPEG圧縮機能を備えていないキャプチャカードを利用する場合は、MPEG圧縮をソフトウェアで行うことになる。このような圧縮を行うソフトウェアで、現在Linuxで利用できるものについては、前回も説明したとおり、「mpeg_encode」や「mp1e」などがある。このうち、mp1eは、ビデオデバイスから読み込んだビデオデータと音声デー



タを、リアルタイムでMPEG-1形式にエンコードできる。そこで、今までに構築してきたテレビ自動録画システムを一部拡張して、mp1eを利用して録画データをMPEG-1形式で保存できるようにしてみる。

### ●mp1e 1.9pre

前回、mp1eは作者によるメンテナンスが継続されているかどうか不明であると述べた。ところが、前号の原稿締め切り直後に、最新のバージョン「1.9pre」のソースファイルとバイナリファイル（RPM形式）が公開されるようになった。それまでに入手可能だったバージョン1.8.1のmp1eは、コンパイル作業がかなり面倒で、とくに最新のディストリビューションで利用しようとすると、ソースファイルの修正やライブラリの変更などの作業が必要だったのだが、バイナリファイルが利用できるようになったことで、MPEG-1キャプチャの「壁」はかなり低くなったのだ。

ただし、Linux側のビデオキャプチャドライバを「Video for Linux 2」に更新しないと、mp1eを利用できないことが多い。また、Red Hat Linux 7.1の場合、配布されているxawtvのバイナリは正しく動作しない。このため、カーネルの再構築やデバイスドライバの入れ替え、xawtvプログラムの修正やコンパイルなどの作業は必要になる。このような作業に自信がない場合は、残念ながら、MPEG形式による録画はあきらめるしかない。

また、本記事の手順どおりに作業をしても、うまく動作する保証はない。利用しているマシンの構成、とくにサウンドカードやキャプチャカードの機種、ディストリビューションの種類、カーネルの構成の違いなどで、まったく異なる結果になるからだ。本記事中では、Video for Linux 2について、ソースコードを修正している部分もあるが、これらはすべて「その場しのぎ」の対処でしかないため、すべての環境で有効とは限らない。これらの点について十分ご理解いただいたうえで、各自の責任で挑戦してみてほしい。

なお、今回はRed Hat Linux 7.1（カーネル2.4.2）の環境を例にしているが、前回までのTurbolinux 6.0（カーネル2.2）の場合も、ほぼ同様の手順で構築できる。カーネル2.2環境での注意点については、別途コラムで解説する。

## インストールの流れ

mp1eをインストールしてMPEG-1形式でのキャプチャを可能にするには、大まかに次のような一連の作業が必要になる。


1. Video for Linux 2のインストール
   - 1.1. カーネルの再構築
   - 1.2. GNUコンパイラのダウングレード
   - 1.3. ソースファイルの修正
   - 1.4. コンパイルとインストール
2. mp1e-1.9preのインストール
3. xawtvの再インストール
4. tvcaptureスクリプトの修正

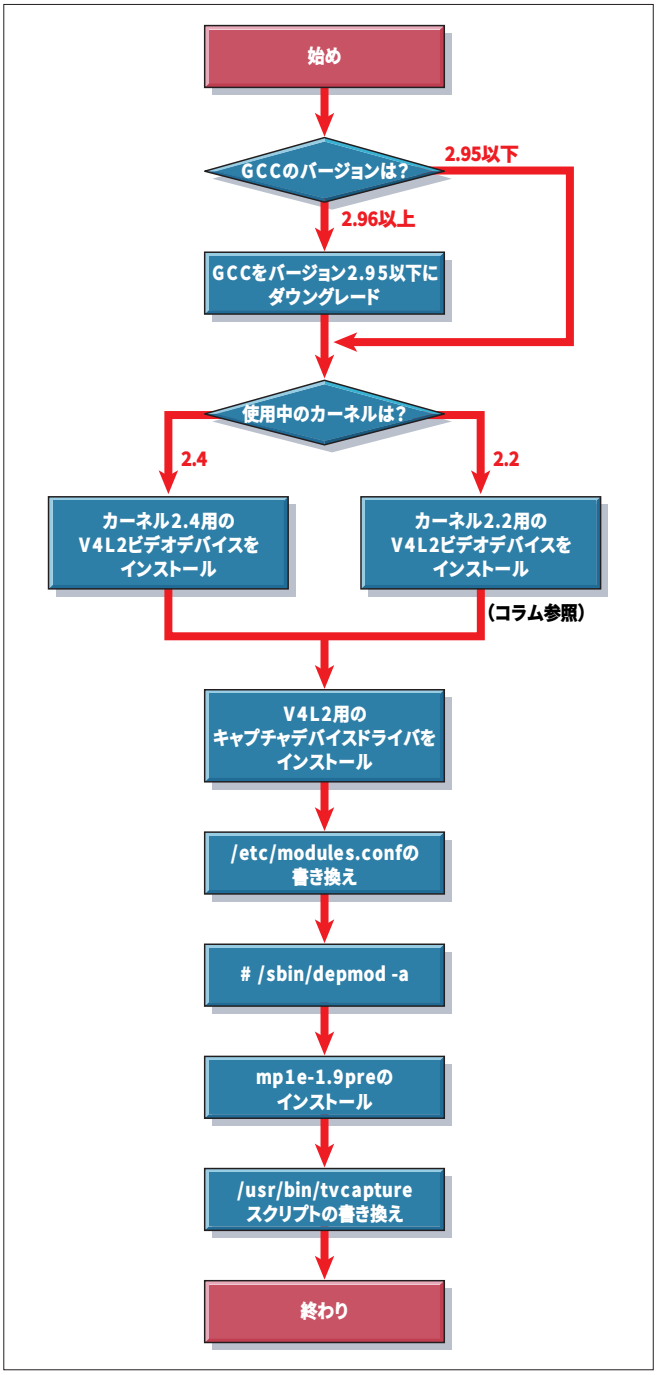




図1　mp1eインストールまでの流れ

これらのうち、場合によっては必要のない作業もあるのだが、どのような手順で進めればよいのかを、フローチャートにまとめてみたので参考にしてもらいたい（**図1**）。実際には、それぞれのステップを実行してみて、エラーなどが発生するたびに、それに対処することになる。

## Video for Linux 2のコンパイルとインストール

mp1eは、ビデオキャプチャ処理を行う場合に、Video for LinuxのAPIを利用している。すなわち、いままでに利用してきたbttv.oなどのドライバ類を使っているわけだ。このVideo for Linuxには、現在、オリジナルのVideo for Linux（便宜的に「Video for Linux 1」と呼んで、記事中では「V4L1」と略すことにする）と、これを拡張した「Video for Linux 2」（同じく、「V4L2」と略す）の2つのバージョンが存在する。V4L2の全体的な仕様は、Bill Dirks氏が開発し、サンプルのドライバなどを公開している（**画面1**、http://www.thedirks.org/v4l2/）。BTTVチップ用のV4L2ドライバは、Justin Schoeman氏が主催するプロジェクトで開発され、http://bttv-v4l2.sourceforge.net/のWebページ（**画面2**）で公開されている。今回は、このV4L2ドライバを利用する。

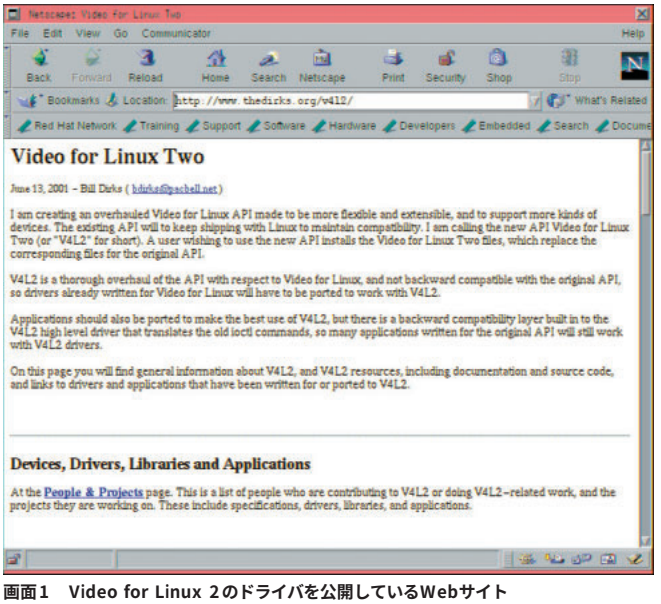
V4L2の仕様はV4L1の上位互換になっており、mp1eやxawtvなどのV4Lアプリケーションのほとんどは、どちらの環境でも動作するように作られていることが多いのだが、筆者の環境で試した限り、mp1eの最新バージョン「1.9pre」はV4L2の環境でしかうまく動作しなかった。もちろん、マシンの構成や利用するLinuxディストリビューションによっては、V4L1の環境でも問題がない場合もあると思うが、その場合でも、カーネルのバージョンによっては、V4L1のドライバを最新のものに入れ換えないと、アプリケーションが動作しないことがある。結局、同じ手間をかけるなら、V4L2をインストールしてしまったほうがよいだろう。

ちなみに、現在のLinuxカーネルには、bttv.oなどのV4Lドライバが取り込まれており、多くのディストリビューションでは、デフォルトの状態でV4Lが利用できるようになっているが、カーネルに組み込まれているV4LドライバはV4L1で、しかもかなり古いバージョンである（カーネル2.4でも事情は変わらない）。

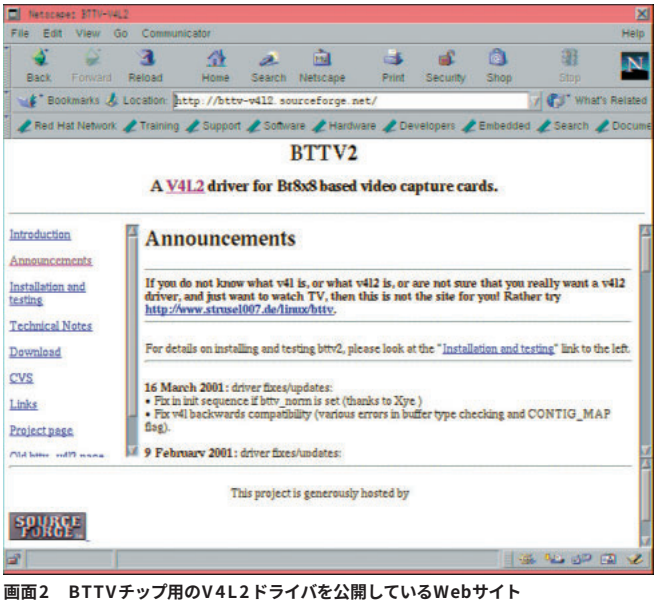
### ●カーネルの再構築

理屈のうえでは、V4L2のドライバをインストールするだけなら、カーネルの再構築などの作業を行う必要はないのだが、現在のLinuxディストリビューションの多くは、インストールしたばかりの状態では、カーネルのソースツリーの状態が、実行中のカーネルの構成と一致していないため、ドライバ類をコンパイルするのに支障がある。そこで、ソースツリーの状態を整えるために、カーネルの再構築の作業が必要になってくるのだ。

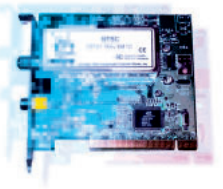
カーネル再構築の手順について、ここですべてを説明する




画面1　Video for Linux 2のドライバを公開しているWebサイト（http://www.thedirks.org/v4l2/）
カーネル2.2用と2.4用のビデオデバイスドライバは別なので注意する。各種ビデオキャプチャカード用のドライバについては、画面2のサイトから入手する。




画面2　BTTVチップ用のV4L2ドライバを公開しているWebサイト（http://bttv-v4l2.sourceforge.net/）
キャプチャカードドライバはカーネル2.2と2.4とも同じである。



余地はない。各自、書籍などを参考にしてほしいが、二、三、注意すべき点をあげておきたい。

- **make xconfig（あるいはmake menuconfigなど）を実行して、カーネルに組み込む機能を設定する。**
- **「I2C Support」（Character devicesメニューの下にある）を「Y」または「M」に設定し、自分のマシン環境に応じてI2Cインターフェイスの設定を有効にしておく（自信がない場合には、すべてを「M」にしておけばよい）。さらに、「Hardware sensors support」の項目の「Hardware sensors support」を「M」に、「Other I2C devices」を「Y」に、「Brooktree BT869 Video Modulator」を「M」に設定する。**
- **「Video For Linux」（Multimedia devicesメニューの下）を有効（Y）にして、「Video Adapters」の項目中で、利用するビデオキャプチャデバイス（本連載では「Bt848 Video For Linux」）の項目を「M」に設定する。なお、前述の「I2C Support」が有効な状態になっていないと、Video For Linuxの項目は選択できないので注意すること。**
- **「Sound」の項目を、自分のサウンドカードの利用形態に応じて設定しておく。**
- **その他の項目については、各自のマシン環境などに応じて適宜設定する。**
- **再構築したカーネルをインストールして使用する必要はないが、ディストリビューションのデフォルトのカーネルと再構築したカーネルの構成が著しく異なっていると、アプリケーションのコンパイルができないこともあるので、そのような場合には再構築したカーネルでシステムを起動する。**

make xconfigの設定が終わったら、

```
# make depend
# make clean
# make bzImage
# make modules
```

をそれぞれ実行する。必要なら、

```
# make install
# make modules_install
```

も実行する。

### ●サウンド環境の確認

インストール作業を進める前に、ここで、使用中のLinuxマシンのサウンドドライバの状態を確認しておこう。当たり前のようだが「サウンドドライバが本当に正しく動作しているかどうか」を確認する。サウンドの再生が問題なくできていても、録音機能がまったく働いていないことがまれにあるからだ。このような場合、mp1eのインストールがうまくいっても、いざ録画を行おうとすると、プログラムがハングアップしたり、録画したMPEGファイルに音声がまったく入っていないといったことになる。

### ●GNUコンパイラのダウングレード

これは、まったく頭の痛い問題なのだが、GNUのコンパイラにはバージョン間の互換性の問題が多く存在する。V4L2ドライバをコンパイルする場合も、GNUコンパイラのバージョンに注意が必要だ。とくに、Red Hat Linuxを利用している場合、コンパイラのバージョンを古いものに「ダウングレード」しないと、ドライバのコンパイルは不可能である。

Red Hat Linux 7.1にバンドルされている「gcc-2.96」は、GNUコンパイラの正式リリースではなく、「gcc-3.0のスナップショット」という位置づけになっている（http://gcc.gnu.org/gcc-2.96.htmlを参照のこと）。このため、それ以前の最終の正式リリースである「gcc-2.95」との互換性はなく、アプリケーションのコンパイルなどに支障が起こる可能性があるのだ。実際、V4L2ドライバのソースファイルは、gcc-2.96ではコンパイルできない。このため、gcc-2.95にダウングレードする必要がある。

コンパイラのダウングレードは次のような手順で行う。

**1. http://www.rpmfind.net/などで、gcc-2.95のRPMファイルを検索し、ダウンロードする。**

（本記事のサンプルでは、Polish(ed) Linux Distributionのgcc-2.95.3-27.i686.rpmを利用している）。

**2. すでにインストールされているgcc関連のパッケージをアンインストールする。**

まず、gccをアンインストールするコマンドを実行する。

```
# rpm -e gcc
```

すると、gccに依存しているパッケージ（gccを利用しているFORTRAN、C++などのコンパイラ類）が表示されるので、それらをすべてアンインストールしてから、改めてgcc

のパッケージをアンインストールする。

### 3. cppのパッケージをダウングレードする。

GNUコンパイラのプリプロセッサ（cpp）のバージョンを、コンパイラのバージョンに合わせる必要があるので、cpp-2.95.3-27.i386.rpmをダウンロードして、インストールする。このとき、依存関係のエラーメッセージが出てインストールできないことがあるが、無視して強制インストールする（rpmコマンドの-Uオプションは「アップグレード」の意味だが、ここではダウングレードになる）。

```
# rpm -U --force cpp-2.95.3-27.i386.rpm
```

### 4. gccのパッケージをインストールする

```
# rpm -i gcc-2.95.3-27.i686.rpm
```

以上で、GNUコンパイラのダウングレードは完了である。もし、FortranやC++などのコンパイラも必要なら、同じバージョン系列のコンパイラのRPMファイルをダウンロードして、インストールする。

#### ●V4L2ソースファイルの修正とコンパイル

次に、V4L2のインストール作業に移る。まず、http://bttv-v4l2.sourceforge.net/などから、V4L2関連のソースファイルを入手する。このソースファイルは、カーネルのバージョンが2.2の場合と2.4の場合で、ダウンロードすべきファイルが異なるので注意する。カーネルのバージョンが2.2系の場合は、前回説明したように、「videodev20010302.tgz」（ファイル名後半の日付の部分はバージョンによって変わる）と「driver-20000804.tgz」（同上）だが、カーネルが2.4系の場合は、「videodevX-20010613.tgz」というファイルと、カーネル2.4用の「driver-20010316.tgz」の2つのアーカイブを入手する。

まず、videodevX-xxxxx.tgzのアーカイブを、適当な作業用ディレクトリに展開する。

```
$ tar xzf videodevX-20010613.tgz
```

すると、videodevXという名前のディレクトリが作成され、その中にソースファイルが展開される。新しく作成されたvideodevXに移動して、

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

を実行すればvideodevXのコンパイルとインストールは問題なく終わるはずだ。ここでエラーが出る場合は、前述のカーネルの再構築作業がうまくいっていない可能性がある。

次に、driver-xxxxx.tgzのアーカイブを、別の作業用ディレクトリに展開する。

```
$ tar xzf driver-20010316.tgz
```

これも、本来ならば「make; make install」でインスト-

▼リスト1　bttv.hファイルの修正部分（赤枠で囲んだ部分を追加する）

**ファイルの末尾にある以下の部分を修正する**

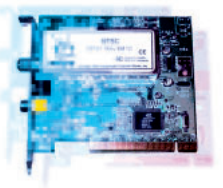
```
/* i2c */
#define I2C_CLIENTS_MAX 8
extern struct i2c_algo_bit_data bttv_i2c_algo_template;
extern struct i2c_adapter bttv_i2c_adap_template;
extern struct i2c_client bttv_i2c_client_template;
extern void bttv_bit_setscl(void *data, int state);
extern void bttv_bit_setsda(void *data, int state);
extern void bttv_call_i2c_clients(struct bttv *btv,
unsigned int cmd, void *arg);
extern int bttv_I2CRead(struct bttv *btv, unsigned char
addr, char *probe_for);
extern int bttv_I2CWrite(struct bttv *btv, unsigned char
addr, unsigned char b1,
             unsigned char b2, int both);
extern void bttv_readee(struct bttv *btv, unsigned char
*eedata, int addr);

#endif /* _BTTV_H_ */
```

▼リスト2　/etc/modules.confの変更部分（Red Hat 7.1の場合）

**赤枠で囲んだ部分を変更する**

```
alias eth0 eepro100
alias parport_lowlevel parport_pc
alias sound-slot-0 es1371
post-install sound-slot-0 /bin/aumix-minimal -f
/etc/.aumixrc -L >/dev/null 2>&1 || :
pre-remove sound-slot-0 /bin/aumix-minimal -f
/etc/.aumixrc -S >/dev/null 2>&1 || :
alias char-major-81 bttv2
alias usb-controller usb-uhci
```



ルできるはずなのだが、Red Hat 7.1の環境ではコンパイルエラーが発生してインストールできなかった。そこで、展開されたファイルの中の「bttv.h」というファイルを、リスト1のように修正する。あとは、

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

を実行すればインストールできるはずだ。Red Hat 7.1の場合make installを実行した際に、「depmod: コマンドが見つかりませんでした」というエラーが表示されるが、気にせずに次のコマンドを手動で実行すればよい。

```
# /sbin/depmod -a
```

次に、/etc/modules.confファイルを編集して、bttvドライバのロードを指定している部分を、bttv2に変更して（**リスト2**）、改めて「/sbin/depmod -a」を実行しておく。

以上で、V4L2のインストールは完了である。念のため、システムを再起動してから次の作業に進む。

## xawtvの再インストール

Red Hat 7.1のパッケージ版では、xawtvはデフォルトではインストールされない。「Power Tools」という名前のCD-ROMに、RPMファイルの形で収録されている。ところが、このRPMをインストールしても、テレビの受信をすることはできるのだが、v4lctlコマンドを使ってチャンネルを切り替えるなどの操作ができない（セグメンテーション違反のエラーが発生する）。このため、xawtvをソースファイルからコンパイルしてインストールする必要がある。

xawtvのソースファイルは、http://bytesex.org/xawtv/index.htmlから入手できる（**画面3**）。現時点での最新バージョンは3.53なので、このWebページからxawtv_3.53.tar.gzファイルをダウンロードする。そして、適当な作業用ディレクトリを作成して、アーカイブを展開する。

```
$ tar xzf xawtv_3.53.tar.gz
```

xawtv-3.53という名前のディレクトリ中にファイルが展開されるので、このディレクトリに移動して、次のコマンドを実行する。

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

なんの問題もなくコンパイルが完了してインストールされるはずだが、エラーが発生する場合は、カーネルの再構築やV4L2のインストールからやり直してみる。

Red Hat 7.1の場合、xawtvを自分でコンパイルしてインストールした場合、ビデオキャプチャデバイスのデバイスファイルの名前に注意が必要になる。xawtvは、デフォルトの状態では、キャプチャデバイスのデバイスファイルが「/dev/video」であることを想定しているのだが、Red Hat 7.1では「/dev/video0」がキャプチャデバイスのデバイスファイルになっている。このため、xawtvコマンドやv4lctlコマンドを実行する場合、次のように、デバイスファイル名を指定してやる必要がある。

```
$ xawtv -c /dev/video0
$ v4lctl -c /dev/video0 setstation ......
```

後述するtvcaptureスクリプトの修正の際には、この点にも注意を払うこと。


**画面3　xawtvのソースファイルを公開しているWebページ**
**バージョンアップは頻繁に行われているようだ。**

## mp1eのインストール

前述したように、今回はバイナリがリリースされたばかりのmp1eの最新バージョン「1.9pre」を利用することにした。しかし、これは正式バージョンではないため、将来は公開が取りやめになる可能性もある。また、バイナリファイルなので、Linuxの環境によってはうまく動作しない場合も考えられる。このため、現時点での最終正式バージョンである「1.8.1」のソースファイルをコンパイルする方法についても、**118〜119ページのコラム**にまとめておく。

バイナリファイル（RPMファイル）からのインストールは簡単である。RPMファイルをhttp://prdownloads.sourceforge.net/zapping/mp1e-1.9pre1-1.i386.rpmからダウンロードしたあとに、

```
# rpm -i mp1e-1.9pre1-1.i386.rpm
```

を実行するだけでよい。これで、/usr/binディレクトリにmp1eコマンドがインストールされるはずだ。

なお、このRPMファイルのmp1eコマンドも、先ほどのxawtvと同様に、ビデオキャプチャデバイスのデバイスファイルが/dev/videoであることを前提としているので、Red Hat 7.1の環境でキャプチャを行うときには、「-c /dev/video0」オプションを追加して、ビデオデバイスファイルの名前を指定する必要がある。

### ●mp1eのコマンドラインオプション

ここで、mp1eコマンドの使い方を簡単に説明しておこう。mp1eでは、録画するデータのビットレート、フレームレート、キャプチャするフレームの数などをオプションで指定することができる。たとえば、ビットレート1.5Mbps、フレームレート24fps、フレーム数1000でキャプチャを行う場合、次のように実行する。

```
$ mp1e -c /dev/video0 -b 1.5M -f 24 -n 1000 > capture.mpeg
```

なお、mp1eコマンドではキャプチャ結果のデータは標準出力に出力されるので、リダイレクトでファイルに保存する必要がある。

mp1eコマンドのオプションは、

```
$ mp1e --help
```

を実行すれば表示される。


**図2　自動録画システムの概念図**
最終的な録画処理は、tvcaptureスクリプトから呼び出されるキャプチャプログラムが行う。いままでは、キャプチャプログラムとして「streamer」を使っていたが、これを「mp1e」を呼び出すように変更する。

▼リスト3　tvcaptureスクリプト（mp1eバージョン）

ファイルの末尾にある赤枠で囲んだ部分を変更する

```
#!/usr/bin/perl

# tvcapture -- capture TV program using streamer
#
# usage : tvcapture -p <program_name> -l <length_in_min>
-c <Channel_Name> -d <directory>
#
# Copyright (c) 2001 by Zentaro Takeda
# Personal use Only

require "getopt.pl";

$capture_program = "/usr/bin/mp1e";
$v4lctl = "/usr/local/bin/v4lctl";
$bash = "/bin/bash";

$fps = 15; # frames per second
$bitrate = "0.5M"; # mpeg bitrate
$capture_device = "/dev/video0"; # capture device file

$streamer_opt = "-b $bitrate -f $fps -c $capture_device
";

# channel name must be same as defined in  ~/.xawtv
%channel = ("NHK"=> 1,
            "NHK-EDU"=> 3,
            "NTV"=> 4,
            "TBS"=> 6,
            "FUJI-TV"=> 8,
            "TV-ASAHI"=> 10,
            "U-AIR"=> 11,
            "TV-TOKYO"=> 12);

&Getopt('plcd');

if ($opt_p =~ /[^A-Z]/ ) {
  $program_name = "PROGRAM";
} else {
  $program_name = $opt_p;
}

                    <中略>

# get current date and time
($sec, $min, $hour, $mday, $mon, $year, $wday, $yday,
$isdst) = localtime();

$year = $year + 1900 ; # y2k correction
$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 = $dirname . "/" . $program_name . "_" .
$timestring . "_" . $channel_name . ".mpeg";

$commandline = "$v4lctl -c $capture_device setstation
$channel_name 2\>/dev/null; $capture_program -n $flames
$streamer_opt \> $recfile \&";

exec $commandline;
```

## tvcaptureスクリプトの修正

ここで、本連載で作成しているテレビ自動録画システムの概念図をもう一度見てもらいたい（**図2**）。この構成では、実際のビデオキャプチャ処理を行っているのは、tvcaptureスクリプトであり、このスクリプトからstreamerコマンド（xawtvに付属のキャプチャツール）を呼び出している。mp1eを使って、MPEG-1形式でのキャプチャを行うようにするには、このtvcaptureスクリプトを書き換えて、streamerコマンドの代わりにmp1eコマンドを呼び出すように改造してやればよい。ほかのスクリプト類や設定ファイルは、一切書き換える必要はない。

mp1eを使うように変更したtvcaptureスクリプトを**リスト3**に掲載する。赤い枠で囲んだ部分が、今回変更した箇所である。mp1eを呼び出すようにした以外に、前述のデバイスファイル名の問題に対処するための変更も加えている。また、mp1eはフレームレートだけでなくビットレートも設定できるようになっているので、スクリプト中でビットレートの値を設定できるようにしている。これらの値を適当に調整することで、自分の目的にもっとも適した画質でキャプチャできるようにしてもらいたい。

## MPEG-1ファイルの再生について

MPEG-1形式でキャプチャしたデータは、WindowsマシンのMedia Playerなどで再生することができる（**画面4**）。

Linux用のMPEG-1再生ソフトウェアにはさまざまな種類


画面4　キャプチャしたmpegファイルをWindowsマシンで再生する

があるので、ここですべてを紹介することはできないが、現在もっともよく使われているのは、MP3プレーヤとしても利用者の多い「XMMS」（X Multimedia System）だろう。最近のXMMSで、入力プラグインとして「smpeg-xmms 0.3.3」（libsmpeg_xmms.so）がインストールされていれば（**画面5**）、MPEG-1のビデオファイルを再生することができる（**画面6**）。

## キャプチャ条件の設定

xawtv付属のstreamerコマンドを使ってキャプチャを行う場合と同様に、mp1eを使う場合も、マシンの性能などに応じて、フレームレートやビットレートを自分で調整する必要がある。mp1eの場合、フレームレートを指定しないと、ビットレートとマシン性能に応じて、最適なフレームレートが選択されることになっているのだが、実際には、コマ落ちが発生したり、音声との同期がとれなくなったりすることが多いので、明示的にフレームレートを指定する必要がある。

録画したデータをハードディスクに保存して、番組ライブラリを作成したいような場合には、利用可能なハードディスクの容量に応じて、ビットレートを決め、次に、それに見合ったフレームレートを決めるのが現実的だろう。保存する番組の内容、たとえば、高画質で保存する必要のある番組なのかどうか、動きの速い画像が多いかどうかによって、ビット

### Column

#### カーネル2.2環境でのV4L2のインストール

V4L2のドライバは、カーネルのバージョンによってインストールするファイルやインストールの方法が異なる。前回も簡単に説明したのだが、インストールの手順をもう少し詳しく解説しておこう。

**1. ソースファイルの入手**

V4L2のドライバは、ビデオデバイスドライバとキャプチャカードドライバの2つの部分に分かれている。このうち、ビデオデバイスドライバについては、使用中のキャプチャカードの機種によらず、同じものをインストールする。キャプチャカードドライバについては、キャプチャカードの機種（グラバチップの種類）によって、インストールすべきファイルが異なるので注意する。本連載では、BTTVシリーズのチップを使ったカードを前提としているので、BTTV用のキャプチャカードドライバを入手することになる。

V4L2のビデオデバイスドライバは、Webページhttp://www.thedirks.org/v4l2/から入手する（ファイル名は、videodev20010302.tgz）。ここには、カーネル2.2用のソースとカーネル2.4用のソースがあるので、間違えないようにする。このWebページでは、インストールの手順についても簡単に解説されているので、参考になるだろう。

キャプチャカードドライバについては、カーネル2.2と2.4で取得するファイルに違いはない。http://bttv-v4l2.sourceforge.net/から、driver-20000804.tgzというファイルを取得する。

**2. ビデオデバイスドライバのインストール**

まず、videodev20010302.tgzファイルを展開する。このファイルは、カーネルに付属のソースツリーを上書きするので、必ず、カーネルのソースツリーのトップディレクトリ（/usr/src/linuxなど、Red Hat 7.1の場合は/usr/src/linux-2.4）に移動してから、rootユーザーの状態で展開する。

```
$ cd /usr/src/linux-2.4
$ su
# tar xvzf /tmp/videodev20010302.tgz
```

次に、カーネルの再コンパイルを行うのだが、すでに、bttvのドライバが有効になっている場合には、モジュールの再コンパイルと再インストールだけ行えばよい。

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

本文中で解説したように、カーネルソースの状態が使用中のカーネルと一致していない場合や、bttvが有効になっていないような場合は、

```
# make xconfig
# make depend
# make bzImage
# make install
```

を実行してカーネルの再コンパイルとインストールを行ってから、改めて先ほどの手順でモジュールの再コンパイルとインストールをする必要がある。

**3. キャプチャカードドライバのインストール**

キャプチャカードドライバのインストールについては、カーネル2.2と2.4で手順は同じである。本文中で解説しているのと同じように、driver-20010316.tgzファイルを展開して、

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

を実行すればよい。

**4. /etc/modules.confの修正**

ドライバ類のインストールがすんだら、/etc/modules.confを編集して、「bttv」のロードを指定している部分を「bttv2」に書き換える（本文中、リスト2を参照）。

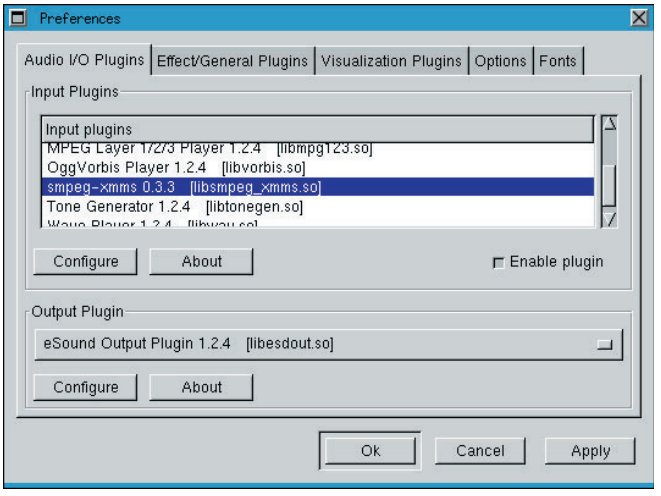
あとは、

```
# depmod -a
# rmmod bttv.o
# modprobe bttv2
```

を実行するか、マシンをリブートすればV4L2が使えるようになる。

レートとフレームレートのバランスを決めてほしい。

なお、mp1eでは、動きの激しい場面を低いビットレートで保存すると、モザイク状のノイズが目立つようになるが、文字や静止画の説明図のような場面では、ビットレートが低くても、かなり解像度の高い画面を保存できる。たとえば、語学番組のように、画面上に表示される文字を判読したい場合でも、500kbps（0.5Mbps）程度のビットレートで、十分実用に堪える画像を保存できるようだ。フレームレートについては、15fps程度でも、それほど不自然な感じはしないが、スポーツ中継のように動きの激しい画面では、25fps以上のフレームレートがないと、細かい動きを判別するのは難しくなるようだ。




画面5　XMMSで入力プラグインを確認する
XMMSのウィンドウ左上のアイコンをクリックして「オプション」-「設定」を選択すると、プラグインのインストール状況を確認できる。「入力プラグイン」のリストボックスに「smpeg-xmms」が含まれていればMPEG-1ビデオを再生できる。

※ここで紹介したリスト、mp1e関連のファイルは、Linux magazineのホームページ、
http://www.ascii.co.jp/linuxmag/
からダウンロードできます。


画面6　xmmsでのMPEG-1ファイルの再生
音声ファイルの再生と同じように、一時停止や巻き戻しなどの操作が可能だ。

## Column

### mp1e-1.8.1のインストール

mp1e-1.9preがうまく動かない場合には、mp1e-1.8.1をインストールすることになる。mp1e-1.8.1は、かなり古いLinux環境を前提に作られているので、Red Hat 7.1の環境でのコンパイルには若干の作業が必要になる。なお、Linuxのディストリビューションによっては、このような作業の必要なしにコンパイルできる場合もあるので、まず最初に、ソースコードに付属のドキュメント（READMEファイル）の手順でコンパイルを試みてみること。

**1. ソースファイルの展開**

適当な作業用ディレクトリの中でソースファイルを展開する。

```
$ tar xzf mp1e-1.8.1.tgz
```

mp1e-1.8.1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。通常なら、ここでREADMEの指示通りにコンパイルを実行するのだが、サウンドドライバにALSA以外を使っている環境では、エラーが発生してコンパイルできないことがある。そこで次の手順でエラーを回避する。

**1. ソースディレクトリ中のconfig.inファイルをviなどのテキストエディタで開き、次の行を見つける。**

```
AC_CHECK_LIB(asound, main)
```

この行の先頭に「dnl」という文字列を追加する（すなわち、この行をコメントアウトする）。

```
dnl AC_CHECK_LIB(asound, main)
```

**2. 次のコマンドを実行して、configureスクリプトを作り直す。**

```
$ autoconf configure.in > configure
```

**3. systemsディレクトリ中のmpeg.hファイルをテキストエディタで開き、末尾に次の行を追加する。**

```
#define
    DBL_MAX 1.701411834604692294E+38
```

（実際には1行で書く）

4. ソースディレクトリに戻り、通常のビルド処理を実行する。

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

以上の操作で、/usr/local/binディレクトリにmp1eコマンドがインストールされるはずだ。

# 箱の中のペンギンたち

文：みわよしこ
*Text : Yoshiko Miwa  miwachan@pp.iij4u.or.jp*

## 第7回　組み込みシステムの世界におけるLinux

多くのLinuxユーザーにとって、Linuxとはデスクトップ環境でありサーバ環境であります。つまり、「パソコンのOS」です。そして「パソコンのOS」には、それほど多くの選択肢はありません。たとえば現在、Intel系のCPUを搭載したマシンで利用できるユーザー環境のうち、主要なものはLinux、Microsoft Windows、FreeBSD、Solaris、BeOSといったあたりでしょう。

### 「ワン・オブ・ゼム」としてのLinux

ところが組み込みシステムの世界では様相はまったく異なり、OSの選択肢は100種類以上にも達します。選択肢の数からいえば、Linuxは「100分の1以下」なのです。数や量の違いは、質の違いを必ずもたらします。一般パソコンユーザーと組み込みシステム技術者には、さぞかし異質な世界が見えているに違いありません。

今回は、組み込みシステム開発の立場から見たLinuxの姿を追ってみたいと思います。

### OSの名前は大きな問題ではない

「Intel Inside」のロゴマークを模した「linux inside」というロゴマークを記憶している読者は少なくないでしょう。'98年から'99年ごろにかけて、筆者は「linux inside」というシールを貼り付けたマシンをしばしば目にしました（筆者自身のLinuxマシンにも貼り付けてありました）。Linuxユーザーにとって、自分が使っているOSがLinuxであることは、アピールする価値がそれほどに高いことでした。特に'98年以前、まだLinuxが日本で一般の人々に十分に注目されていなかった時期には、なおさらその傾向が強かったといえます。

しかし、組み込みOSの世界ではOSの名前が問題になることは極めて稀です。製品を購入する側にとって重要なことはOSが何であるかではなく、その製品の性能やコストでしかありません。特にコストがシビアな問題になる大量生産品の場合、OSの名前を出すことによって余分なコストが必要になることは、商品の売れ行きにマイナスの影響を与えます。したがって、OSの名前はたいていの場合、パンフレットや製品添付のドキュメントに記載されることはありません。

しかしながら、Linuxは組み込みシステムの世界で急激にシェアを広げています。昨年初頭の段階では1%でしたが（トロン協会調査。ちなみにMicrosoft WindowsとMicrosoft Windows CEを合わせて6.7%）、現在のシェアはその数倍以上になっているでしょう。

さて、組み込みシステムの場合、OSはどのように選択されるのでしょうか？

## 組み込みシステムにおけるOSの選択

パソコンにインストールするOSを選択する場合には、用途、価格、ユーザー本人の好みなどによって「このOSを」という選択が行われます。Linuxの場合、「最近話題になっているからインストールしてみたい」という動機によるインストールも珍しくありません。いずれにしても、たいていの場合、Linuxはユーザー本人が「Linuxを使いたい」と考えることによって選択され、マシンにインストールされます。

しかし組み込みシステムの場合、「Linuxを使いたいから」という理由によってLinuxが選択されることはまず考えられません。

組み込み用途のOS・リアルタイムOSには、100種類以上の選択肢があります。このうち、主に利用されているのは10種類程度です。さらに歴史の古い企業では、自社開発したOSを保有している場合もあります。

この中から、通常はアプリケーションに合わせてOSが選択されます。また、「自社開発のプロセッサを利用するため必然的に自社開発のOS」といった選択が行われる場合もあります。場合によっては「OSなし」という選択もあります。つまり、ここでLinuxが選ばれたとしても、組み込みシステム開発技術者にとっては「たまたま、今回のOSはLinux」ということに過ぎないのです。

## 組み込みシステム開発技術者がOSに求めるものは？

開発者にとって最も重要なツールは開発環境です。特に、多様なプロセッサが利用される組み込み用途では、どのプロセッサ上でも同じコンパイラ・デバッガを利用したいというニーズがあります。良質の開発ツールが提供されているOSは、ほかの条件が同じであればより選択されやすくなります。

このため、Linuxの組み込みシステム業界への普及を目的として2000年7月に設立された「日本エンベデッドリナックスコンソーシアム（Emblix）」では、今年1月から「開発環境ワーキンググループ」を発足させ、良質の開発ツールを安価に提供する体制の構築を目指しています。このことは、日本における組み込み用途で現在も広く利用されているITRONで、ツールの開発までは組織的には行われなかったことへの反省もあるようです。

いずれにしても、Linuxの組み込み用途での開発環境が整備されれば、組み込みシステム開発技術者にとっては「仕事


図1　家電製品とパソコンにおけるOSの重要性
ユーザーにとって組み込みOSの名前は「どうでもいい」要素。

のやりやすい環境の提供されたOSがひとつ増えた」という状況が発生します。しかし、それ以上ではありません。

また、リアルタイム性能を強く要求される用途では、当然ながらリアルタイム性が重視されます。リアルタイム性能だけで比較すれば、Linuxより優れたOSがいくつか存在します。たとえばITRONには、Linuxのように豊富なネットワーク機能やソフトウェア・スタック機能はありませんが、Linuxより優れたリアルタイム性能があります。

さらに「既存のソフトウェア資産の有効利用」という問題も避けて通れません。ソフトウェア技術者であれば誰でも、既存のソフトウェア資産をなるべく有効利用して新しいソフトウェアを開発したいと考えるものですが、組み込みシステムの世界も例外ではありません。そこにもし大量のITRONのソースコードが存在すれば、後続システムもITRONで……と考えるのが自然というものです。このような環境にある開発者が望むのは、LinuxがITRONを置換してしまうことではなく、LinuxとITRONのAPIを同時に利用でき、それぞれの機能を活かした開発、既存のITRONのソースコードを利用しながら、新機能をLinuxで追加するような開発を行えることです。パソコンOSの世界における「Linux 対 Windows」といった図式に類似するものは、組み込みシステムの世界にはないと言えるでしょう。

## 組み込みシステム業界から見たLinux

ここで視点を変えて、組み込みシステム業界全体から見たLinuxの姿を描いてみましょう。

まず、開発者がいなければ開発はできません。したがって、「開発者の確保」は非常に重要なのですが、2000年初頭のトロン協会の調査結果で、「リアルタイムOSの問題点」の第1位に挙げられていたのは「技術者が不足、またはいない」という問題です。このことは、前述した「どのプロセッサでも同じコンパイラ・デバッガを使いたい」という開発者のニーズとも関係します。少ない人員で開発を行わなくてはならないとすれば、複数のプロセッサで同じ開発環境を利用できるメリットは多大です。さらに、LinuxのAPIで組み込みシステム開発が可能になれば、UNIX技術者や大学でUNIXを学んだ学生を開発の「即戦力」とすることができます。このような面からも、Linuxの組み込み用途への期待が高まっています。

また一定規模以上の企業では、縮小する事業の技術者を拡大する事業の技術者へ転換する必要が常に存在します。たとえば最近では、消費者向けデジタル製品のニーズは増大する一方ですが、大規模システム構築のニーズはどちらかと

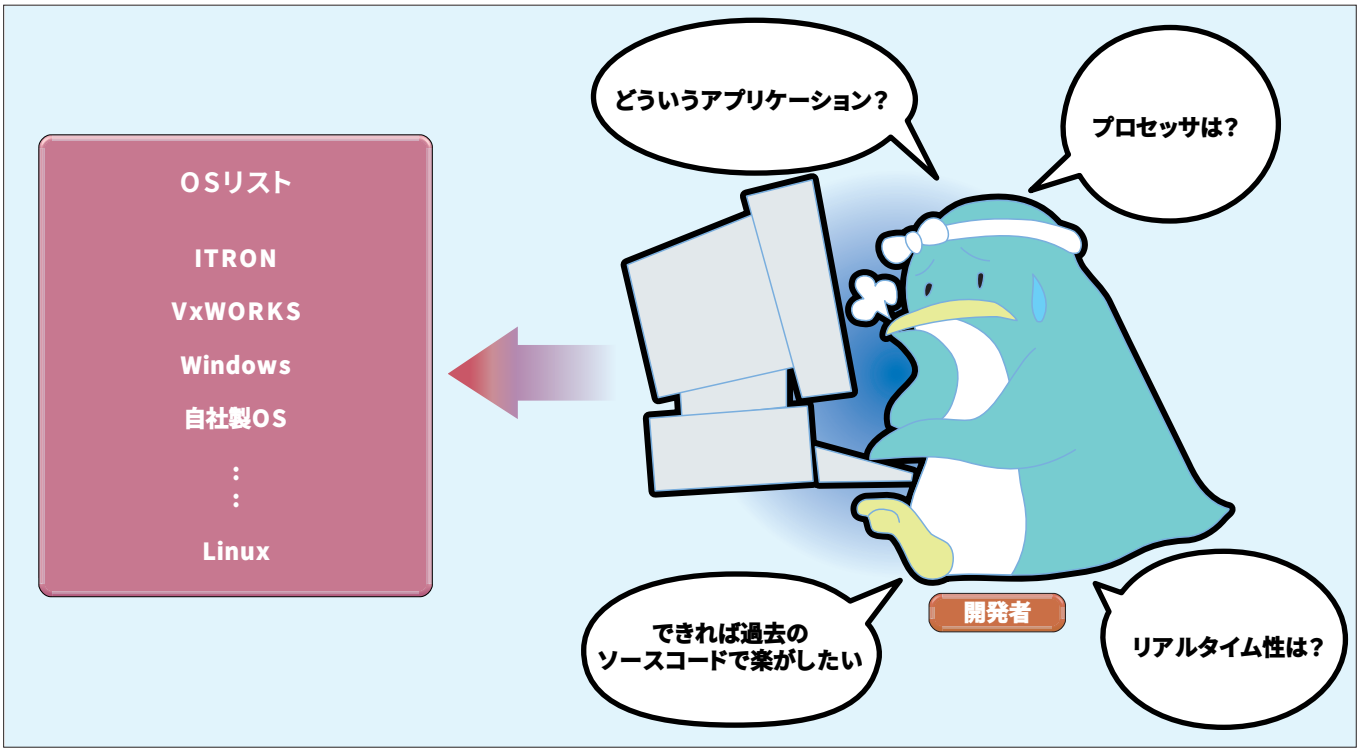



図2　組み込みシステムにおけるOSの選択
Linuxはいろいろな要素を考えてOSを選択する際の選択肢のひとつにすぎない。

いうと減少しています。大規模システムはUNIX系OSで構築されることが多いので、UNIX系OS技術者が企業内で「余る」という状況が発生します。もし、消費者向けデジタル製品の組み込みシステム開発が、Linuxやそのほかの UNIX系OSで行われれば、この「余った」UNIX系OS技術者をそちらに無理なく転換することが可能になります。企業内における人的資源の有効活用の手段のひとつとしても、Linuxには注目が集まっているといえます。

次に、ネットワーク通信機能を利用する製品のニーズが市場で高まっているという状況を考慮すると、プロトコルスタックが標準で含まれており、インターネットに代表されるネットワーク通信機能を容易に開発できるUNIX系OSは、組み込みシステム業界にとって非常に魅力的な存在です。また通信機能のみではなく、ストレージも同時に要求されることが多いことを考えると、ファイルシステムを備えているUNIX系OSはより重要性を増してゆくと考えられます。一般消費者にとって馴染み深い存在である携帯電話を取っても、「着メロ」や「待ち受け画像」、インターネット利用、日本語入力機能など、ネットワーク対応とストレージの重要性は年々増してきています。

3つめに、Linuxがオープンソースコミュニティの産物であることのメリットも忘れてはなりません。開発ツールのベンダーに限らず、企業には「M&A」がつきものですが、開発者にとって、馴染んだ開発ツールが供給されなくなることの痛手は大きなものです。Linuxのようにオープンソースコミュニティで開発され続けているツールには、「ベンダーのM&Aに脅えなくてよい」というメリットがあります。特に大企業で継続して長期間にわたる事業を計画する際には、ベンダーの事情に開発環境を左右される心配がないというメリットは多大です。

最後に、卑近な意味での「フリー」、すなわちロイヤリティが発生しないというメリットもあります。これは小規模ベンチャーにとっては非常に重要な要素です。市販の組み込みOSの場合、多大なロイヤリティのために開発が行いにくいという事情があるからです。

大企業にとっても、OSの利用に関するコストがかからないメリットはもちろんゼロではありません。しかし、コスト要因はOSのロイヤリティ以外にも数多く存在しますし、開発効率など、全体のバランスを重視した結果としてLinuxが選択されないというケースもあります。ロイヤリティが発生しないということは、それほど大きな魅力とはならないようです。

## 組み込みシステム業界にとってのLinuxの現在

製品の開発は、限られた人的、資金的、時間的リソースを有効利用して行われます。そのリソースを、販売して利益をもたらす製品、利益をもたらす部分の開発に注力したいと考えることは、製品を開発する側にとって極めて自然なことです。

OSは組み込みシステムにとって、部品のひとつに過ぎません。ハードウェア部品を選択するのと同じように、供給体制・サポート体制が整備されているOSを選ぶことは、開発にかけるリソースを有効に活用するのに役立ちます。

最近、組み込み用途向けのLinuxディストリビューションも数多く発表され、各ディストリビュータはそれぞれに高い性能と、安定したサポート体制を誇っています。このようなことを考えると、「組み込みシステム業界が安心して利用できるOSという土俵に、Linuxがやっと乗ってきた」というあたりが、現在の状況と言えそうです。

## 組み込みシステムにとってのLinuxの今後

一般Linuxユーザーにとって、「Linux」とは市販されている数多くのLinuxディストリビューションのことであり、またLinuxカーネルのことです。エミュレータを使用する場合を除いては、1台のマシンの上で複数のOSが同時に動作することはありません。Linuxはあくまで単独で利用されるOSです。

しかし、組み込みシステムにLinuxを利用する場合、Linuxが一般的なLinuxディストリビューションのような姿をしている必然性はまったくありません。また、Linuxが単独で組み込まれていなくてはならない必然性もありません。

組み込みシステムにとってのLinuxは、言わば「Linuxの姿をしていないLinux」です。必要に応じて、他のOSと同じ場で協調して働くLinuxです。また、場合によってはほかのOSの手助けをするLinuxです。

Linuxは今後も組み込みシステムという場で、一般Linuxユーザーには想像もつかない多様な形態への変容を続けてゆくでしょう。またその変容の過程や結果が、一般Linuxユーザーにとっての「Linux」に反映されることも当然考えられます。デスクトップOS、サーバOSの世界と組み込みOSの世界はまったく異質なものではありますが、相互に作用しあって将来をより多様に・多彩にすることが大いに期待できると筆者は考えています。

# 日本エンベデッドリナックスコンソーシアム アドバイザリー委員 田丸喜一郎氏に聞く

Interview

**日本エンベデッドリナックスコンソーシアム（Emblix）は、主に家電製品を開発する企業の立場から、日本の組み込みLinuxシステム業界を牽引する存在です。Emblixの会員企業には、技術力を誇る中小組み込みシステムメーカーと並んで、大手電機メーカーが数多く社名を連ねています。多様な企業を「束ねる」活動はどのようなものなのか、非常に興味深いところです。**
**今回は、Emblixの最近の活動についてお話を伺いました。**

――はじめまして。Emblixは、もうすぐ創立1周年ですね。

**田丸：**ええ、この1月からはワーキンググループ活動も開始し、具体的な活動に結びついてきているところです。

――まず、ワーキンググループ活動についてお話を聞かせてください。現在、「開発環境ワーキンググループ」「ハイブリッド・アーキテクチャ・ワーキンググループ」「リアルタイム・ワーキンググループ」という3つのワーキンググループがありますね。まず、開発環境ワーキンググループでは「開発環境の標準化」を目指しているとのことですが、ここでの「標準化」とはどういうことでしょうか？

**田丸：**コンパイラやデバッガの開発を、ツールベンダーがばらばらにやると効率が悪いですよね。たとえば、n個のディストリビューションがあってm個のベンダーがあって、それらの開発が完全にバラバラに行われるとすると、開発すべきツールはn×m個になるわけです。

――それは無駄が多いですね。

**田丸：**そこで、このワーキンググループで無駄のないツール開発を行うために必要となるものを、いろいろと準備しているというわけです。まずは、インターフェイス仕様の標準化ですね。今は「どのインターフェイスか」を含めてまだディスカッションの段階なのですが。

写真　日本エンベデッドリナックスコンソーシアム アドバイザリー委員 田丸喜一郎氏（たまるきいちろう）
1981年慶應義塾大学工学研究科博士課程修了。工学博士。同年、株式会社東芝に入社し、半導体技術研究所にてマイクロプロセッサおよび応用ソフトウエア開発環境の研究・開発に従事。その後、東芝本社技術企画室を経て、現在、システムLSI開発センター参事。現在の役職は、（社）トロン協会プロジェクト推進委員、日本エンベッデッドリナックスコンソーシアム アドバイザリー委員、株式会社モンタビスタソフトウエアジャパン取締役など。

――そういった部分の標準化が行われていると、ツールベンダーはその標準に則ってツールを開発すればよいということになりますね？

**田丸：**そういうことです。その結果、開発サイドは良質のツールを安価に手に入れることが可能になります。

――現在の一般Linuxユーザーの感覚では、開発環境の標準は「gcc」「gdb」といったいわゆる、「GNU処理系」なのですが。

**田丸：**実際にシステムを開発する立場から見ますと、GNU処理系の機能は決して十分に高くないのです。たとえば、組み込みソフト開発ではソースコード管理からオブジェクト生成までを扱える統合開発環境が主流なのですが、GNU処理系だけでは統合開発環境を構築できません。そこで、開発環境を整備するための活動が必要だという認識になったんです。

――次に、リアルタイム・ワーキンググループについてお尋ねさせてください。現在はいくつかのリアルタイムLinuxディストリビューションを比較検討されているということですが、Emblixという組織でそういう検討をしなくてはならない理由が、なんとなく腑に落ちないんです。開発を行っている企業がそれぞれ、ツールベンダーからカタログを取り寄せたり、試用版を利用してみたりして選択を行えばよいことではないでしょうか？

**田丸：**その「ツールベンダーのカタログ」が問題なんです。

――どういうことでしょうか？

**田丸：**現在のところ、用語もその定義も各ベンダーでバラバラなんです。

――それは困りますね。

**田丸：**そうでしょう？　まず、用語を統一することから始めなくてはならないのが現状です。それにカタログにある特性数値は単純に比較できないんです。

――といいますと？

**田丸：**もともと、数値だけで単純に特性を比較するということに無理があるんです。仕様によって特性数値が変わるということは非常によくあります。ですので、まず数値の測定の

やり方を統一し、その統一されたやり方でリアルタイムLinuxディストリビューションの性能特性を測定して、はじめて「カタログの特性数値を比較する」ということがナンセンスでなくなるんです。

──そういえば、オーディオ機器のレビューなどを見ていますと測定条件が必ず明らかにされていますし、比較は同一条件においての測定が行われていますね。当たり前といえば当たり前のことなのですが……。

**田丸：**現在のリアルタイムLinuxディストリビューションの場合、まずカタログをそのレベルまで持ってこなくてはならないという状況です。どの会社の性能一覧表も、同じ基準で、同じ書き方で書かれるようになるというのが、このワーキンググループの目標です。Linuxの間だけでの比較だけではなく、LinuxとほかのOSの比較という場面でもその必要性は高いんです。

──地味ですが重要なことですね。

**田丸：**ええ、まずは「どういう性能表示をすればよいのか」ということから活動してゆきたいと考えています。まず、ユーザーがOSの選択を誤らないようにするための手助けが、このワーキンググループの役割と言ってもよいでしょう。開発方法や開発手法の提案も将来はカバーしたいですね。

──ワーキンググループはもうひとつ、「ハイブリッドアーキテクチャ・ワーキンググループ」がありますね。こちらは「Linux on ITRON」が主目的ということですが、その目指すところはどういったことなのでしょうか？

田丸：Linuxを含めて、複数のOSのAPIをカバーする仕組みを作ることですね。単一OSではなく、複数のOSのAPIを使いたいというニーズは以前から高いんです。たとえばITRONとJavaを合わせた「JTRON」ですとか「Windows CE on ITRON」といった試みがこれまでも行われています。

──その流れの上にLinux on ITRONがあるということですね。

**田丸：**そうです。それぞれのOSの機能や特性によってAPIを選んで、組み込みシステム開発で要求される「個別対応」で最も適切な解決が行えるようになることが理想です。Linuxであるかどうかということではなく、トータルで「リアルタイム性が良く、機能が豊富である」ということが重要ですので、たとえば高度のリアルタイム性を要求される部分にはITRONを利用し、そうでない部分にはLinuxを利用するといった形態が考えられます。それにITRONの既存のソースコードを利用したいというニーズもありますしね。

──ソフトウェア資産の有効活用は、開発効率を重視する開発の現場では重要な問題ですね。

**田丸：**それに、Linux on ITRONはLinuxから見てもメリットがあるんですよ。ITRONはドライバが非常に豊富ですが、Linuxのドライバは組み込みシステム開発という面から見るとまだ不十分なんです。Linux on ITRONが実現すると、Linuxを利用した開発の場でITRONの豊富なドライバを利用できます。

──実現が楽しみですね。ところで、この3つのワーキンググループの目的は、結局のところどういったことなのでしょうか？

**田丸：**ひと言で言うと、「Linuxをカバーする仕組みを作りたい」ということです。「Linuxを改造する」ということではなく。

──Linuxの優れた点とそうとは言えない点を明らかにして、システム開発の場で、ほかのOSと長所を活かし合えるようにするということですね。デスクトップ環境や、サーバOSとしてLinuxを利用していると、こういう視点はなかなか持てないものですが……。

**田丸：**そうでしょうね。現在、電気機器にスイッチがひとつあれば、そこにはマイコンが1つあるのが普通です。一般の消費者の方々の身近に隠れている組み込みシステムの世界を、ぜひLinux magazine読者の皆さんにも理解いただきたいですね。デスクトップ環境とはまったく異なる世界ですから。

──そういえば、この連載の第1回でEmblix会長の中島達夫先生（早大教授）にインタビューさせていただいたのですが、「パソコンは、コンピュータの汎用性の高い一形態に過ぎない」とおっしゃっていました。

**田丸：**その通りです。ワークステーションやパソコンは汎用性の高い最終製品であり、より大きなシステムの部品でもあるわけです。私たちはそういう見方をしています。

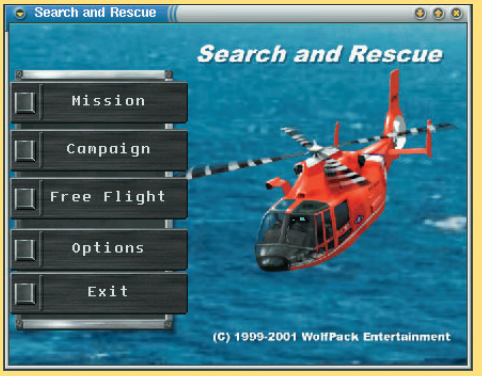
──まったく視点の異なる立場からのお話を伺えて、目が覚める気がします（笑）。最後に、今後のEmblixの活動の展開についてお尋ねさせてください。たとえば海外の組み込みLinux団体との協業といったワールドワイドな展開は考えられていらっしゃらないのでしょうか？

**田丸：**Emblixの会員企業の多くは、ワールドワイドにビジネスを展開されていますので、海外の関連団体との協調は重要です。たとえば、米国のEmbedded Linux Consortium（ELC）とはこれまでも何度か会合を持ってきましたが、今後は日本のビジネスにも関わっているELCの役員の方々にEmblixのアドバイザリー委員に就任いただき、より整合のとれた活動を進めていく予定です。

──かつてのLinuxの開発過程のようにグローバルな活動が期待できそうですね。本日は大変ありがとうございました。

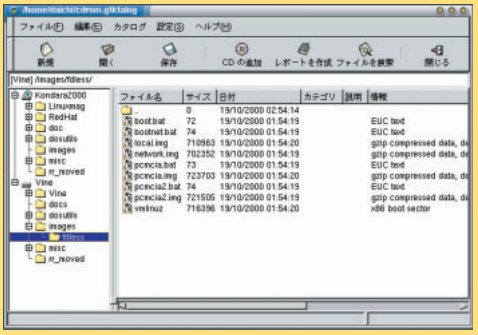
# Free Application Showcase

文：出井 一
*Text : Hajime Dei*




Wolfpack's Search And Rescue　P.140

Gnect　P.142



GTKtalog　P.134

GNOMEのCD-R作成統合環境
**Gnome Toaster** 129

パーティションを丸ごとイメージファイル化
**Partition Image** 132

CD-ROMのファイル情報をまとめたカタログを作成
**GTKtalog** 134

フラクタル図形の3D等高図をリアルタイム表示
**Kisomandel** 136

C言語風の文法を持つ計算ソフト
**calc** 138

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** 140

自分の駒を4つ並べる対戦パズルゲーム
**Gnect** 142

マウスの動きを認識してアプリを起動
**KGesture** 143

※紹介したソフトは、すべて付録CD-ROMに収録されています。

GNOMEのCD-R作成統合環境

# Gnome Toaster

バージョン：1.0Beta1　　ライセンス：GPL

http://gnometoaster.rulez.org/
http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html（cdrecord）

Gnome Toasterは、CD-Rの作成をGNOME環境で行うソフトだ。cdrecordやcdrdao、cdda2wavなどのフロントエンドとして動作し、データトラックとなる仮想ファイルシステムやオーディオトラックの登録、CD-Rへの書き込みなどをマウス操作で行える。付属の日本語カタログにより、ボタンやダイアログが日本語で表示される点もうれしい。実行にはGNOMEとcdrecordなどのツール類が必要だ。

### ●ツール類のインストール

まず、CD-R作成関係のツールcdrecordをインストールする。最新版は1.9で、mkisofsやcdda2wavも含まれる。作者が配布しているtarボールのほか、Vine PlusにRPMのバイナリ・ソースパッケージが用意されている。

バイナリパッケージを利用する場合は、「rpm -Uvh cdrecord-1.9-0vl3.i386.rpm」としてインストールする。ソースパッケージを「rpm --rebuild cdrecord-1.9-0vl3.src.rpm」としてリビルドし、/usr/src/redhat/RPMS/i386に作成されたバイナリパッケージをインストールしてもよい。

一方、tarボールの場合は、展開先のDEFAULTS/Defaults.linuxの30行目を「INS_BASE= /usr/local」に変更後に「./Gmake.linux」としてビルドし、「su」でrootになってから「make install」でインストールする。

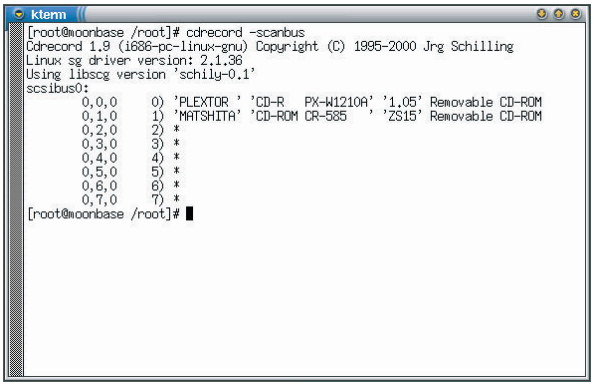
なお、このバージョンのcdrecordは、トラックアットワンス（TAO）書き込みとディスクアットワンス（DAO）書き込みの両方に対応している。

### ●Gnome Toasterのインストール

Gnome Toasterの最新版1.0Beta1は、tarボールとRPMパッケージの両方が配布されている。

作者が配布しているRPMパッケージはMandrake 8やRed Hat 7.0／7.1用なので、国産ディストリビューションではtarボールからRPMバイナリパッケージを作成するとよい（**リスト1**を参照）。バイナリパッケージは、「rpm -Uvh /usr/src/redhat/RPMS/i386/gtoaster-1.0Beta-1.i386.rpm」としてインストールする。

一方、tarボールを利用する場合は、「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。




画面1　cdrecordがCD-Rドライブを認識していることを確認する。

▼リスト1　Gnome Toasterのtarボールから RPMパッケージを作成する

```
$ tar zxvf gtoaster2001060411.tgz
$ mv gtoaster gtoaser-1.0Beta1
$ tar zcvf gtoaster-1.0Beta1.tar.gz gtoaster-1.0Beta1
$ cd gtoaster-1.0Beta1
$ ./configure
$ su
# cp ../gtoaster-1.0Beta1.tar.gz /usr/src/redhat/SOURCES
# rpm -bb gtoaster.spec
# exit
```

### ●CD-Rドライブの準備と確認

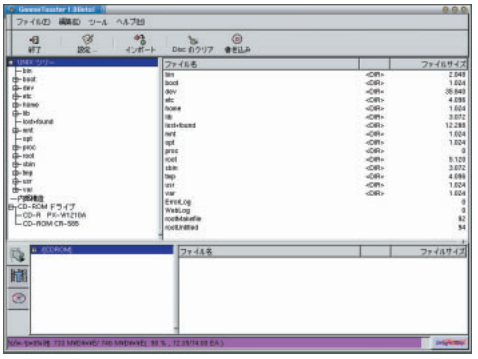
ATAPI接続のCD-Rドライブを接続している場合は、カーネルのSCSIエミュレーションサポートなどのモジュール作成（あるいはカーネルの再構築）を行い、/etc/lilo.confの内容を修正する必要がある。

こうした作業の詳細は、「CD-Writing HOWTO」（http://www.linux.or.jp/JF/JFdocs/CD-Writing-HOWTO.html）の第2章を参照されたい。また、本誌バックナンバー2000年11月号の54ページも参考になる。

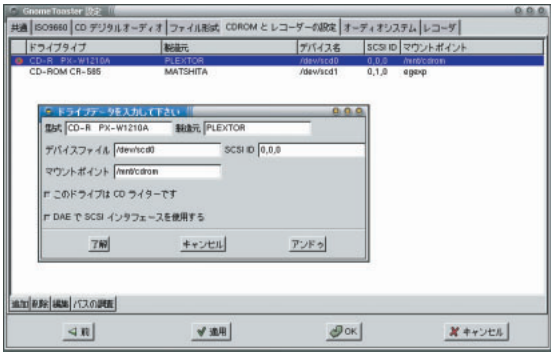
設定が完了したら、rootになった状態で、「cdrecord -scanbus」として、CD-Rドライブが認識されていることを確認しよう（**画面1**）。

### ●Gnome Toasterの起動と初期設定

cdrecordを利用したCD-R作成にはスーパーユーザー（root）権限が必要だ。一般ユーザーのまま書き込みを行う場合は、cdrecordにsuidビットを設定する必要があるが、ネットワークに



画面2　Gnome Toasterのウィンドウは上下の領域に分かれた構成だ。



画面3　利用するCD-Rドライブをリストから選択する。設定の変更も可能。

接続されたマシンではセキュリティ上のリスクになる。そこで、「su -」としてrootになった状態でGnome Toasterを起動するとよいだろう。

ktermなどのコマンドラインで「gtoaster&」として起動すると、上下に分割されたウィンドウが開く（**画面2**）。上部はデータソース、下部はCD-Rに書き込むファイルやトラックという構成だ。日本語環境では、メニューなどはすべて日本語で表示される。

初めて起動した場合は、ツールバーの[設定]ボタンを押して設定ダイアログを開き、いくつか設定の確認と修正を行う必要がある。

まず、[CD-ROMとレコーダーの設定]ページで、書き込みに使用するCD-Rドライブが選択されている（左端に印が付く）ことを確認する。デバイス名（/dev/scd0など）やSCSI ID、マウントポイントなどはすでに設定済みだが、間違っている場合は左下の[編集]ボタンで修正しよう（**画面3**）。

また、DAO書き込みにcdrecordを使う場合は、[レコーダ]ページの[クライアントを選択する]（英文では「Use alternative client」）にチェックを付ける必要がある（**画面4**）。チェックしない場合はcdrdaoが使われる。その他の設定は変更不要だ。

### ●仮想ファイルシステムに登録

ウィンドウ左上の領域に含まれる「UNIXツリー」には、ルートウィンドウ以下のディレクトリがツリー構造で表示され、クリックで選択したディレクトリのファイル一覧が右上の領域に表示される。

一方、ウィンドウ下部は、左端のタブにより、ファイルマネージャ／トラックエディタ／書き込み設定の各ページを切り替え可能だ。

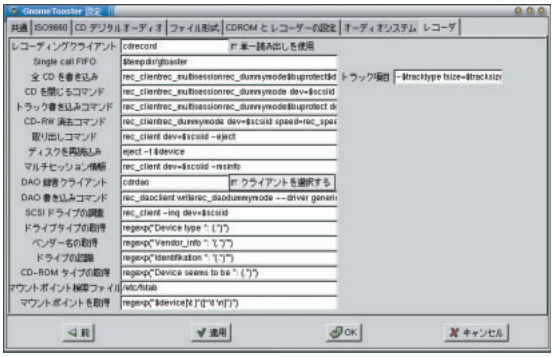
CD-Rのデータトラックに関しては、一番上のタブで表示されるファイルマネージャを利用する。このページには、CD-R作成時にデータトラックとして書き込まれる仮想ファイルシステムのディレクトリ構造とファイル一覧が表示されている。

仮想ファイルシステムにファイルを登録するには、「UNIXツリー」以下のファイルをファイルマネージャにドラッグ&ドロップすればいい。ディレクトリをドロップすると、その下のファイルやサブディレクトリも含めて一括登録される（**画面5**）。また、ファイルの合計サイズが一番下のステータスバーにグラフ表示されるので参考にするとよいだろう。

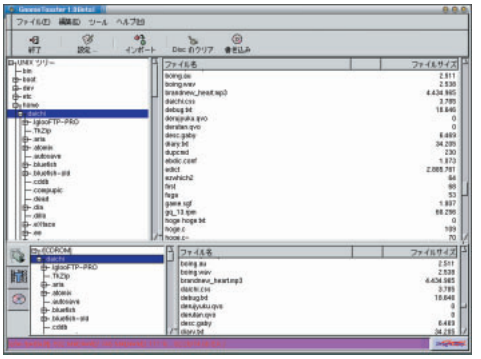
なお、マルチセッションのCD-Rに追記する場合は、CD-Rをドライブに挿入後、ツールバーの[インポート]ボタンを押して、現在のデータトラックの内容を仮想ファイルシステムに取り込む必要がある。

仮想ファイルシステムに登録したファイルやディレクトリは、右クリックメニューから名前の変更や削除などが可能だ。また、仮想ファイルシステム上にディレクトリを直接作成することもできる。

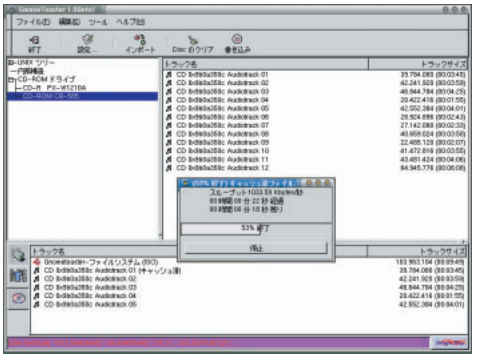
なお、現在のGnome Toasterの仮



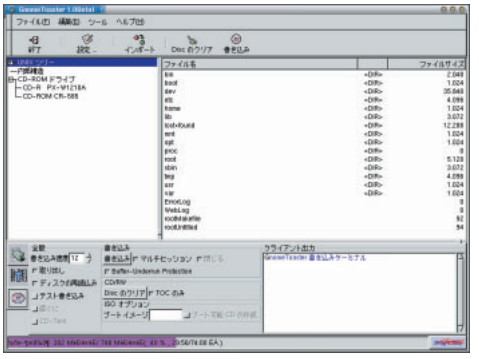
画面4　DAO書き込みにcdrecordを利用する場合はここをチェックする。



画面5　仮想ファイルシステムにドラッグ&ドロップでファイルを登録。



画面6　音楽CDからトラックリストに登録。リッピングは自動実行される。



画面7　書き込み速度などを設定した後で[書き込み]ボタンを押そう。

想ファイルシステムでは、シンボリックリンクを扱えない点に注意されたい（将来改善の予定）。このため、シンボリックリンクをドラッグ＆ドロップすると、リンク先のファイルやディレクトリの内容が登録される。

### ●オーディオトラックの登録

CD-Rにオーディオトラックを作成するには、ウィンドウ下部の表示を中央のタブのトラックリストに切り替える。このほか、CD-ROMのデータトラックを丸ごとコピーしたり、ISOイメージからCD-Rを作成する場合もトラックリストを利用する。

トラックリストには、CD-R作成時にトラックとして書き込まれる内容が一覧表示される。すでに仮想ファイルシステムにファイルを登録している場合は、「Gnometoaster-ファイルシステム」というデータトラックが表示されているはずだ。データの登録はドラッグ＆ドロップで行う。

たとえば、音楽CDのオーディオトラックをCD-Rにコピーするには、ウィンドウ左上の領域に含まれる「CD-ROMドライブ」以下のツリーから、音楽CDを挿入したCD-R／CD-ROMドライブ名をクリックして選択する。

すると、ウィンドウ右上の領域にオーディオトラックの一覧が表示されるので、コピーしたいトラックをウィンドウ下部のトラックリストにドラッグ＆ドロップすればいい。自動的にcdda2wavが起動して、オーディオトラックからWAVEファイルへの変換（リッピング）が行われる（**画面6**）。

一方、WAVE／MP3／Oggファイルなどから CD-Rのオーディオトラックを作成する場合は、ウィンドウ左上の領域に含まれる「UNIXツリー」から、対象となるファイルをドラッグし、（ファイルマネージャではなく）トラックリストにドロップすればいい。

なお、通常の（左ボタンによる）ドラッグでは、CD-R作成時にデコーダ（sox／mpg123／ogg123など）が起動してオーディオデータに変換されるが、右ボタンを使ってドラッグすると、ドロップした時点で変換される。

トラックリストに登録したトラックは、リスト内でのドラッグ＆ドロップにより順番を変更できる。ただし、データトラックはリストの先頭に配置する必要がある。

### ●CD-Rに書き込みを行う

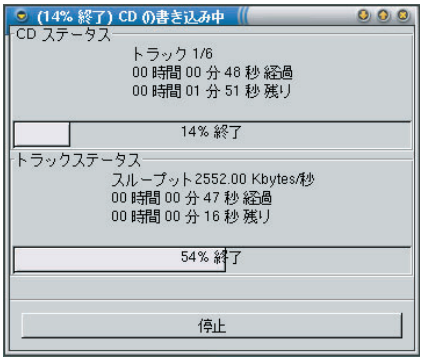
データトラックやオーディオトラックの登録が完了したら、ウィンドウ下部の表示を一番下のタブの書き込み設定ページに切り替える（**画面7**）。

まずは、書き込み速度の選択やマルチセッションの有無、Burn Proofを利用するかといった設定を行う。また、トラックアットワンス（TAO）書き込みではなくディスクアットワンス（DAO）書き込みを利用する場合は、左下の[直ぐに]をチェックすればいい。

ブランクCD-R（あるいはマルチセッション設定で書き込んだCD-R）をCD-Rドライブに挿入後、ツールバーか書き込み設定ページの[書き込み]ボタンを押そう。

初期設定では、mkisofsによるISOイメージの生成とcdrecordによるCD-Rへの書き込みが並行して（オンザフライで）行われる。作成中はダイアログに進行状況がグラフ表示される（**画面8**）ほか、ウィンドウ右下の[クライアント出力]にも、ツールの出力や各種メッセージが表示される。

青い文字で「CDの書き込みが終了しました」というメッセージが表示されれば作業は終了だ。ウィンドウ左上の領域の「CD-ROMドライブ」以下のツリーからCD-Rドライブ名をクリックして選択すると、右上の領域に現在のCD-Rのトラックリストが表示される。




画面8　書き込み処理の進行状況はダイアログの棒グラフで確認できる。

パーティションを丸ごとイメージファイル化

# Partition Image

バージョン：0.3.6　　ライセンス：GPL

http://www.partimage.org/

Partition Imageは、指定したパーティションの内容をイメージファイルとして保存できるソフトだ。CD-Rなどに保存しておけば、何か問題が起きた場合に復元できる。ext2／ReiserFSのほか、FAT16／32にも対応している。使用しているデータブロックだけを保存するうえに圧縮されるため、イメージファイルはコンパクト。指定サイズに分割することも可能だ。動作にはnewtなどが必要となる。

### ●ビルドとインストール

Partition Imageは、安定版（0.3.6）と開発版（0.5.x）の二種類のtarボールが配布されており、今回は安定版を取り上げる。「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。実行ファイルは、rootのみ実行可能な/usr/sbinにインストールされる。

tarボールに含まれるSPECファイルからRPMのパッケージを作成することも可能だ。「su -」でrootになった状態で、「rpm -tb partimage-0.3.6.tar.gz」とすると、/usr/src/redhat/RPMS/i386にバイナリパッケージが作成される。これを、「rpm -Uvh partimage-0.3.6-1.i386.rpm」としてインストールすればいい。

このほか、フロッピーやCD-Rからブートして Partition Imageを実行するためのイメージファイルも配布されている。これらについては後ほど説明することにしよう。

### ●起動前の注意点

起動する前に注意すべき点を2つ挙げておく。1つめは、Partition Imageを利用したパーティションの保存や復元にはroot権限が必要なことだ。あらかじめ「su -」としてrootになっておこう。なお、suコマンドの引数に「-」を付け忘れると、rootになっても/usr/sbinがPATHに含まれないため、Partition Imageを実行できないので注意されたい。

2つめは、マウント中のパーティションに対しては保存や復元操作を行えないことだ。つまり、保存や復元の対象となるパーティションは、あらかじめumountコマンドを使ってアンマウントしておく必要がある。

もし、ルートパーティションのようにアンマウントがそもそも不可能なパーティションを保存／復元したいなら、後述するフロッピーブートによる方法を利用しよう。

### ●パーティションを選択する

コンソール（Xを起動していない状態）か、rxvtなどの端末画面で「partimage」とすると、パーティションの一覧などを含む最初の画面が表示される（**画面1**）。なお、ktem上で実行すると、表示色の関係で画面の文字が見にくくなってしまう。

画面の上から順に、パーティション一覧が表示されたリストボックス、イメージファイル名を指定するテキストボックス、保存か復元かを選択するラジオボタン、処理を進めるボタンなどのGUI部品が並んでいる。

まずは、イメージファイル化して保存するパーティションを、↑／↓キーを使ってリストから選択する。Linuxの代表的なファイルシステムであるext2や、ジャーナリング機能を備えたReiserFSのほか、WindowsやDOSで使われるFAT16／32も選択可能だ（NTFSは部分的な対応のみ）。

続いて、Tabキーでカーソルを移動して、イメージファイル名をフルパスで指定する。gzipで圧縮する場合は「.gz」、bzip2で圧縮する場合は「.bz2」という拡張子を付けておこう。

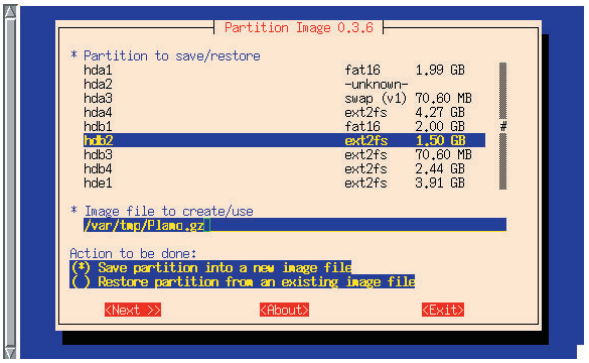
なお、イメージファイルの作成先となるパーティションは、十分な空き容量を確保する必要がある。gzipやbzip2で圧縮されるとはいえ、保存対象となるパーティションのサイズによっては数百Mバイト～数Gバイトのイメージファイルが作成されるからだ。

### ●オプション設定と保存実行

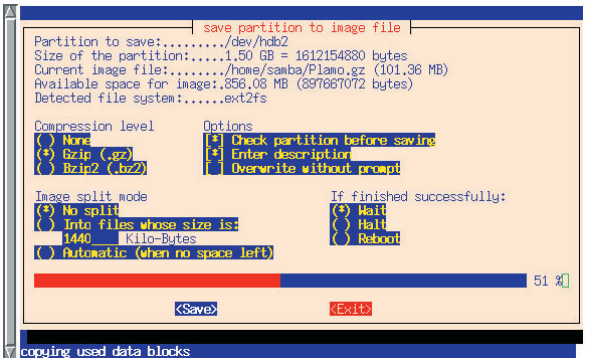
Tabキーを押して<Next>>ボタンまでカーソルを移動させ、スペースキーを押して次の画面に切り替えよう（**画面2**）。ここでは、イメージファイルの圧縮や一定サイズの分割といったオプションを設定した後、実際にイメージファイルの保存を行う。

イメージファイルの圧縮は、[Compression Level]の3つのラジオボタン、「None」「Gzip」「Bzip2」から選択する（選択操作はスペースキー）。bzip2による圧縮はかなりの時間を要するので、通常はgzipによる圧縮を利用するのがお勧めだ。

また、イメージファイルを複数のリムーバブルメディア（CD-Rなど）に分割保存するには、[Image Split mode]の「Into files whose size is NNNN Kilo-Bytes」を選択する。「NNNN」には「614400」（600Mバイト相当）など分割サイズをKバイト単位で記述す




画面1　最初の画面では、保存・復元するパーティションの選択を行う。




画面2　圧縮方式や分割サイズなどを設定してから保存処理を実行しよう。

ればいい。

オプション設定が完了したら、Tabキーを押して<Save>ボタンにカーソルを移動させてスペースキーを押そう。パーティションの説明を入力後に、イメージファイルへの保存処理が開始される。作成中は画面下の棒グラフで進捗状況を確認できる。

棒グラフが100%に達したら、最初の画面で指定したイメージファイルが完成している。CD-Rに保存する場合は、このイメージファイルをそのままcdrecordやGnome Toaster（**今月号129ページ**で紹介）といったCD-R作成ソフトでCD-Rに書き込めばいい。

### ●パーティションの復元も簡単

イメージファイルからパーティションの復元を行う場合、最初の画面での操作は保存の場合とほとんど同じだ。まず、復元先のパーティションをリストから選択し、次にハードディスクやCD-R上に保存されたイメージファイルのフルパスを指定する。

唯一違うのは、[Action to be done]の設定を「Save partition…」から「Restore partition…」に切り替えることだ。これで、Partition Imageが復元モードになる。

<Next>>ボタンをスペースキーで押すと、復元用のオプション設定画面に切り替わる（**画面3**）。Tabキーを押して<Restore>ボタンまでカーソルを移動させ、スペースキーを押そう。パーティション情報の表示や最終確認の後で復元処理が始まる。

復元先のパーティションに現在書かれている内容は、復元処理によってすべて失われてしまうので注意されたい。また、保存時とは別のハードディスクに復元する場合は、fdiskやpartedなどを使って同じサイズのパーティションを作成してから、Partition Imageを起動する必要がある。

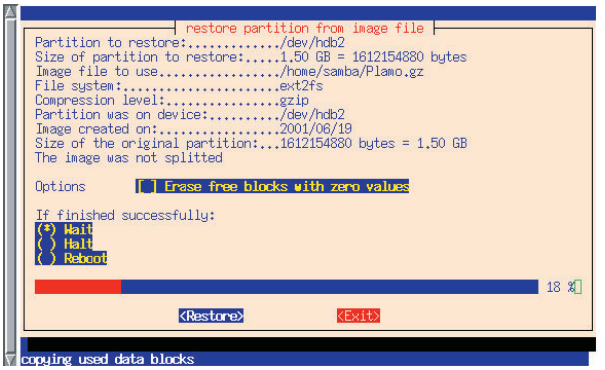
### ●フロッピーからのブートを行う

ルートパーティションをイメージファイルとして保存したり、CD-Rなどから復元したい場合には、別途配布されているフロッピーイメージを利用して2枚組みの専用フロッピー（1.44Mバイト）を作成し、このフロッピーからLinuxをブートしてPartition Imageを実行すればいい。フロッピーイメージは、付録CD-ROMに収録されている。

まず空きフロッピーを2枚用意し、1枚目のフロッピーをFDDに入れてから、「dd if=partimage-0.3.5-bootdisk-1.raw of=/dev/fd0」とする。書き込みが終わったら、2枚目のフロッピーに入れ替え、「dd if=partimage-0.3.6-rootdisk-2.raw of=/dev/fd0」とすれば作成は終了だ。

1枚目のフロッピー（ブートディスク）をFDDに入れてリブートすると、フロッピーからLinuxが起動する。「VFS: Insert root floppy…」とメッセージが表示されたら、2枚目のフロッピー（ルートディスク）に入れ替えてEnterキーを押せばいい。

このほか、ブート可能なCD-Rを作成するEltoritoイメージファイルも用意されており、付録CD-ROMにはpartimage-0.3.6-bootcd-2.imgという名前で収録されている。これを用いてのCD-R作成についての詳細は、Webサイトで参照できる最新マニュアルの4.4を参照されたい。




画面3　保存したイメージファイルからパーティションを復元する。

CD-ROMのファイル情報をまとめたカタログを作成

# GTKtalog

バージョン：0.99.3　　ライセンス：GPL

http://gtktalog.sourceforge.net/

GTKtalogは、CD-ROMなどのファイル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、実際にCD-ROMをマウントすることなく高速にファイル検索などが可能だ。さらに、tarボールやRPMパッケージ内のファイル情報を取り込む「仮想ファイルシステム」（VFS）機能も用意されている。動作にはGTK+が必要だ。

## ●ビルドとインストール

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。「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

tarボールに含まれるSPECファイル（gtktalog.spec）からRPMパッケージを作成することも可能だ。ただし、そのままではエラーが表示されるので、以下のようにしてファイルを修正する必要がある。

適当なディレクトリでtarボールを展開し、src/gtktalog/vfs.cの26行目の後ろに「#include <signal.h>」という1行を追加する。また、RPMのバージョンが3.0.5より古い場合は、gtktalog.specの71行目を「make prefix=$RPM_BUILD_ROOT%{_prefix} install」に書き換えよう。

これらの修正が終わったら、tarボールを展開したディレクトリで、「tar zcvf gtktalog-0.99.3.tar.gz gtktalog-0.99.3」として新たなtarボールを作成する。

あとは、「rpm -tb gtktalog-0.99.3.tar.gz」とすると、RPMバイナリパッケージが作成されるので、「rpm -Uvh /usr/src/redhat/RPMS/i386/gtktalog-0.99.3-1.i386.rpm」としてインストールすればいい

## ●起動と初期設定

ktermなどのコマンドラインで「gtktalog&」として起動すると、左右に分割されたウィンドウが開く。左がディレクトリツリー、右がファイル一覧という構成だ（**画面1**）。

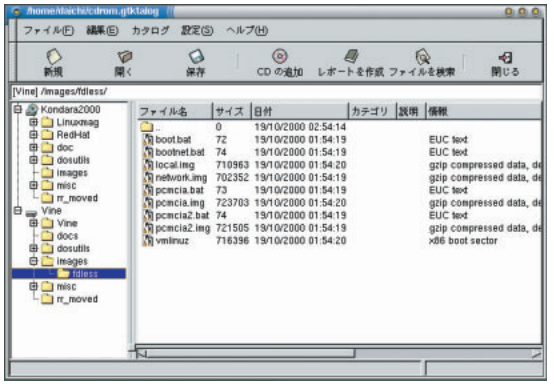
まずは、初期設定を行うため、[設定]－[設定]で設定ダイアログを開こう。ダイアログはジャンル別のツリーで分類されている。最初の[ディスクセットアップ]ページでは、CD-ROMのマウントポイント（初期値は/mnt/cdrom）を確認しておこう（**画面2**）。

続いて、[仮想ファイルシステム]ページに切り替える。ここでは、tarボールやRPMパッケージなどからファイル情報をカタログに取り込むための設定が並んでおり、[編集]ボタンでその内容を確認できる。

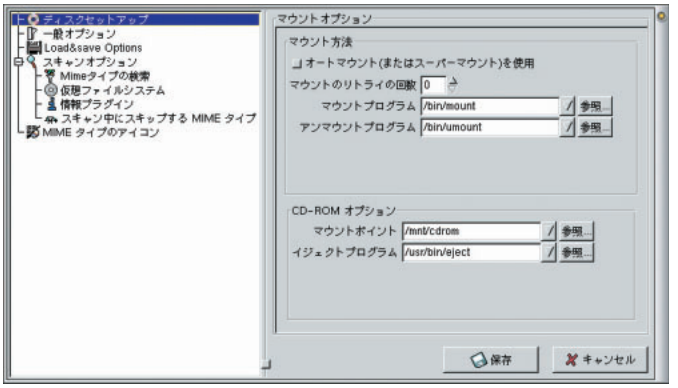
たとえば、bzip2で圧縮されたtarボールの場合、ファイルタイプが「application/x-bzip-compressed-tar」で、ファイル情報を取り出すコマンドは「tar」、オプションは「tjvf」、ファイル名は出力の5番目のフィールド…といった具合だ（**画面3**）。なお、国産ディストリビューションで使われるtarのbzip2対応オプションは「I」（大文字アイ）なので、[プラグインオプション]の設定を「tIvf」に修正しておく必要がある。

## ●CD-ROMのファイル情報を登録

GTKtalogを最初に起動した場合は、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する。次回からは、ハードディスクに保存したカタログファイルを[開く]ボタンで読み込めばいい。カタログファイルは独自形式




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　設定ダイアログでCD-ROMのマウントポイントなどを確認する。

で保存される。

ディスクカタログには、複数のCD-ROM（あるいはその他のメディア）のファイル情報を登録でき、次の2通りの登録方法が用意されている。

### (1) CD-ROM専用の方法

CD-ROMをドライブに挿入し、ツールバーの[CDの追加]ボタンを押すと、自動的にCD-ROMがマウントされ、あらかじめ設定したマウントポイント（初期値は/mnt/cdrom）以下のファイル情報がカタログに追加される。

CD-ROMのすべてのファイル情報を取得すると自動的にイジェクトされるので、次のCD-ROMに入れ替えて[CDの追加]ボタンを押せばいい。この作業の繰り返しにより、複数のCD-ROMを効率よくカタログ化できる。

現在の進行状況はウィンドウ右下のプログレスバーで確認できる。情報取得中も既存のカタログの内容を参照できるし、途中で処理を中止することも可能だ（**画面4**）。

### (2) CD-ROM以外にも適用可能な方法

GTKtalogでは、CD-ROM以外のリムーバルディスク（MOやDVD-RAMなど）や、ローカル／リモートのハードディスクのファイル情報をカタログに登録することもできる。ただし、対象となるディスクをあらかじめマウントしておく必要がある。

[カタログ]－[Add a new directory]でダイアログを開き、ディスクラベルやマウントポイント、仮想ファイルシステム（VFS）利用の有無などを設定する。[了解]ボタンを押すと、マウントポイント以下のファイル情報がカタログに追加される。

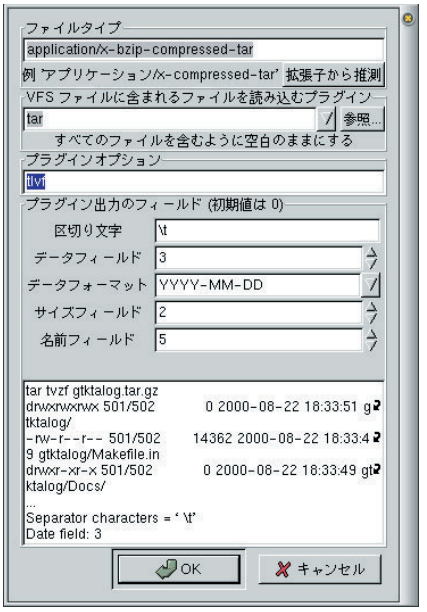
### ●カタログの閲覧と検索

ディスクカタログを作成すると、左側のペインには、CD-ROMごとのディレクトリツリーが表示され、クリックで選択したディレクトリのファイル情報が右側のペインに表示される。

また、仮想ファイルシステム（VFS）を有効にしてカタログを作成した場合は、tarボールやRPMパッケージなどもディレクトリとして表示され、その中に含まれるファイル情報が右側のペインに表示される（**画面5**）。通常のディレクトリとVFSのディレクトリはアイコンの色で区別できる。

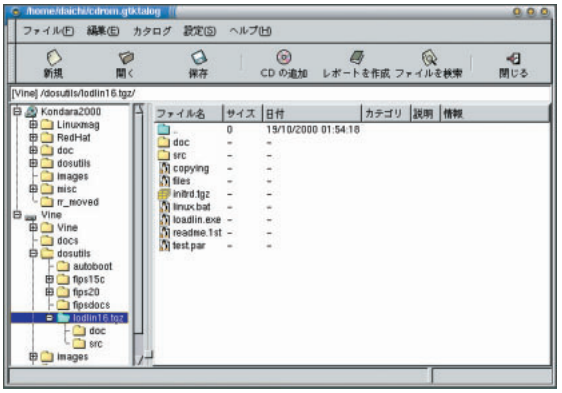
カタログ中のファイル情報は、右クリックメニューで編集や削除を行える。編集に関しては、ファイル名の変更のほか、カテゴリー・説明の追加が可能だ。これらの情報は右側のペインに表示されるほか、検索の際のキーとしても利用できる。

ファイルを検索するには、ツールバーの[ファイルを検索]ボタンを押して検索ダイアログを開く。通常はファイル名のみの単純検索が行われるが、[More>>]ボタンを押すと、正規表現を利用したファイル名パターンやファイルの日付・サイズ、検索するディスクなどの複数の条件を組み合わせた拡張検索に切り替わる（**画面6**）。

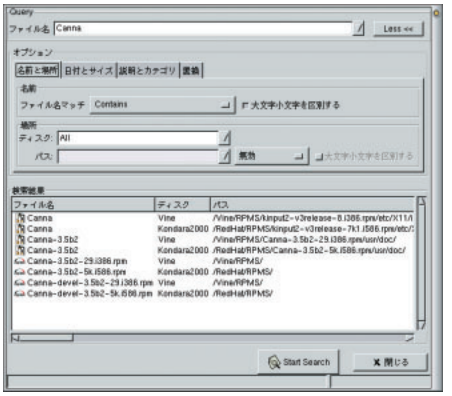



画面3　仮想ファイルシステムで使われるコマンドやオプションを設定。


画面4　CD-ROMのファイル情報を取得中。処理はいつでも中止できる。




画面5　tarボールやRPMパッケージの内容もカタログに登録される。




画面6　複数の条件を組み合わせた拡張検索でファイルを探し出そう。

フラクタル図形の3D等高図をリアルタイム表示

# Kisomandel

バージョン：0.66　　ライセンス：GPL/QPL

http://members.tripod.de/HelmutSteger/

Kisomandelは、マンデルブロ集合やジュリア集合といったフラクタル図形を描画するソフトだ。フラクタル図形の任意の部分を3D等高図として表示し、細かな構造をリアルタイムに確認できる。3D表示の方法は線／面／柱など複数用意され、平滑化レベルも変更可能だ。また、別配布のパレットデータを利用することで、さまざまな色合いの表示に切り替えられる。動作にはQt 2.2以降が必要だ。

### ●ビルドとインストール

Kisomandelはソースとバイナリがそれぞれtarボールとして配布されている。通常はソースのtarボールから各自のシステムに適合したバイナリを作成したほうがよいだろう。tarボールの展開先で、「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする。

一方、バイナリのtarボールは、ビルド済みのバイナリ（i586用）とドキュメント類のみで構成されている。インストーラは含まれていないので、rootになった状態で「cp kisomandel /usr/local/bin」などとして自分でコピーする必要がある。

このほか、実行に必須ではないが、フラクタル図形の表示色を変更するカラーパレットをまとめたtarボールが別途配布されている。適当なディレクトリ（ユーザーのホームディレクトリなど）に展開しておこう。

### ●2つのウィンドウで図形を表示

ktermなどのコマンドラインで「kisomandel&」とすると、ツールバーなどを持つメインウィンドウと、フラクタルウィンドウの2つが開く（**画面1**）。どちらのウィンドウも自由にサイズを変更可能だ。

フラクタルウィンドウには、おなじみのマンデルブロ集合に基づくフラクタル図形が表示されており、小さな白い矩形で囲まれた部分が、メインウィンドウにワイヤーフレームの3D等高図として表示される。

フラクタルウィンドウ上では、マウスのドラッグによって白い矩形を移動でき、メインウィンドウの3D等高図もそれに連動して即座に変化する。ふだんあまり気にすることのないフラクタル図形の細かな構造をじっくりと観察してみよう。

このほか、フラクタル図形の表示範囲を右クリックとクリックで指定したり、ダブルクリックでフラクタル図形を3倍（変更可）に拡大したり、中ボタンのドラッグ（またはCtrl－ドラッグ）によってフラクタル図形をスクロールすることもできる。

### ●3D等高図の表示を変更する

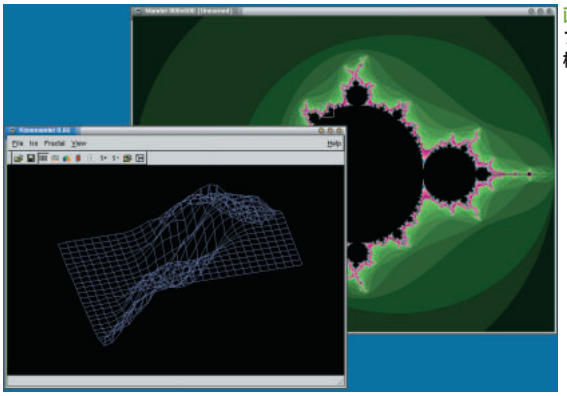
メインウィンドウに表示される3D等高図は、ツールバーのボタンを押すだけで、ワイヤーフレーム（モノクロ）／ワイヤーフレーム（カラー）／面／柱と4種類の表示を切り替えられる（**画面2**）。同様に、境界線の有無や平滑化レベル（10段階）も変更可能だ。

3D等高図上でのドラッグで左右方向、Shift－ドラッグで上下方向にそれぞれ回転し、好きな角度から眺められるほか、Shift－右ボタンドラッグによる拡大／縮小、中ボタンドラッグ（またはCtrl－ドラッグ）によるスクロールなどの操作も用意されている。

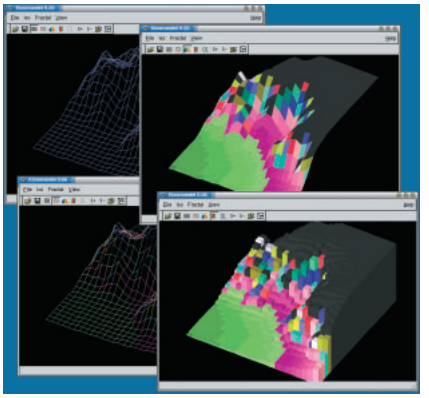
また、右クリックメニューから[Settings]を選択すると、3D等高図の表示に関する設定ダイアログが開く（**画面3**）。

このダイアログでは、3D等高図の解像度や高さ、拡大率、線の色、表示角度などを変更でき、その結果が即座に3D等高図に反映される。

このほか、＋／－キーで解像度を変更したり、rキーで表示を一回転させる



画面1　メインウィンドウとフラクタルウィンドウの2画面構成。



画面2　3D等高図の表示方法は以下の4種類が用意されている。

など、ショートカットキーもいくつか用意されている。詳細は付属のマニュアルを参照されたい。

### ●さまざまなカラーパレットを利用する

フラクタル図形の表示色を変更するには、ツールバーの[Open palette]ボタンを押して、使用するカラーパレットを選択する。

カラーパレットのtarボールには、300種類以上のカラーパレットが含まれており、さまざまな色合いのフラクタル図形を楽しめる（**画面4**）。ただし、カラーパレットの選択によっては、3D等高図が見づらくなる場合もあるので気をつけよう。

なお、[Open Palette]ボタンで開くディレクトリは、設定ダイアログの[Other Settings]ページにある[Default Palette Directory]で変更できる。カラーパレットのtarボールを展開先のpaletteディレクトリに設定しておくといいだろう。

### ●ジュリア集合に切り替える

フラクタル図形をマンデルブロ集合からジュリア集合に切り替えるには、メニューから[Fractal]－[Fractal type]－[Julia]を選択するか、jキーを押せばいい。

すると、現在の座標と拡大率のままジュリア集合に基づくフラクタル図形に切り替わる（**画面5**）。図形が異なるだけで操作の方法は同じだ。もし、座標と拡大率を初期状態に戻したい場合は、[Fractal]－[Reset to default coords]を選択すればいい。[Fractal]－[Fractal type]－[Mandelbrot]を選択するか、mキーを押すと元のマンデルブロ集合の図形に戻る。

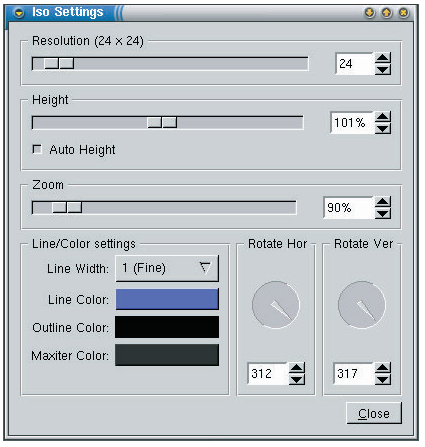
このほか、[Fractal]－[Settings]で開くダイアログ（**画面6**）では、現在の表示範囲を数値で指定できる。また、ダブルクリックによる拡大率や、ジュリア集合の計算に使われる定数なども変更可能だ。

### ●美しい画像を見つけたら

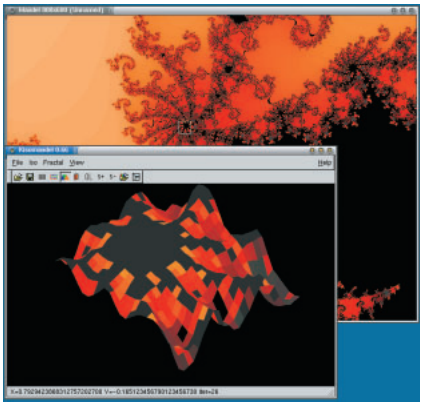
フラクタル図形を表示しながらいろいろとパラメータを変更している最中に、美しい画像を見つけたら、ツールバーの[Save fractal parameters]ボタンを押して、現在のパラメータをファイルに保存しよう。

保存したパラメータは[Open fractal parameters]ボタンで読み込むことができ、保存時と同じ座標と拡大率の地形がウィンドウに表示される。3D等高図の表示方法などは保存されないため、保存時と完全に同じ表示になるわけではない。

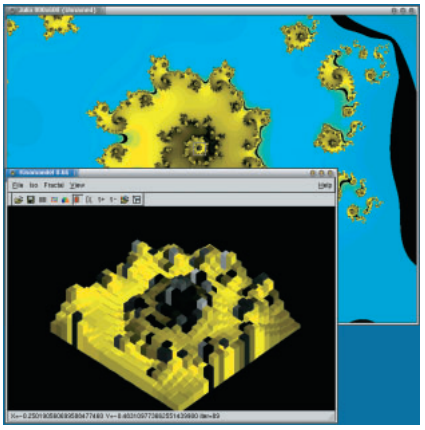
このほか、現在の3D等高図をJPEG／PNG／BMP形式の画像ファイルとして保存したり（[File]－[Save iso picture]以下から形式を選択）、直接プリンタで印刷する（[File]－[Print iso]を選択）こともできる。



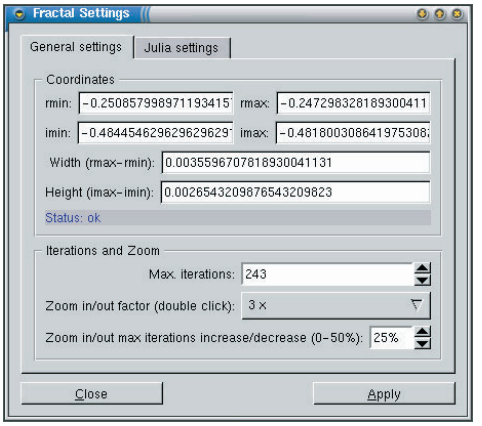

画面3　変更が即座に反映する、3D等高図の表示に関するダイアログ。



画面4　カラーパレットを切り替えると、表示の色合いがさまざまに変化する。



画面5　ジュリア集合によるフラクタル図形に切り替えることもできる。




画面6　表示範囲などのパラメータを数値で指定するダイアログ。

## C言語風の文法を持つ計算ソフト

# calc

**バージョン：2.11.5-4.5　　ライセンス：LGPL**

http://www.isthe.com/chongo/tech/comp/calc/

**calcは、日常の四則演算はもちろん、巨大数値の計算や行列計算までカバーするコンソールベースの計算ソフトだ。280個以上の組み込み関数が用意されており、変数の利用も可能だ。また、C言語風の文法に基づいて、分岐や繰り返しなどのプログラム的な処理を行ったり、ユーザー関数を定義することもできる。さらに、一連の処理をスクリプトとしてファイルから読み込み、一括処理する機能もある。**

### ●ビルドとインストール

calcは、ソース一式をtar＋gzipしたtarボールのみ配布されている。「make -f Makefile.linux」としてビルドし、「su」でrootになってから「make -f Makefile.linux install」でインストールすればいい。

このtarボールにはSPECファイル（calc.spec）が含まれているが、そのままだとパッケージ作成時にエラーが発生する。そこで、筆者がSPECファイルの記述やtarボールのディレクトリ名などを修正して作成したRPMソース・バイナリパッケージを本誌付属CD-ROMに収録した。バイナリパッケージは、「su -」でrootになってから、「rpm -Uvh calc-2.11.5-4.5.i386.rpm」でインストールできる。

### ●基本的な演算を行う

ktermなどのコマンドラインで「calc」とすると、プロンプト「> 」が表示されて入力待ちになる（**画面1**）。まずは、四則演算（＋、－、＊、／）などの演算子を使った簡単な計算式を入力してみよう。Enterキーを押すと計算結果が表示される。

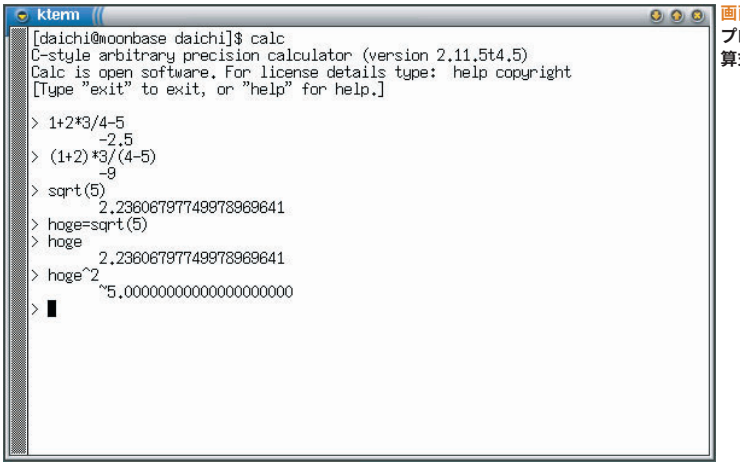
```
> 1+2*3/4-5
        -2.5
```

通常の数式と同様に、「＊」や「／」は「＋」「－」よりも優先され、優先順位が同じなら左から右に向かって順番に評価される。こうした評価順を変えるには、

```
> (1+2)*3/(4-5)
        -9
```

のように、優先したい部分をカッコで囲えばいい。

このほか、答えが整数になる割り算「//」や、余りを求める「%」、べき乗「^」、各種のビット演算子なども用意されている（**表1**）。




画面1　「> 」というプロンプトに対して計算式を入力しよう。

なお、計算式の入力の際には、bashと同じキー操作でコマンドラインの編集や過去の入力内容（履歴）の呼び出しを行える。たとえば、前回の入力内容をCtrl－pキーで呼び出し、カーソルキーやBackSpaceキーで内容を編集して、Enterキーで再度計算するといったことが可能だ。

### ●組み込み関数や変数を利用する

calcには、三角関数などの算術関数やC言語風の文字列操作関数など、さまざまな「組み込み関数」が280個以上も用意されており、何の準備もなし

| | |
|---|---|
| + | 足し算、単項プラス |
| - | 引き算、単項マイナス |
| * | 掛け算 |
| / | 割り算 |
| // | （整数の）割り算 |
| % | 剰余（余り） |
| ^ | べキ乗 |
| ** | （同上） |
| << | 左へのビットシフト |
| >> | 右へのビットシフト |
| \| | ビットOR |
| & | ビットAND |
| ! | 論理NOT |
| = | 代入（入れ子可）※ |
| ?: | 条件制御付き三項演算子 |
| ++ | インクリメント |
| -- | デクリメント |
| \|\| | 論理OR |
| && | 論理AND |
| == | 等しい |
| != | 等しくない |
| <= | 等しいか小さい |
| >= | 等しいか大きい |
| < | 小さい |
| > | 大きい |

表1　calcの主な演算子
※このほか、演算結果を代入する+=、-=なども用意されている。

に使うことができる。

たとえば、5の平方根を求めるには、組み込み関数sqrt()を使って、

```
> sqrt(5)
        2.23606797749978969641
```

とすればいい。

組み込み関数の一覧は、

```
> help builtin
```

とすると、lessを使って表示される（**画面2**）。

また、計算結果を保存するグローバル変数も、宣言などの準備なしに使用可能だ。たとえば、

```
> hoge=sqrt(5)
```

とすると、変数hogeが自動的に用意され、計算結果が格納される。その後はいつでもhogeの値を利用できるようになる。

### ●関数の定義と繰り返し処理

自分で関数を定義するには、defineコマンドを使用する。たとえば、

```
> define hoge(a)
>> {
>>   return 2^a;
>> }
```

とすると、関数hoge()が定義され、

```
> hoge(8)
        256
```

のように、組み込み関数と同様に使えるようになる。関数の定義内でのみ有効なローカル変数を宣言したり、引数省略時のデフォルト値を設定することも可能だ。

さらに、C言語風のif／switchステートメントによる分岐や、for／whileステートメントによる繰り返し、printf()やputs()といった組み込み関数などを組み合わせると、数学的な関数にとどまらない柔軟な処理が可能になる。各ステートメントの詳細は、

```
> help statement
```

で画面に表示されるヘルプを参照されたい。

### ●スクリプトを実行する

ユーザー関数やグローバル変数の定義などをキー入力する代わりに、これらが書かれたスクリプトファイルを読み込むこともできる。スクリプトファイルの読み込みにはreadコマンドを利用する。たとえば、

```
> read myfunc
```

とすると、カレントディレクトリのmyfunc.calが読み込まれ、キー入力した場合と同様に実行される。

指定したファイルがカレントディレクトリに存在しない場合は、環境変数CALCPATHに設定されたディレクトリ、/usr/share/calcの順にファイルが検索される。

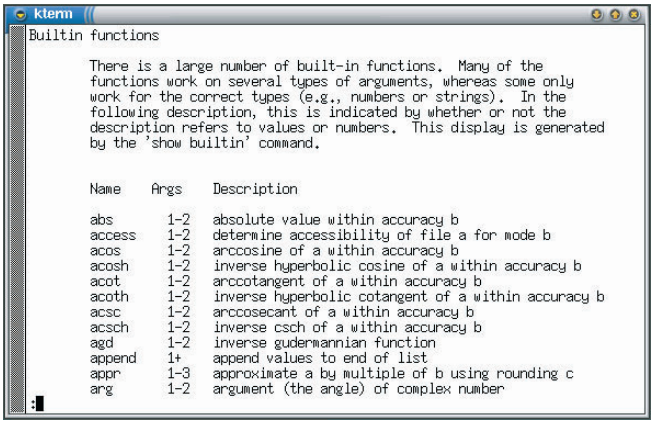
なお、/usr/share/calcには、サンプルスクリプトが多数用意されているので、自分で関数を作成する際の参考にするといい。たとえば、

```
> read pi
```

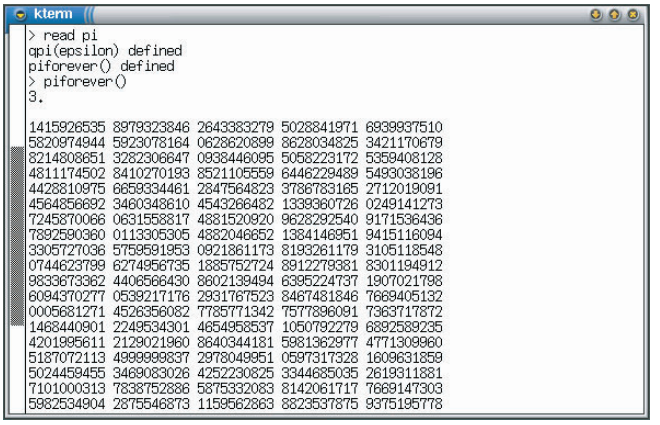
としてpi.calを読み込むと、qpi()とpiforever()という2つの関数が定義される。この状態で、

```
> piforever()
```

とすると、円周率πの値が永久に出力される（**画面3**）。




画面2　calcには280個以上の組み込み関数が用意されている。




画面3　スクリプトpi.calを読み込んで、関数piforever()を実行したようす。

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：0.7e　　ライセンス：GPL

http://wolfpack.twu.net/SearchAndRescue/
http://wolfpack.twu.net/YIFF/（YIFF2）
http://wolfpack.twu.net/libjsw/（libjsw）

Wolfpack's Search And Rescue（以下Search And Rescue）は、ヘリコプターを操縦して、さまざまな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。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。実行には3Dライブラリの Mesa、サウンドライブラリのYIFF、ジョイスティックライブラリの libjswが必要となる。

## ●ビルドとインストール

### (1) ライブラリのインストール

MesaについてはRPMパッケージなどで用意されていることが多いので、YIFFとlibjswについて説明する。

YIFFのtarボールには、サーバやユーティリティのソースも含まれており、「su」でrootになった状態で「./build_and_install」とすれば、まとめてビルドとインストールが行われる。インストール後に、「cp yiff/yiffrc /usr/etc」として、設定ファイルをコピーしておこう。

libjswは、tarボールの展開先で「cd libjsw」→「make -f Makefile.Linux」としてビルドし、「su」でrootになってから「make -f Makefile.Linux install」としてインストールする。ジョイスティックドライバの読み込みやキャリブレーションについては付属のINSTALLを参照されたい。

### (2) Search And Rescueのインストール

Search And Rescueは、tarボールのみ配布されている。まずは、「./configure Linux」として、プラットホームの判定や、ライブラリの存在チェックを行う。なお、YIFFとlibjswが存在しない場合もビルドは可能で、それぞれの機能が無効になるだけだ。

あとは、「make」でビルドし、「su」でrootになってから「make install」でインストールすればいい。インストール先は/usr/gamesだ。環境変数PATHに/usr/gamesが含まれない場合は、ホームディレクトリの.bash_profile内のPATHの設定の末尾に「:/usr/games」を追加しておこう。

## ●初期設定とミッション選択

サウンド機能を利用する場合は、rootになった状態で「yiff&」としてYIFFサウンドサーバを実行しておく必要がある。

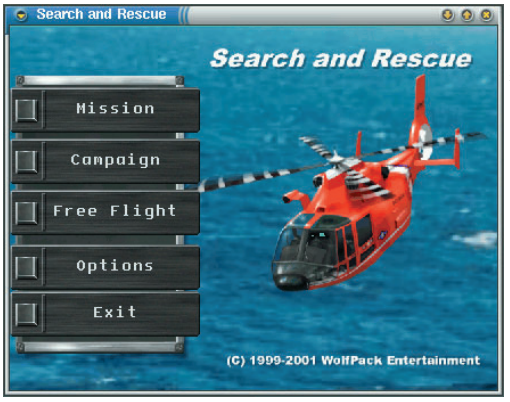
ktermなどで「SearchAndRescue&」として起動すると、メニュー付きのタイトル画面が表示される（**画面1**）。ウィンドウサイズは自由に変更できるので、3Dアクセラレータが有効な場合はもっと大きくするとよい。

初めて起動した場合は、[Options]ボタンを押して、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初期設定を行う。変更した設定は~/.SearchAndRescue以下に保存されるので、次回からは変更の必要はない。

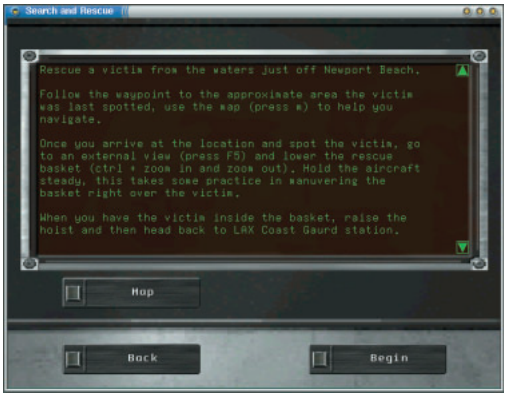
初期設定のままだとヘリのローター音などが出力されないので、[Sound]の設定で、[Sound Level]を「Events, Engine and Voice」に変更する。また、3Dアクセラレータがない場合は、[Graphics]の設定で、[Object texure]以外のスイッチをすべてオフにすると描画処理が軽くなる。

## ●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。現在のところ、トレーニング用の3種類のミッションをプレイ可能だ。ミッションを選択し、右下の[Briefing]をクリックして説明を受けよう（**画面2**）。[Begin]をクリックするとミッショ




画面1
タイトル画面。初めて起動した場合はオプション設定を行う。




画面2
ミッションのブリーフィングを受ける。マップの確認も可能だ。

画面3
ヘリのコックピットからの眺めが表示される。視点は切替可能だ。

画面4
F3キーでヘリの姿勢を確認しやすいスポット視点に切り替える。

ン開始だ。

ウィンドウには、ヘリのコクピットからの眺めが表示されている（**画面3**）。F3キーでヘリを外部から眺めるスポット視点に変更可能だ（**画面4**）。こうしたキー操作についてはF1キーで表示されるヘルプを参照されたい。

以下では、海で遭難した人を救助する「Rescue Over Water」に基づいて、ミッションの流れと主要なキー操作を説明しよう。

まず、PageUpキーでスロットルを開いて、エンジン出力（Thr）の値をヘリが浮上する70%以上にする。ある程度の高度（AGL）になったら、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。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、sawfishなどのウィンドウマネージャがCtrl－←／→キーをショートカットキーとして利用していると、向きを変える操作を行えない。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

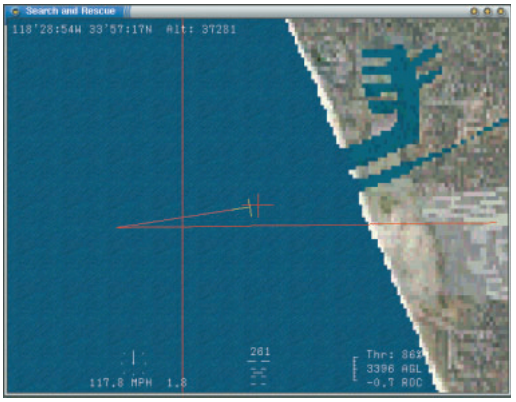
mキーを押してマップ表示に切り替えると、ヘリの位置と向きが逆T字で表示される（**画面5**）。出発地から目的地に赤い線が伸びているので、それを参考にしてヘリを目的地に向かわせよう。マップは＋／－キーで拡大／縮小する（Shiftキー併用で高速拡大／縮小）。F2キーを押すと、元のコクピット視点に戻る。

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%に戻し、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。

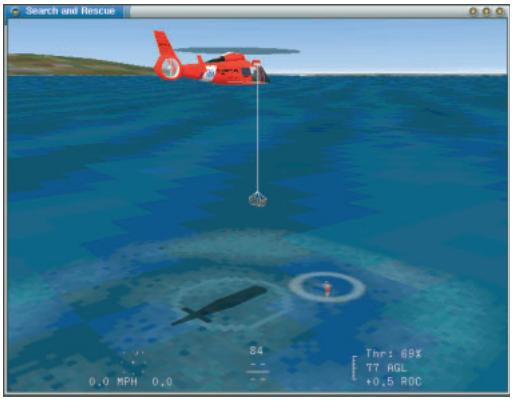
続いて、F5キーを押して救助用バスケット視点に切り替える。Ctrl－←／→キーで向きを変えつつ、海面に浮かぶ遭難者を探そう。このとき、－（マイナス）キーで画面をズームアウトすると見つけやすい。

遭難者を発見したら、ヘリの影を参考にしてゆっくり近づき、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**）。遭難者とバスケットが重なるように、ヘリの位置を微調整しよう。うまく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、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、次の目的地であるヘリポートを目指す。

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となる。



画面5
マップ画面で現在位置と救助に向かう方向を確認しよう。



画面6
救助用バスケットを慎重に遭難者のすぐ側まで下ろす。

## 自分の駒を4つ並べる対戦パズルゲーム

# Gnect

バージョン：1.0.1　　ライセンス：GPL

http://homepages.ihug.co.nz/~trmusson/gnect.html

Gnectは、二人のプレイヤーが交互に自分の駒を並べる対戦パズルゲームだ。ルールは簡単だが、パネルが縦置きなので上下関係を考慮して配置しなくてはならない。人間同士およびコンピュータとの対戦が可能で、コンピュータの思考ルーチンが最強の設定だと、先手でも勝つことは非常に難しい。パネルや駒の外見を切り替えるテーマもいくつか用意されている。実行にはGNOMEが必要だ。

### ●ビルドから起動まで

Gnectは、tarボールとRPMパッケージの両方が配布されている。RPMパッケージはRed Hat 7用なので、Red Hat 6系ディストリビューションを使っている場合はtarボールを利用したほうがよいだろう。ビルドとインストールはconfigureスクリプトを利用する一般的な手順だ。

ktermなどのコマンドラインで「gnect&」とするか、GNOMEメニューの[プログラム]－[ゲーム]－[Gnect]を選択するとウィンドウが開く。[新規]ボタンを押すとプレイ開始だ。

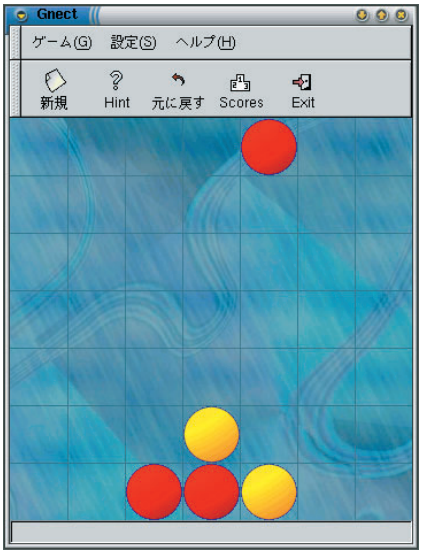
### ●自分の駒を4つ以上並べよう

ルールは簡単で、「二人のプレイヤーが7×7の縦置きパネルに交互に駒を落とし、縦／横／斜めのいずれかの方向に4つ以上先に並べたほうが勝ち」というもの。

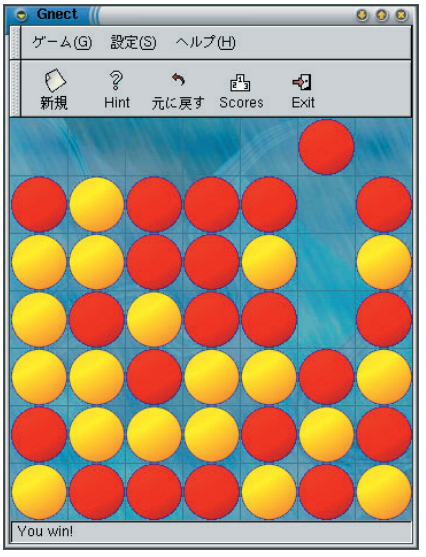
操作にはマウスまたはキーボードを利用する。駒を落とす列をマウスでクリックすると、その列に自分の駒（赤）がひとつ落ちる（**画面1**）。キーボードの場合は、最上部の駒をjキーとlキーで左右に移動させ、kキーで駒を落とす。このように、パネルの最上部は駒を落とす列を指定するために使われるため、縦方向に配置できる駒は実際には6個までだ。

コンピュータプレイヤーの強さは、初期設定のままなら勝利は可能なレベル（**画面2**）。ただし、後手から始めると結構厳しい。

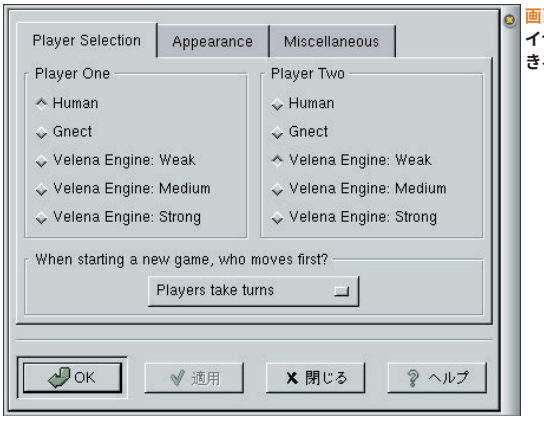
先手・後手の変更や、人間同士での対戦、コンピューターの強さの変更などは、[設定]－[設定]で開くダイアログで行う（**画面3**）。また、パネルや駒の外見を変更するテーマも9種類用意されている（**画面4**）。



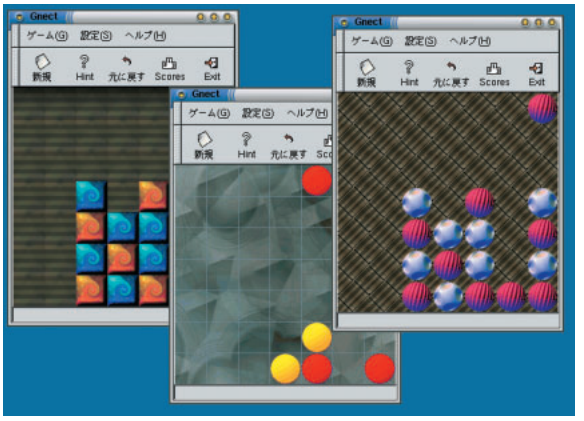

画面1　二人のプレイヤーが交互に自分の駒をパネルに落としていく。




画面2　先にこちらの駒（赤）が4つ以上縦／横／斜め方向に並べば勝ちだ。




画面3　コンピュータプレイヤーの強さなどを変更できる設定ダイアログ。



画面4　駒やパネルの外見を切り替えるテーマが9種類用意されている。

マウスの動きを認識してアプリを起動

# KGesture

バージョン：0.3　　ライセンス：GPL

http://www.slac.com/~mpilone/projects/kgesture.phtml

KGestureは、マウスの動き（ジェスチャー）を認識してアプリ起動などを行うソフトだ。1つのジェスチャーは、複数の直線運動の組み合わせで構成され、デスクトップでのマウスの動きがこうしたジェスチャーと一致すると、対応するアクションが実行される。KDE 2のクライアント間通信機能（DCOP）を利用して、実行中のアプリの動作を制御することも可能だ。動作にはKDE 2.1以降が必要となる。

### ●ビルドから実行まで

KGestureは、ソース一式をtar＋gzipしたtarボールのみ配布されており、ジェスチャーを認識するlibstrokeライブラリのソースも含まれている。「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

ktermなどのコマンドラインで「kgesture&」とするか、KDEメニューの[ユーティリティ]－[KGesture]を選択すると、パネル右端のシステムトレイに小さなマウスのアイコンが常駐する。右クリックするとメニューがポップアップし、認識動作の切り替えや設定変更などを行える（**画面1**）。

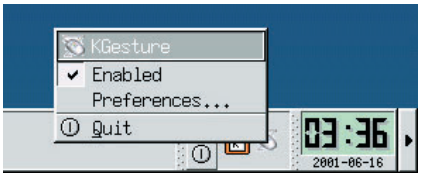
### ●ジェスチャーの登録と実行

メニューの[Preferences]を選択して設定ダイアログを開き、[Gestures]アイコンをクリックして、ジェスチャーページに切り替える（**画面2**）。ここには、登録済みのジェスチャーの名前が一覧表示されており、追加／削除／修正が可能だ。

[New]ボタンを押して、ジェスチャーを新規登録するウィザードを起動しよう。最初にジェスチャー名を設定すると、ジェスチャー登録画面に切り替わる（**画面3**）ので、左ボタンを押したまま直線移動（ストローク）を連続して行おう。通常は、3個以上のストロークでジェスチャーを構成することが望ましい。また、誤認識を防ぐため、同じジェスチャーを3回繰り返して登録する必要がある。

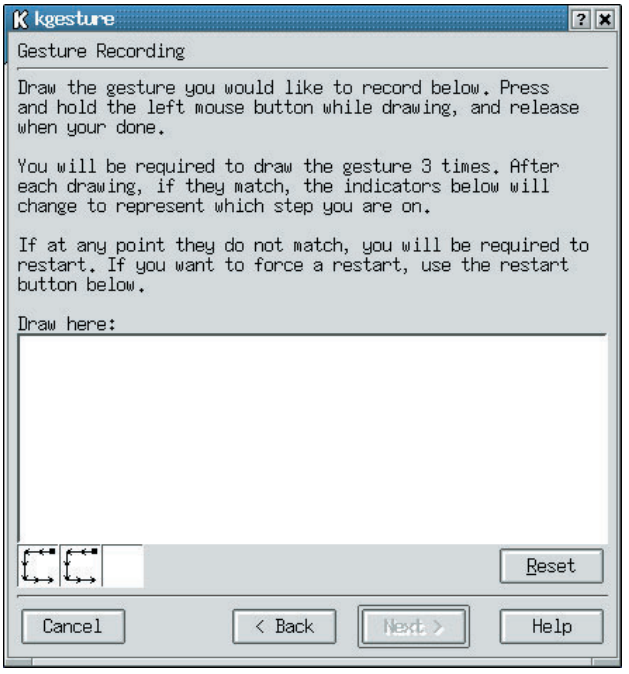
3回とも同じジェスチャーと認識されると、アクション設定画面に切り替わる。単に指定したアプリを起動するだけでなく、KDE 2のデスクトップ・コミュニケーション・プロトコル（DCOP）を利用して、実行中のアプリの動作を制御することも可能だ。

登録したジェスチャーを利用するには、KDEのデスクトップ画面でマウスを移動させればいい（左ボタンを押す必要はない）。KGestureが認識すると、ジェスチャーの名前と図がポップアップした後、対応するアクションが実行される。ジェスチャー動作を行う前後にしばらくマウスを静止させておくことがコツだ。




画面1
タスクトレイのアイコンを右クリックしてメニューを開く。


画面2
設定ダイアログには登録済みのジェスチャーが一覧表示される。




画面3
ウィザード形式のジェスチャー登録画面。同じ動作を3回行う。

Software Catch Up

Javaで記述されたXML対応のプレゼンテーション作成ソフトウェア

# Muffin

**ジャストシステムが精力的に展開する、Javaプラットフォーム向けオフィススイートの一角を成すソフトウェアがMuffinだ。プレゼンテーション作成ソフトとして使いやすくまとめられたインターフェイスは、長年のノウハウといえるだろう。**

文：塩田紳二
*Text : Shinji Shioda*

| | |
|---|---|
| 価格 | 未定 |
| 問い合わせ先 | ジャストシステム<br>03-5412-3939<br>http://www.justsystem.co.jp/ark/ |

Muffinはジャストシステムが開発中のJavaで記述されたプレゼンテーション作成ソフトウェアだ（**画面1**）。同社のJavaアプリケーションには、すでに出荷中の一太郎Arkがあるが、現在は以前紹介したChoco（表計算）と、このMuffinを開発中である。いまのところ、Technology Preview版（以下、TP版）を配布中で、とりあえずプログラムの概要を知ることが可能だ。

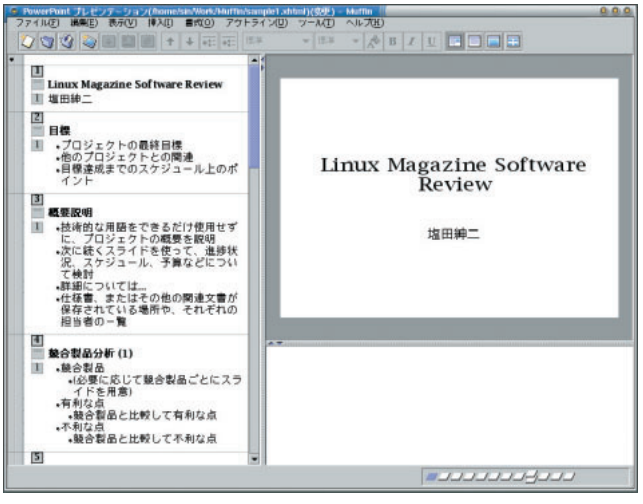
今回は、このMuffinを取り上げることにする。ただし、前述のようにまだ製品になっているわけではないので、いくつか未実装の機能などもある。このため、ソフトウェアを紹介するにとどめ、細かい評価は行わず全体的な方向性についてのみ評価することにする。

## プレゼンテーションソフトとは

ビジネスにおいては多くの資料を作成する必要があるが、プレゼンテーションのときに配布したり、スライド、液晶プロジェクタなどに表示させる資料を作ることも多い。これらは、一般的には「プレゼンテーション資料」と呼ばれ、ワープロなどで作成するレポートなどとはまた別の作り方が必要となる。

米国などでは、プレゼンテーションの成否がビジネスの重要なポイントとなるため、コンピュータ登場以前からリバーサルフィルムなどを使うスライドを作成するなど、かなり費用をかけることが一般的であった。このような背景からプレゼンテーション資料作成ソフトが開発され、ワープロなどと並ぶ地位を占めていた。

プレゼンテーションソフトが登場した当初はカラープリンタも高価だったため、レーザープリンタを使ってモノクロのOHPを作ったり、作成したデータを業者に持ち込むなどしてスライドやOHPを作成していた。しかし、コンピュータの画面解像度が上がり、LCDプロジェクタなどが登場すると、コンピュータを使って直接プレゼンテーションを行うことが一般的となった。なお、かつてスライドを作ってプレゼンテーションを行っていた経緯から、プレゼンテーションソフトでは作成したページを「スライド」と呼ぶことが多い。実際には、PCの画面やOHPへの出力だったりするのだが、すべてこう呼んでいる。ただし、出力されるものはその形態により、また別の呼び名がある。ハンドアウトはいわゆる「配付資料」で、これはレーザープリンタなどに対して出力を行い、1ページに複数のスライドを配置する。また、個々




画面1　Muffinのメインウィンドウ
Muffinはすでに出荷中の一太郎Arkと同じく、Javaで記述されたプレゼンテーションソフト。

のスライドに対してメモを付加し、それを同時に印刷するものは「ノート」、「スピーカーズノート」などと呼ばれる。

余談だが、現在最も広く使われているWindows上のMicrosoft PowerPointは、かつてMacintosh用のアプリケーションとして作られ、開発元の会社をマイクロソフトが買収したことで同社の製品となった。

国内では、配布資料やボードを使ったプレゼンテーションが多く行われており、ワープロを使って配布資料を作ることが長らく続いたが、最近ではプレゼンテーションソフトを使うことが多くなってきた。もっとも、国内においてプレゼンテーションソフトが登場した当時は、プレゼンテーション資料作成の意味を理解せず、単なる描画ソフトとして使っていた人も多かったという。

プレゼンテーション資料は、説明を行いながら見せることを前提としているため、話の要旨を簡潔にまとめたものになる。ワープロで作成する文書のように長い文章を書かずに、短い文章を使い、ポイントだけを記述する。また、説明のストーリーに応じてページを作り、必要ならば図やイラストなどを交える必要もある。たとえていえば、紙芝居の絵のようなものが「プレゼンテーション資料」である。

このため、多くのプレゼンテーションソフトには、図とそれほど長くない文章をうまくレイアウトさせるような仕組みを持つ。また、各ページの統一感や、話題に合ったイメージを適用するような仕組みもある。

これまで、いくつかのLinux上のオフィススイートを見てきたが、プレゼンテーションソフトに関してはマルチページ対応のドローイングツールとして、ドローイングソフトのエンジン部分を共有したものが多かった。これは、主にレイアウト機能を提供するものだ。

しかし実際には、プレゼンテーションのストーリーを作るアウトライン機能や、1つの作成物からスライドや配付資料（ハンドアウト）、話者用のノートといったものが作成できるという機能が必要になる。また、パソコンを使ってオンラインでプレゼンテーションできるような機能（スライドショー）や、ページ切り替え効果なども必要だろう。また、最近では、サウンドや箇条書き項目の表示効果なども必要になるだろう。

つまり、プレゼンテーションソフトとは、プレゼンテーション資料作成の機能とプレゼンテーションそのものを行う機能の大きく2つの機能から構成されているものといえる。

## Muffinの概要

Muffinは未完成ながら、基本的な機能はひととおり備えている。まず、資料作成という点では、独自のアウトラインエディタを持ち、これを使って全体のストーリーを作成すれば資料ができあがるようになっている。また、HTML（XHTML）での出力が可能で、Netscape NavigatorなどのWebブラウザを使うことでスライドショーが行える。なお、最終的には専用のスライドショー機能が提供されるという。このほか、クリップアートが附属し、簡単なイラストであればこれを使うこともできるようだ。

作成面での特徴は、

**①アウトライン編集機能**
**②パーソナライズドメニュー**
**③スライドの並列配置**

などがあり、プレゼンテーション関連機能としては、

**①XHTML＋JavaScriptによるWebブラウザでのプレゼンテーション**
**②上下反転表示**

などがある。

Muffinでは、各スライドは「タイトル」と「本体」からなり、基本的なレイアウトが決められている。本体部分は、箇条書きや文章、図版（クリップアートや画像ファイル）で構成される。ただし、現時点で公開されているTP版には図形の描画機能がない。製品版ではSVG（Scalable Vector Graphics。W3Cで定めたベクトル画像記述言語。XMLで記述されている）によるグラフィック描画機能が提供される予定だ。

作成したプレゼンテーション資料は一太郎Ark同様、XMLとして管理し、ファイル形式もXMLとCSS（Cascading Style Sheet）を利用するXHTML形式を使っている。一太郎Arkでは、XMLを扱うDOM（Document Object Model。W3Cで定めたXMLなどを扱うための標準APIセット）を内部的に持ち、これを操作する形で編集作業を行うような構造になっている。

出力がXMLであるため、作成したプレゼンテーション資料をXMLを利用可能なほかのアプリケーションで扱える可能性がある。

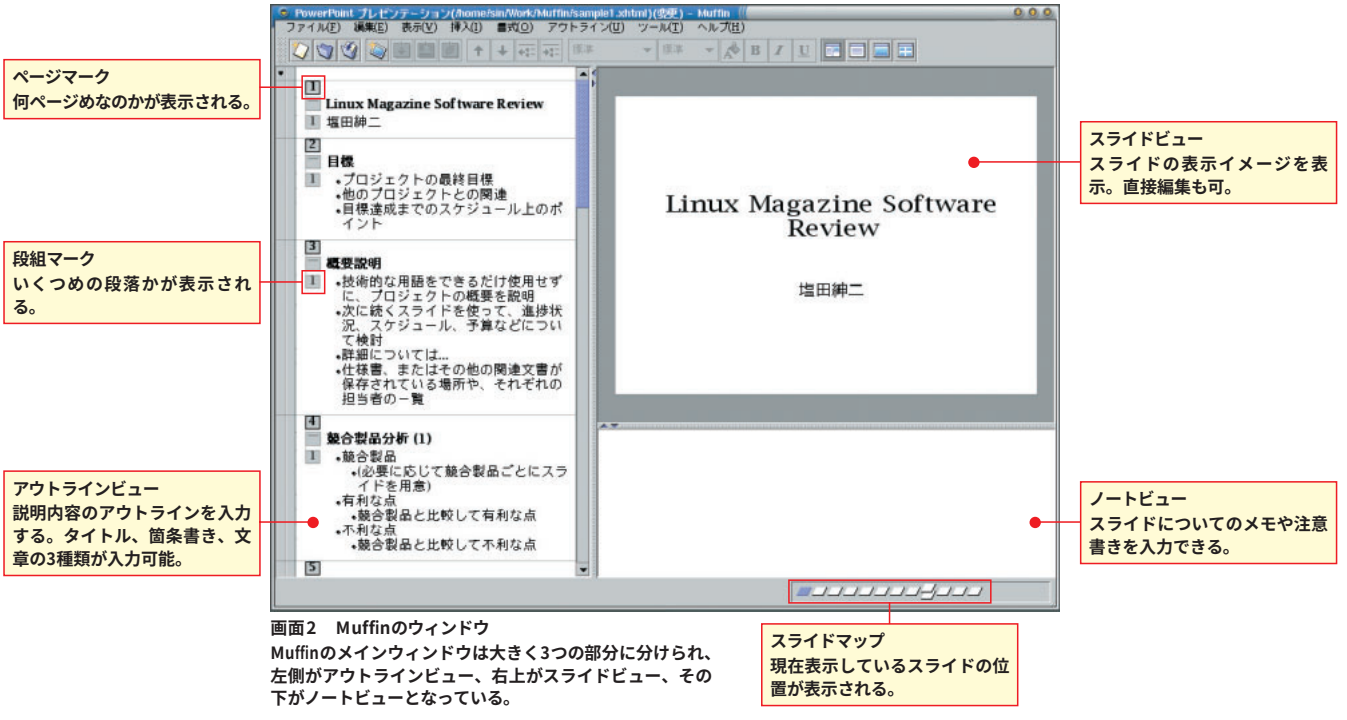
## GUI周り

MuffinはJavaを使って記述されており、基本的にはJFCを使ったGUIとなるが、同社の一太郎Arkとの統一性も配慮されている。ウィンドウはタイトルバーの下にメニューバー、ツールバーを持ち、最下部にステータスバーを

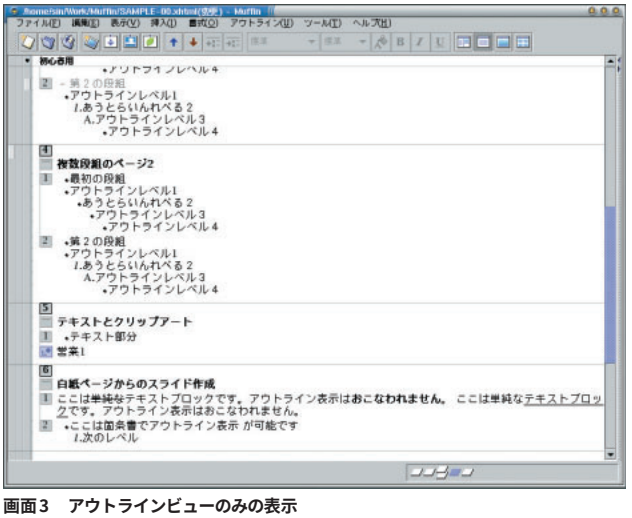
持つ（**画面2**）。

メインのウィンドウはいくつかの表示モードを持ち、スライドの作成や並べ替えなどの作業に応じて便利なように考慮されている。機能上はいくつかの表示モードになるのだが、実際にはアウトラインやスライド表示を行うモードと、スライドの一覧と各スライドのプロパティを表示する2つのモードとなる。

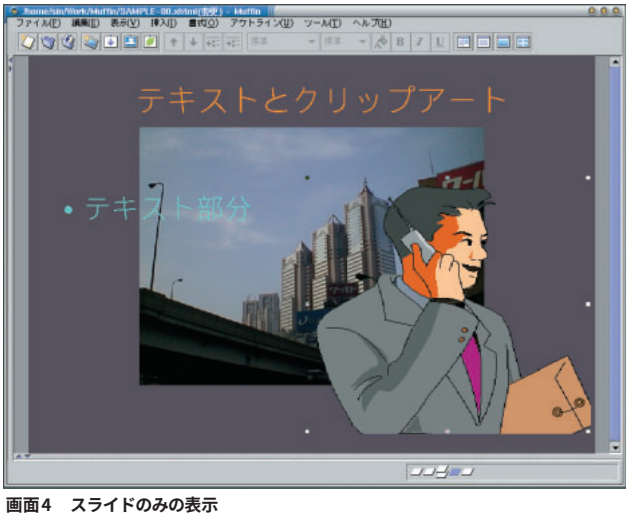
1つはウィンドウ全体を3つの領域に分割し、左側がアウトラインを表示する「アウトラインビュー」、右上がスライドの表示される「スライドビュー」、右下が個々のスライドのメモを表示する「ノートビュー」という領域にしたもの。この3つの領域は、メインウィンドウを3つのフレームに分割して表示可能で、それぞれの領域の大きさを調整可能だ。ただし、表示位置や分割方法は変えることができず、アウトラインビューは常にウィンドウの左半分、スライドビューは右上側、ノートビューは右下側と決まっている。また、このモードで、アウトラインビューとスライドビューはウィンドウ全体に拡大表示可能で、それぞれ「アウトライン」、




**画面2　Muffinのウィンドウ**
Muffinのメインウィンドウは大きく3つの部分に分けられ、左側がアウトラインビュー、右上がスライドビュー、その下がノートビューとなっている。



**画面3　アウトラインビューのみの表示**
メインウィンドウは、ツールバーのボタンでアウトラインビューのみの表示にすることが可能だ。




**画面4　スライドのみの表示**
ツールバーのボタンでスライドのみの表示にすることもできる。

「スライド」というメニュー項目やツールバーボタンが用意されている（**画面3・4**）。

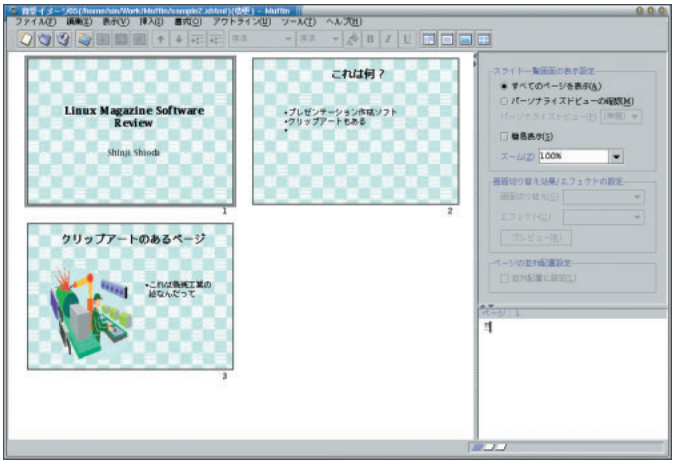
もう1つは、ウィンドウ左側にスライドを並べて表示する「スライド一覧」（**画面5**）である。こちらの場合、ウィンドウの左半分にスライドを並べて表示し、右側に選択されたスライドの表示方法などを指定する領域と、ノートを表示する領域が上下に並ぶ。ただし、TP版ではスライド切り替え時の効果設定などは未実装のため、並びやパーソナライズドビュー、スライドの並列配置（後述）の確認などしか行えない。

ウィンドウの下にはステータスバーがあるが、そこには「スライドマップ」と呼ばれる領域がある。これは、スライド1枚1枚に対応した矩形のシンボルで、現在表示しているスライドを表すほか、これをクリックすることで該当のスライドを表示させることができる。なお、スライドが並列配置になっている場合にはシンボルが上下に並ぶ。

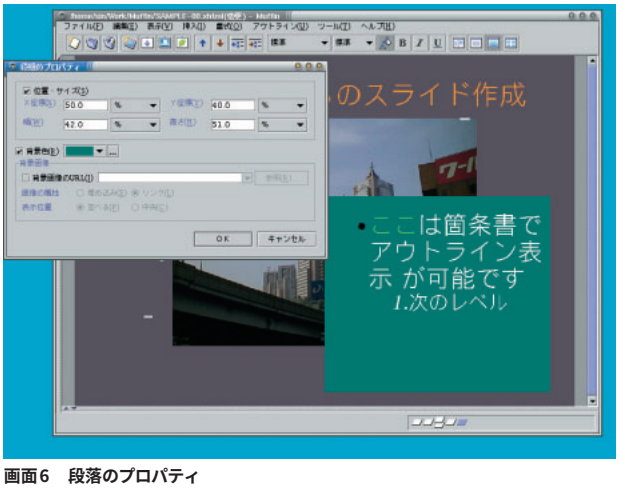
基本的な操作は、メニューやツールバーから行うが、Muffinではキーボードショートカットも多数登録されており、キーボードだけでもかなりの操作が可能である。このあたりは、長年ワープロでユーザーインターフェイスを研究してきた同社ならではという感じである。

## スライド上のオブジェクト

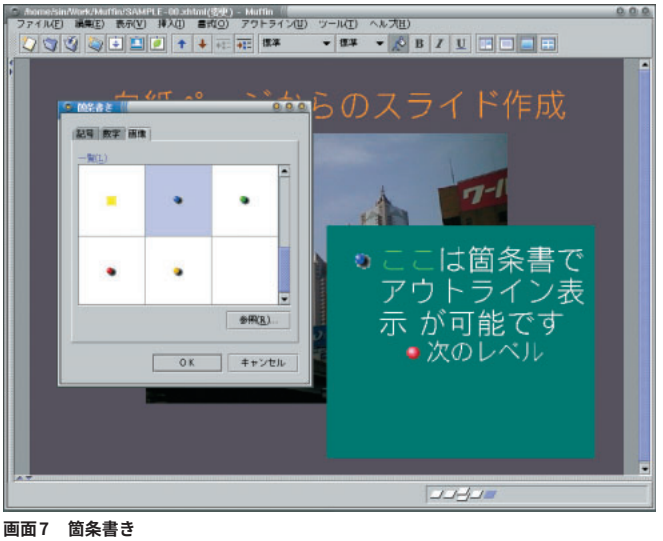
スライドはあらかじめサイズを決めた領域内に、文字やグラフィックなどを配置することで作られる。ここだけを見れば、ドローツールとまったく同じである。スライドは背景部分と、その上のオブジェクトから構成される。オブジェクトとしては、テキストブロック（段組み）、タイトル（特殊なテ



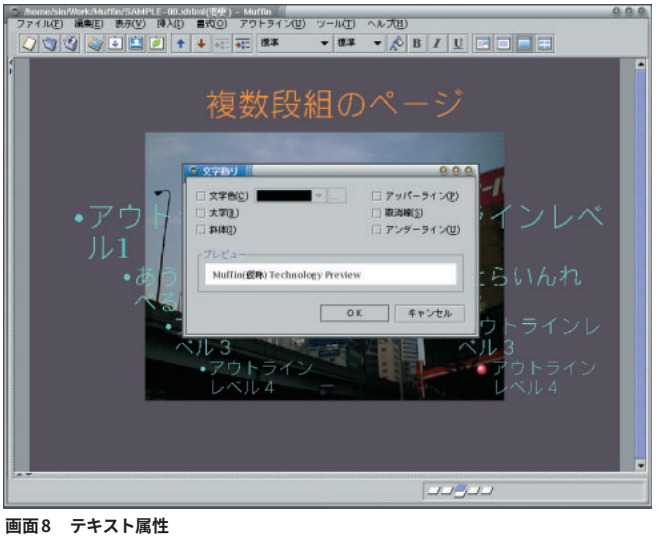

**画面5　スライド一覧**
スライドの一覧を表示する場合、パーソナライズドビューなどを適用させることも可能。




**画面6　段落のプロパティ**
個々のテキストブロック（段落と呼ばれる）は、個別に背景や背景画像などを設定することが可能。




**画面7　箇条書き**
箇条書きでは、行頭のBulletの指定や番号を付ける指定が行える。




**画面8　テキスト属性**
個々の文字について、色や太字、斜体などの属性を指定できる。

キストブロックといえる)、クリップアート、画像ファイルがある。なお、TP版ではいわゆる線画などのベクトル描画機能がないが、SVGを採用する予定なので、最終製品では図形の描画が可能になるはずである。

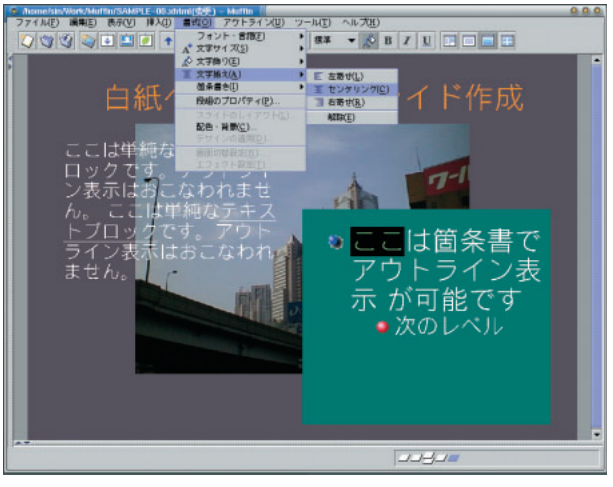
テキストブロックは、位置、サイズの指定のほかに、背景色や背景画像を個別に指定できる(**画面6**)。箇条書きの場合、テキストの頭に付く記号(一般的にBulletという)を丸や四角、あるいはグラフィックとして指定することもできるし、数字やアルファベットにすることも可能だ(**画面7**)。1つのテキストブロック内では各レベルごとに指定が可能で、第1レベルをグラフィックによるBulletとし、第2レベルを数値、第3レベルをアルファベットなどと指定できる。なお、TP版ではアウトラインレベルのインデント量などについては設定できないようである。

テキストには、ボールド、イタリック、アンダーライン、アッパーライン、取消線、文字色の飾り(**画面8**)と、サイズ指定(上付き、下付きを含む)、文字揃え(右寄せ、左寄せ、センタリング(**画面9**)、フォント(ただしJavaの設定による。**画面10**)が指定可能だ。

背景は、デフォルトのテキスト、タイトルの表示色、背景色、背景画像から構成される(**画面11**)。通常、各スライドの背景設定は、「マスター」と呼ばれる設定を引き継ぐようになっている。ただし、これは各スライドごとに個別に指定が可能で、たとえばマスターで背景画像を指定したとしても、個々のスライドで別の背景画像を指定したり、背景画像を入れないという設定も可能である。

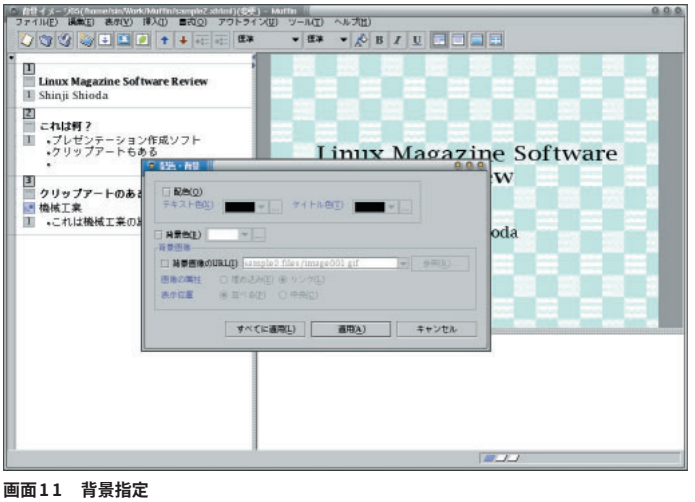
マスターは各ページの書式を統一するためのもので、タイトルページ用と本文ページ用(**画面12**)、および配付資料用(ただし、TP版ではメニュー項目はあるものの選択できない。おそらく未実装と思われる)がある。

ここでは背景の設定のほか、タイトルと本文のテキスト属性の設定が行え

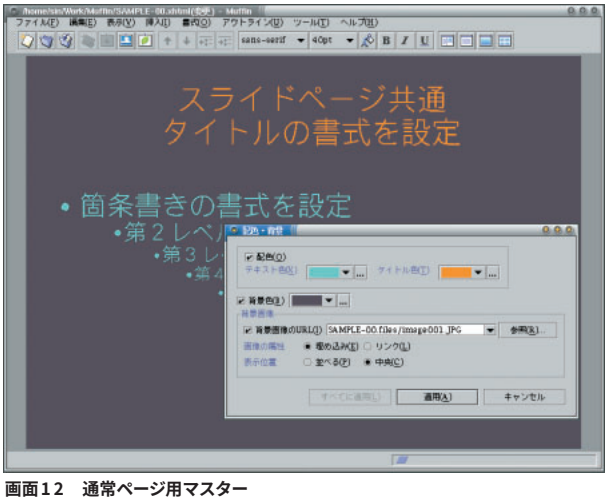



画面9　文字揃え
段落内での文字揃えは、左寄せ、センタリング、右寄せの3つが指定可能。


画面10　フォント指定
MuffinはJavaアプリケーションであるため、Javaの管理するフォントを利用する。ただし、OS側が管理しているフォントを指定することもできる。




画面11　背景指定
各ページの背景や背景画像、文字色などは個別に指定可能。特に指定しないとマスターでの指定が利用される。




画面12　通常ページ用マスター
マスターは、標準の背景設定やテキスト設定を行うためのものだ。個々のスライドで特に指定を行わなければこの設定が使われ、スライド全体の書式などを統一できる。

る。最初にマスターを指定してしまえば、新規ページの作成はこのマスターを元にして行われ、すべてのページで同じ背景とテキスト属性が設定される。なお、前述したように個々のページで設定を変えることも可能だ。また、編集途中でマスターを編集すると、個別指定していないすべてのページのデザインが変更になる。

## 個々の機能を見る

では、次にMuffinの特徴的な機能を見ていくことにしよう。

### ■アウトライン機能

Muffinはスライドを作るとき、アウトラインビューを使ってワープロのように文字入力を中心として作成することもできる。これまで紹介してきたLinux用オフィススイートでは、ドローツールのエンジンを使ったプレゼンテーションソフトが多く、スライドの作成はドローツールの図版作成方法と同じく、画面上に部品を配置していくような方法を採るものがほとんどであった。

しかしPowerPointなど、本格的なプレゼンテーションソフトではこうしたアウトラインからの入力、作成が行えるのが普通で、その意味では単独で提供されるMuffinは、これまで紹介したLinux上のオフィススイートのものと一線を画するものだ。

アウトラインビューでは、スライド1枚がワープロの1ページのように区切って表示が行われる。スライドはタイトルと本体からなり、本体は複数の段組み（スライド上のテキストブロックに対応）や図版に対応する。

個々の段組みは、単純なテキストまたは箇条書きの2種があり、箇条書きの場合にはアウトライン機能が利用でき、テキストにレベルを設定して表示を行わせることができる。

アウトラインビューはテキストのみの表示となるため、スライド自体の表示よりも高速に行えるというメリットがある。また、本来の文書作成とレイアウトの作業を分離できるため、文書の作成のみに専念できるというメリットもある。

### ■パーソナライズドビュー

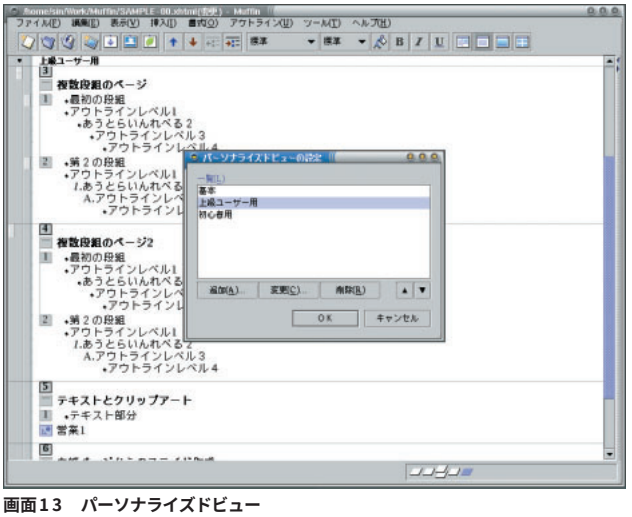
パーソナライズドビューとは、スライド中のテキストやスライドの表示、非表示を切り替えるものだ（**画面13**）。たとえば、同じ内容で違う相手に対して利用するプレゼンテーション資料を作る場合、この機能を使うことで本文テキストなどを切り替えることができる。また、ビューとして複数の設定が可能なため、1つのスライドをいくつものパターンで共有させることが可能になる。

操作としては、アウトラインビューの左側にバーがあり、該当するテキストの横にある部分をクリックすることでオン／オフが切り替わる。オフにしたテキストや画像は非表示となり、その部分が詰められて表示される。

複数のパーソナライズドビューを設定すると、このバーが複数になり、現在選択されているビューにマークがつく。なお、アウトラインビューエリアの一番上の部分には、選択中のパーソナライズドビュー名が表示される。

### ■スライドの並列化

Muffinではスライドは線形に並べるだけでなく、同一ページ位置に複数のスライドを設定できる。これを使うことで、スライド全体をコピーすることなくその一部ページのみ差し替えるこ



画面13　パーソナライズドビュー
パーソナライズドビューは、テキストや図版、スライド自体の表示を制御するもので、複数の設定（ビュー）を定義できる。

画面14　スライドの並列配置
Muffinでは同一ページ位置に複数のスライドを配置できる。この機能とパーソナライズドビューを組み合わせると、並びの違う資料を1つのファイルで実現できる。

となどが可能である。ただし、あくまでも別ページとして扱われるため、たとえば書式の設定やレイアウトなどは個別に設定する必要がある。

スライドを並列配置すると、スライドマップ上はスライドを表すマークが上下に並び、スライド一覧ではページ番号の下に並列配置されていることを表す赤い帯が表示される（**画面14**）。

なお、このスライドの並列化とパーソナライズドビューを組み合わせることで、複数のシナリオを1つのプレゼンテーション資料ファイルで実現できる。パーソナライズドビューは、スライド自体の表示を制御することができるため、表示に利用するスライドをパーソナライズドビューで指定しておけば必要なスライドのみを表示させることができるわけだ。

### ■スライドの出力

作成したスライドは、HTML形式のファイルに出力することが可能である（**画面15**）。このとき、Webブラウザ上でスライドショーを行うためのJavaScriptなどが付加され、Muffinのない環境でもプレゼンテーションを行うことができる。

出力形式としては、目次やスライド、ノートなどをフレームで区切って出力するものや、単にスライドのみを表示するもの、目次とアウトライン、アウトラインのみの4つが選択できる（**画面16**）。このため、これを使ってプレゼンテーションを行うこともできるし、ユーザー向けの配布資料とすることも可能だ。

そのほか、文字コード設定や、指定されている画像を埋め込むか、元のファイル位置へのURLのみとするかといった指定ができる。画像ファイルに対して後者の指定を行うと、画像ファイルは出力HTMLと同じディレクトリに作られるサブディレクトリにすべて置かれ、各スライドHTML中のURLはその画像ファイルを示すように変更される。

### ■スライドの上下反転

TP版はスライドショー機能が未実装のため、最終的な形態がどのようになるかは不明だが、Muffinにはスライドを上下反対にして表示する機能がある。これは、ノートパソコンなどを使って、対面で直接プレゼンテーションなどを行う場合に利用するものだ。

つまり、ノートパソコンの液晶部分を奥に倒して相手に見えるようにし、それを使ってプレゼンテーションが行えるわけである。なお、この反転機能は単に表示のみの機能で、スライド自体は変更しない。また、実際に動かしてみたところ、フォントの表示などがうまくいかなかった。これはJava側の問題なのか、TP版ゆえの問題なのかは不明である。

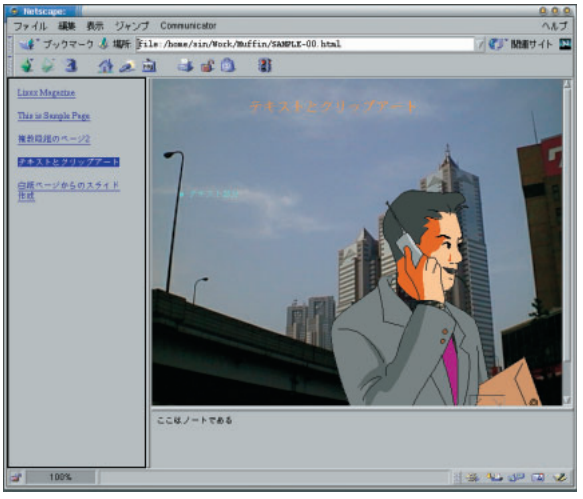
機種によっては、相手に見えるように液晶を倒すことができないノートパソコンもあるが、特に機材を用意することなく、簡単にプレゼンテーションするためには有効な機能だろう。

### ■プラグインによる機能拡張

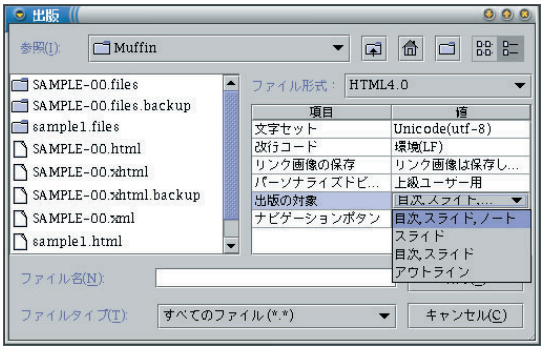
一太郎Arkでは、プラグインと呼ばれる専用形式のJavaプログラムを作ることで、一太郎Arkの機能を拡張することができた。たとえば、Microsoft Wordの文書ファイルの読み込みなどは、プラグインとして提供されている。Muffinにもこのプラグイン機能が装備されており、ユーザーが必要な機能を追加することが可能となる（**画面17**）。

もっとも、Javaのプログラミングと固有の形式や一太郎Ark、Muffin内部の構造などに熟知しないとプラグインの作成は難しいが、一般ユーザーレベルでは配布されているプラグインにより必要な機能のみを取り込めるというメリットもある。

現状では、Microsoft PowerPointのファイルから、アウトライン（テキスト）のみを取り出す「PowerPointコ




**画面15　Webブラウザへの出力**
Muffinで作成したプレゼンテーション資料は、Webブラウザで表示可能な形式に出力ができる。




**画面16　出版ダイアログ**
HTMLへの出力では、目次付きやスライドのみなど、いくつかの形式を指定可能。

ンバータ」と、作成した資料をDOMツリーとして表示する「DOMツリー表示プラグイン」（**画面18**）の2つが提供されている。

**■クリップアート**

Muffinにはスライド作成で利用できるクリップアートが附属する。これは、附属のクリップアートブラウザ（**画面19**）で選択してスライド内に挿入する。なお、TP版ではクリップアート自体はGIFファイルとして提供されている。

このクリップアートブラウザは、カテゴリ別などで最大9つのクリップアートをブラウズできる。また、検索機能があり、キーワードからクリップアートを検索することも可能だ。試しに「ペン」というキーワードで検索をかけてみると、ペンの絵を含む画像や「ペンキ缶」などが検索結果として表示された。これは、クリップアートファイルとは別に、キーワードなどが登録してあるXHTMLファイルがあり、これでグループ分けなどが指定されており、クリップアートブラウザはこれを読み込んで動作しているためである。これ自体が特殊なXMLブラウザとなっており、検索機能などがあるわけだ。

クリップアートの記述にまでXMLを使うあたり、XMLに対するかなりのこだわりが感じられる。

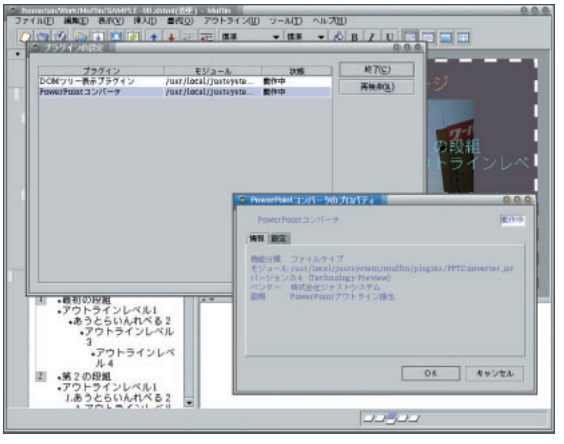
## 製品版に期待

Muffinは現在TP版であるため、いくつかの不具合や未実装の機能がある。このため、ここでは細かい部分ではなく、全体のコンセプトなどを評価してみることにする。

未実装の機能があるものの、プレゼンテーション資料作成のためのソフトウェアとしては基本的な機能を満たしている。ドローツールを使った簡易版のソフトウェアに比べると、アウトライン機能などにより作成も容易だろう。ソフトウェア自体の動作ではまだもたつきが感じられ、不必要なところでウィンドウ全体の再描画などが起こるが、こういった部分は最終製品ではチューニングされるだろう。若干気になるのは、多重に表示されるダイアログボックスで、最後に表示されるものがタイミングによってウィンドウの下に隠れてしまうことがある。このあたり、モーダルなダイアログボックス（閉じない限りほかの操作が行えないダイアログボックス）とするような配慮が必要と思われる。

パーソナライズドビューとスライドの並列配置は、1つのファイルで複数のシナリオが記述でき、同一テーマで複数のプレゼンテーション資料を作る必要がなく便利そうな機能である。

製品版が出てみないとわからない部分があるため製品版に期待というところだが、昨年7月にTP版が公開されて以来、その後音沙汰がない。そろそろベータ版のリリースなど、何らかの動きがあることを望みたい。



**画面17　プラグイン**
Muffinは、プラグインと呼ばれる特別な形式のJavaプログラムを使って機能を拡張することができる。

**画面18　DOMツリー**
標準で附属しているDOMツリーの表示。XMLによるスライドの定義をツリー形式で表示したものだ。

**画面19　クリップアートブラウザ**
クリップアートもXMLファイルで名称などが管理され、キーワードによる検索も可能。

# 隠喩としてのコンピュータ

## あのadminはどこへ消えた？

文：豊福 剛
*Text : Tsuyoshi Toyofuku*
*illustration : hmm*

ある事務所がフレッツADSLを導入するというので、電話工事業者の知り合いからルータの設定を頼まれた。事務手続きが遅れていて、まだ局側の回線の準備も整わず、プロバイダのIDも発行されていなかったので、結局、作業の当日には何もすることがなかったのだが、ひとつだけ困ったことがあった。

その事務所のLANにはWindows 98のマシンが5台ほど接続されていた。ハブにADSL用のルータを接続して、Webブラウザからルータのデフォルトの IPアドレスである192.168.1.1にアクセスしてみた。ところが、ルータにアクセスできない。pingをかけたらタイムアウトしてしまう。

そこで winipcfgコマンドを実行してみた。たしかにIPアドレスが設定されていたが、そのアドレスは192.168.x.xではない。それでは固定のIPアドレスかと思い、TCP/IPのプロパティを確認してみた。しかし、その設定は「IPアドレスを自動的に取得」になっているではないか。そこのLANにはADSLルータ以外にはDHCPサーバ機能を実行するマシンは存在しない。それなのに、マシンにIPアドレスが割り振られていたのだ。

### 169.254.x.xって何？

そのIPアドレスは169.254.x.xになっていた。

RFC1918ではプライベートIPアドレスは、10.0.0.0～10.255.255.255（クラスA）、172.16.0.0～172.31.255.255（クラスB）、192.168.0.0～192.168.255.255（クラスC）と定義されているから、このIPアドレスはプライベートIPアドレスではないはずだ。

一体どういうことになっていたのか。

謎は解けた。Windows 98にはAPIPA（Automatic Private IP Addressing）という機能があって、DHCPサーバが見つからないときは、169.254.x.xが勝手に割り振られてしまうのだ。このIPアドレスはLINKLOCALアドレスと呼ばれているそうで、IANA (Internet Assigned Numbers Authority) によって登録されているらしい。

このLINKLOCALアドレスはプライベートIPアドレスであるのか、非常に気になったので、これに関するRFCがないか探してみたが、見つからなかった。それで、インターネットドラフトを探してみたら、IANAによって登録

されているIPv4アドレスブロックに関するメモが見つかった（http://www.ietf.org/internet-drafts/draft-manning-dsua-06.txt）。このメモによると、奇妙なことに、APIPAについてはRFCになっていないにも関わらず、APIPAによるLINKLOCALアドレスの自動設定を無効にする方法はRFC2563になっているのである。ちなみにLINKLOCALアドレスについては、インターネットドラフトが公開されていた（http://www.ietf.org/internet-drafts/draft-ietf-zeroconf-ipv4-linklocal-02.txt）。

ともかく、謎のIPアドレスの原因はわかったので、ルータの設定はなんとかなりそうなのだが、それにしても、このAPIPAという機能、いかにもマイクロソフト的なセンスの産物というか、余計なお世話だと思う。

## Something Wicked happened!

このところ、ネットワーク関連のちょっとした作業が続いている。先日も、とある会社の知り合いから、サーバの調子がおかしいので、原因を調べてくれと頼まれた。聞けば、数年前にLinuxに詳しいアルバイトくんがサーバを立ち上げたものの、そのアルバイトくんが別の会社に就職してしまって以来、ずっと担当者不在の状態で動いていたらしい。これと似たような状況の会社、きっと少なくないだろうな。

とりあえずDNSサーバを調べてみたら、bindしか動いていないはずなのに、ディスクの空き容量が残っていないのである。その直接的な原因は、膨大なログファイルにあった。なんとそのDNSサーバはSPAMメールの中継に使われていたのだ。

bindのバージョンが4.9.5だったこともあって、既存の設定をいろいろいじるよりも、ゼロからインストールし直すことにした。たまたま予備のマシンがあったので、それに新しい環境を一式インストールし、適当なタイミングでマシンを切り替えることにした。

DNSの移行は比較的スムーズにいったのだけれども、メールサーバの移行は、思わぬところでトラブルが起きてしまった。というのも、移行用のマシンのNICが、VIA Rhineのチップだったのだ。

日曜日の午後、新しいメールサーバの設定を終えて、テストも無事成功して、やれやれ一件落着と思っていたら、

翌日の朝、メールが使えないという電話で起こされた。メールサーバが落ちたのではなくて、そのマシンのネットワーク機能そのものがダウンしているようだ。pingが届かないのである。

立ち上げ直してもらったメールサーバは、とりあえずIP到達可能になった。sshでログインして、/var/log/syslogを調べたら、次のようなメッセージが記録されていた。

```
kernel: eth0: Something Wicked happened! 001a.
kernel: eth0: Something Wicked happened! 001b.
kernel: eth0: Transmit timed out, status 0000, PHY
status 782d, resetting...
```

おいおい、eth0が原因かよ、というわけである。「何か邪悪なことが起きました」というメッセージも、なんだかなあ、である。もうちょっと気のきいたメッセージにはならないのか、と思いつつ、「something wicked happened」でgoogleに検索をかけたら、同じような現象がいろいろ報告されていた。

などと調べていたら、sshが反応しなくなってしまった。またダウンしてしまったようだ。この調子では、リスタートしても1時間もしないうちに、またダウンしてしまうだろう。そこで、いったん移行前のメールサーバに戻すことにした。

ドライバのバージョンに問題があったのかもしれないが、そこそこ負荷のあるメールサーバなので、NICを交換することで話をつけた。これまでいくつかLinuxサーバを立ち上げてきたが、いつも3Comの製品を使っていたので、今回のようなトラブルは初めて経験したのだけれども、やはりサーバに使うNICは、そこそこ定評のあるものを使うべきだなと痛感したのだった。NICを交換して、/etc/modulesの設定を変更したら、問題なくNICを認識してくれた。その後は問題なく動いている。

## いまさらながらqmail

ところで、このメールサーバのMTA（Mail Transfer Agent）には、定番のsendmailではなくて、qmailを使っている。といっても、qmailの熱烈な支持者というわけではなく、そこの会社では、たまたまqmailを使っていた

ので、データの移行を考えると、qmailを継続したほうが作業が楽だと思っただけだ。それに、sendmailからqmailに移行するノウハウについては、いろいろ情報があるけれども、その逆は情報がなかったこともある。

そこの会社のメールサーバを初めて見たときは、/etc/aliasesにほとんど何も設定されていなかったので、驚いてしまった。メールアドレスに使う名前は、ユーザIDとは別のものが使われていたので、いったいどこで設定をしているのか、見当がつかなかったのだ。

各ユーザのホームディレクトリを調べたら、Maildirというディレクトリに、new、cur、tmpという3つのディレクトリがあって、newには新着メールが1つずつファイルとして作成されていた。なるほど、mbox方式ではなくて、Maildir方式というのは、こういうことだったのかと実感したのである。

以前sendmailを立ち上げているサーバで、/var/spool/mailにあるメッセージファイルの一部に問題が生じて、メッセージをうまく読み取れないことがあった。しかし、qmailのアプローチだと、個々のメールメッセージごとにファイルを作成するので、こうしたトラブルはかなりの程度防げるだろう。

qmailの場合、/var/qmail/alias/ディレクトリに.qmailで始まるファイルを置けば、そのファイルの中で指定したアドレスにメールを転送してくれる。また、ユーザのホームディレクトリに.qmailで始まるファイルを置くことによって、各ユーザーが独自にメーリングリストを設定できるのも便利な機能のひとつだろう。

とはいえ、sendmailでは/etc/aliasesをちゃんと管理していればいいので、アドレスの漏れや残骸を発見しやすいのに対して、qmailではあちこちに.qmailファイルが多数拡散している状況にもなりえるわけで、そうなると、システム全体でエイリアスの参照関係がどうなっているのか、把握しにくくなるともいえる。

そこで、とりあえず/var/qmail/alias/にある.qmailファイルについては、その内容とホストに登録されているユーザーの相互参照ができるPerlスクリプトを作成してみた。ユーザーの登録と削除が頻繁に発生する場合、社内用のメーリングリストに漏れや残骸が発生しやすいので、こうしたツールを自作してみるだけで、ずいぶんと運用が楽になるだろう。いずれにせよ、いまではqmailをとても気に入っている。

*Profile*

**とよふく つよし**

1962年東京生まれ。メディアデザイン研究所技術顧問。訳書に『Javaプログラムクイックリファレンス』『Java分散コンピューティング』（オライリージャパン）『GIMPパーフェクトガイド』（エムディーエヌコーポレーション）などがある。

# オープンソースとセキュリティ

文：安田幸弘
*Text : Yukihiro Yasuda*

## アカウント・コレクター？

先日、何人かのSEとインターネットのセキュリティに関する話をしていたとき、その場にいたある人から、「ところでインターネットで他人のコンピュータを攻撃して、何が面白いんでしょう?」と聞かれたことがある。そう言われてみれば、確かにそうだ。

そのときは、「まあ、アカウントのコレクターか、踏み台にするためのアカウント集めみたいなもんじゃないかなあ」と答えたのだが、管理者のアカウントならともかく、Windowsをクラッシュさせてみたり、他人のファイルにアクセスしても、たいして面白いことはなさそうだ。そんな攻撃を面白いと思うかどうかのセンスが、プロのSEとオタクの違いなのかもしれない。

実際、最近はログを眺めていると、怪しげなアクセスが増えたように思う。使っていないtelnetやFTPポートにアクセスしてくるやつ、あやしげなICMPパケットを投げてくるやつ、無差別にポートスキャンをかけてくるやつなんてのは日常茶飯事だし、そのへんで配布されている攻撃用キットあたりを使っているとおぼしき露骨な攻撃も珍しくない。近頃のように常時接続の利用者が増えてくると、古い攻撃キットでもそれなりに成果があるのかもしれない。

ためしにポートスキャンを検出したときに、スキャンをかけてきたサイトにfingerをかけるようにしてみたら、いまどきfingerdなんて動かしているサイトが結構あって、誰もログインしていなかったりする。

おそらく、踏み台にされているサイトなんだろう。やれやれ。

以前、クラックされたサイトのアカウントの「コレクション」がクラッカーの間で交換されているというような話を聞いたことがあるけれど、ありそうな話だ。

## オープンソースは安全なのか

オープンソースはセキュリティに強いか弱いかという議論がある。だが、この議論はあまり意味がないんじゃないかという気がする。

理屈では、チェックする「目」の数が多ければ、それだけセキュリティホールの検出の機会も増えるはず。それはその通りだけど、果たして星の数ほどあるオープンソースソフトのセキュリティホールをいちいちチェックする「目」の数は、そんなに多いのだろうか。逆に、報告が続出するIISのセキュリティホールにしても、ソースが公開されていないからセキュリティに問題があるというわけでもないだろう。あるいは、見つけやすいオープンソースの穴なんかよりも、IISの穴を見つけるほうが話題性が高い、あるいはコレクションとしての交換価値が高いのかもしれない。そうだとすれば、難易度の高い非公開ソフトの穴を探すクラッカーたちの「目」の数が多くなり、それだけIISに見つかる「穴」が増えたと考えたほうが正しいように思う。

もっともセキュリティってやつは、セキュリティホールが「ある」か「ない」かというだけでもない。超有名サイトはともかく、個人や中小企業のレベルで常時接続しているような零細サイトにとって切実な問題は、セキュリティホールが公開されたときにどれだけ迅速に対応できるかどうかだ。最近はメーカーもそこそこ対応が速くなったとは言うものの、日本語版のセキュリティパッチのリリースを何カ月も待たされるということは珍しくなかった。これは相当いらいらさせられる。穴のサイズが大きいか小さいかはともかく、明らかなセキュリティホールがあることがわかっていながらサーバの運用を続けなければいけないというのは、単に精神衛生だけの問題じゃない。

その点、オープンソースは対応が速い。セキュリティ面でオープンソースが安心感を与える理由

のかなりの部分が、このような対応の速さではないだろうか。もしディストリビュータや開発者からのアップデートが遅れても、いざとなればパッチを当てて再コンパイルするという手もある。オープンソースもクローズドソースも、どちらもソフトウェアである限り無数のバグや欠陥を抱えて動いている。どちらがよりバグが少ないか、欠陥が少ないかという議論やら、ソースコードを見る「目」の数なんかより、問題が表面化したときに、どれだけ迅速に問題解決の手段が提供されるかどうかを論じたほうが、よっぽど生産的だと思うのだが。

## セキュリティ意識を共有する

ある会合で、「要するにオープンソースは安全なのか?」という話題になった。そのときは、「十把ひとからげにオープンソースと言ってもピンキリだ」、というような結論だったように記憶しているが、このとき、ぼくは何となくその結論に違和感を持った。いや、ピンキリだという結論自体は極めてあたりまえの話なのだけど、オープンソースとクローズドソースいう配布形態の持つ意味は、それが優れているのか劣っているのか、安全なのか危険なのかという話だったんだろうか。

セキュリティについて言えば、どんなコードが危険なコードで、どのように書けば安全なコードになるのかというセキュアなコードに関する知識をソースコードを通じて共有することが、オープンソースの最大の意義なのではないだろうか。おそらく、オープンソースのコミュニティの中で共有されたセキュアなコードに関するノウハウは、バイナリ配布の商用ソフトにも取り入れられているだろうし、そのことがソフトウェア全体のセキュリティを高めてきたのではないだろうか。

インターネットで無制限に配布されている攻撃用ソフトにしても、もしあれにソースコードがついていなければ、単なる悪意のソフトでしかなかっただろう。ソースコードが公開される攻撃ソフトは、目に見える形でクラッキングの手口を公開するものと言える。公開された手口を使った攻撃は、効果が激減するのは明らかだ。むしろシステムのセキュリティに責任を負う立場としては、こうした攻撃ソフトもサイトの保安のために大いに有益な情報になり得る。

「サイバーテロ」とやらを口実として、一部でこの種の攻撃ソフトの公開を非合法化すべきだという意見もあるらしい。だが、もしそんなことをすれば、単に危険な攻撃ソフトが一般のシステム管理者の手に届かない地下に潜るだけではないのだろうか。……もっとも、本当はぼくが知らないだけで、地下でもっと強力な攻撃ソフトが公開されているのかもしれないが、いずれにしても非合法の攻撃ソフトなんてものがあれば、これはかえって危険だと思う。もちろんTFNやTrinooのように、単に悪意の攻撃のためのエゲツないソフトもあるものの、原則として攻撃ソフトの非合法化は、一般の管理者がそのソフトを使って自分が管理するサイトをチェックするチャンスと防御の有益な手がかりを奪い、逆に悪意の攻撃者に攻撃のチャンスを与えるものでしかない。

そもそも、もしTFNやTrinooのような攻撃を防ぐことが困難だとすれば、それはインターネットなりTCP/IPなりがその程度のものでしかないということじゃないのだろうか。もし攻撃ソフトの所持を禁止して、その程度のインフラを延命させたところで根本的な解決にはならず、単に犯罪者を量産するだけだ。そんなくだらない法律に労力を割くより、世界の政府はIT社会のために、安全なインフラを守るための知識の共有にもっと力を入れてほしいものである。

*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*

**5・25　AMDとTransmeta次世代バスなどで提携**
**5・29　Intel Itanium正式発表**
**6・1　米国PSINet、会社再建手続きに入る**
**6・4　シャープLinux採用のPDAのプロトタイプを公開**
**6・8　Office XP 国内出荷開始**

今月は、米国に2回も取材に行かねばならないので、結構ハード。これが終わったら、今度は家族を旅行に連れて行かねばならない。なんか休まる暇がないって感じ。個人的には、温泉ってのがいいんだけど、家族としては、やはり海がいいのだとか。沖縄ツアーのパンフを見せられ、夏休み前だと安いといわれるとココロも動くというもの。でもまた飛行機乗るの？

### Office XPは登場したけれど

Microsoftは、Officeの新バージョンであるOffice XPの出荷を開始した。Officeシリーズは、これまで、Microsoftの新しいOSが出る前にバージョンアップが行われるのが恒例。ひとつには、新しいOSで「より便利に」利用できるようにして、新OSへの移行を促すためでもある。というのも、ユーザーからすると、いまや、Microsoftの看板商品は、OSではなく、Officeシリーズなのである。

さて、筆者は原稿書きなどにWordを利用しているが、実は、前回のOffice2000のアップデートにはかなりがっかりした。というのも、結局、かえって使いにくくなってしまったうえ、新しい機能の恩恵をなにひとつ受けられなかったからである。というわけで、今回のバージョンアップについては、かなり否定的である。もっとも仕事柄、動かしてみないとまずいのだが、これについては、Office XP関連の仕事が来たらやろうかと思っている次第。さすがに価値を見い出せないものに身銭を切るのはつらいものがある。

という話は、巷でもささやかれていて、今回のバージョンアップをすべきかどうか、しないとなにが不利益になるのか、といった話題で持ちきりである。

また、今回よりOfficeシリーズは、アクティベーションと呼ばれる操作が必要になる。これは、不正コピーを防止するためのもので、インストール後、インターネットに接続するか、Microsoftに電話して、認証を受ける必要がある。たしかに気軽な不正コピー（カジュアルコピー）は違法であるが、そのために正しく利用しているユーザーに面倒な手間を要求するのもいかがなものか。これに加えて、最近くすぶっているMicrosoftへの不信感というのが、このアクティベーションに対して不安を抱かせる理由にもなっているという。かつて、Wordのドキュメントにイーサネットカードの MACアドレスが埋め込まれて、作成者が追跡可能になっていたこともあったし、Windowsに埋め込まれていた暗号化キーのひとつが米国政府用となっていて、政府機関による解読が可能だったと騒がれたこともあった。

何気なしに使うのなら、問題はないように見えるが、よく考えるとちょっと怖い気もするソフトになってしまったような感じである。

### 米国プロバイダ業界も一波乱

さて、米国のプロバイダは、ユーザー数1位のAOLが2600万人と2位以下を大きく引き離していて、目下行われているのは2位争いである。これまで、マイクロソフトのMSNが約500万人で2位、3位には480万人のEarthLinkが付けていた。ここにきて、無料プロバイダのNetZeroとJunoが合併、新たにUnited Onlineとしてスタートすることになった。しかもユーザー数は約700万人で、一気に2位に躍り出た。

つまり、MSNは3位に転落してしまったのである。マイクロソフトのオンライン事業については、いろいろと言われて

いるが、コンテンツ系は結局どれもうまくいってない。かろうじてプロバイダとしてMSNがなんとか残っており、とりあえず業界2位まで登ってきたわけだが、それが抜かれてしまったわけである。

MSNは、ユーザーの囲い込みという点では、重要な役割を持つ。なので、MSNは、AOLを目標にユーザーを増やし続けなければならない。しかし、ここで3位に転落してしまったことで、United Onlineが当面の目標になったわけである。

先頃、AOLが値上げを発表し、業界はこれに追従するのでは、という観測が流れていたが、こうなると、値上げどころの話ではなくなってしまう。というのも、United Onlineは、無料接続のメニューもあれば、無制限利用のメニューでも、かなり低価格なプロバイダだからである。これを武器に、ユーザー拡大に走るというのが、考えられるシナリオ。だとすると、MSNなどは値上げが難しくなり、逆に値下げも検討しなければならない。3位のEarthLinkは、時間無制限のメニューが月19.95ドルとMSNよりも安く、旧Junoが14.95ドル、旧Net Zeroは9.95ドルだった。

米国でも景気後退という局面にあり、ユーザーは価格に敏感になっているとすると、普通に考えるとここは値下げしかないだろう。AOLが値上げしたので、値下げの効果は大きいと思われる。

となると、米国のプロバイダ市場は、一気に値下げ方向に動く可能性があり、下位のプロバイダは苦しくなって、身売りという話にもなってくるだろう。というわけで、今年は、米国のプロバイダ業界も一波乱あるのではという話。ちなみにAOLタイムワーナーでは、全社員がAOLのメールを使うように強制されたのだとか。そういえば、AOLのメールを使った「You've Got Mail」という映画があったが、これってワーナーブラザースの映画だった。こんなところに合併の縁があったのかも。

## 64ビットOSはどうなる

インテルは、5月29日にようやくItaniumの正式出荷を発表した。これってずいぶん前から話は出ているし、イベントなどでも試作機が大量に並んでいたりして、新しい製品ではないようなイメージがあるが、これでようやく正式な製品として出荷できるようになった。もっとも、多くの企業は、試作機のまま、数千システムぐらい客に納品しているのだとか。

LinuxはすでにIntelのItanium用で64ビット版が登場しているが、MicrosoftもWindows XPの64ビット版を完成させようとしている。以前のイベントで見たら、違いは起動時のロゴに64ビットVersionって出るだけで、あとは普通のWindows XPだったけど、製品版もそうなんでしょうね。

ところで、インテルが製品の発表から出荷を、ここまで引っ張ってきた理由は明かになっていないが、ひとつには、64ビット版Windows XPを待っていたというウワサも。以前に取材したメーカーの人は、「MicrosoftのOSが出荷されるまで、製品出荷を待つ」なんてことを言っていた。これが、Microsoftからの圧力のせいなのか、MicrosoftのOSがないと売れないという判断なのかは不明だが、インテルが正式出荷を発表した前後にベータテストが開始されるというあたり、なにか奇妙な符合を感じる。

これに対抗するかのように、AMDはCrusoeの開発元TransmetaにX86-64とHyperTransportの技術を提供するのだとか。x86-64は、現在のx86アーキテクチャを64ビット化するもの。インテルが、8086を32ビット化したときに使った手法（命令の前にプリフィックスを置くことで命令を64ビット化する方法）で64ビット化を行うという手法だ。たしかにソフトウェアで命令を変換するCrusoeチップには向いているのかも。それにすでにVLIWチップになっているCrusoeでIA-64（これもVLIWだ）をエミュレートするのは難しそうだし。

いまだ登場していないAMDの64ビットCPU（Hammerシリーズ）だが、さて、どんなOSが動くようになるのやら。Microsoftは対応するのでしょうかね。このあたりが、ポイントという気がしないでもない。まあ、交渉ぐらいはしているとは思いますが……。

それで、x86-64をTransmetaが採用すると数のうえでは2社。安価な64ビットマシンが欲しいユーザーには、ありえる選択なのかも。もっとも、データベースとかの64ビットアドレッシングを必要とするアプリケーションが、手に入るかどうかって問題もあるのですが、たとえば、オープンソースのデータベースなんかを対応させちゃうってことも不可能ではないわけで、ひょっとすると、意外に健闘することになるのかも。


米国PSINetは会社再建手続きに入ったが、傘下のグループ企業には影響なしと発表された（http://www.jp.psi.net/）。

正しきタコへの道

# 初級Linuxer養成講座

## シェルスクリプト入門〜繰り返し処理（2）

**forを使った繰り返し処理は、大量のファイルなどを操作しなければならない場合に便利である。今回は、これを応用して、ファイルに「番号」を付ける方法を説明しよう。例として「MP3ファイルのファイル名変更」を取り上げているが、どんな場面でも応用が利くはずだ。**

文：竹田善太郎
*Text : Zentaro Takeda*

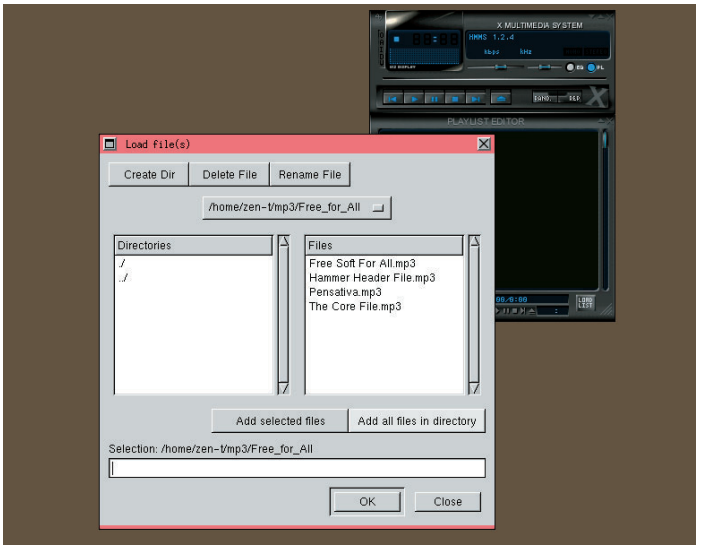
この連載中では、覚えておくと便利な操作方法を説明するたびに、**付箋紙にでも書き留めて、ディスプレイの脇に貼っておくとよい**と書いてきた。Linuxに限らず、コンピュータの使い方をマスターするのに、これほど効率のよい方法はないからだ。

いろいろなパソコンの解説書を読むと、その内容の98%くらいは、1回限りで読み飛ばしてしまってかまわないものである。しかし、**残りの2%**の部分に、**覚えておくと便利**というよりも、**知らないと困る**ような内容が含まれている。本来なら、そのような内容だけ**別冊の付録**にして、いつでも参照できるようにするのが親切というものなのだが、そのような配慮のなされている本は少ない。別冊にするとコストが倍増するからだ。

そこで、原始的な方法のようだが、便利そうな使い方に行き当たるごとに、ちょっと面倒でも付箋紙に書いて、目に付くところに貼っておくことをお勧めするのだ。人間の記憶というものは本当にいい加減なので、細かなコマンド操作などを一字一句間違えずに覚えておくことなど、**まず不可能**である。毎日、日常的に使う**lsコマンド**や**cpコマンド**であっても、すべてのコマンドラインオプションを丸暗記するのは大変だし、そんなことを覚えておくくらいなら、もっと生産的なことに精力を傾けるべきだろう。

めったに使わないコマンドなら、使うたびに**manコマンド**や**--helpオプション**で調べるのも結構だが、**どんなコマンド名だったかも思い出せない**と、これを調べるのは相当の手間がかかってしまう。このような場合、紙に書いたものが貼ってあれば、すぐに思い出すことができるので、無理して**暗記**する必要もない。もちろん、手帳などに書き留めておいたり、コンピュータ上のファイルやPDAなどにメモしておいてもかまわないのだが、ふだん同じ机の上で作業しているのであれば、ディスプレイに貼り付けたメモほど目に付きやすいものはないだろう。GUIのデスクトップ画面に付箋紙のようにメモを表示するソフトも数多くあり、いろいろと使ってみたことがあるのだが、どうもなじみが悪くて、使い続けているものはない。

「コマンドの使い方がべたべたと貼ってあると**はずかしい**ので、**いち**




**画面1**
XMMSでディレクトリ中のファイルを読み込むと……。再生順序を指定するリスト（プレイリスト）を作成しないと、「名前順」に読み込まれるようになっている。

早く覚えようと努力するという効果もある」とは筆者の知人の弁だが、さもありなんだ。もちろん、ログイン用のパスワードや企業秘密のデータなどを貼り付けておくのはもってのほかだろうが、ちょっと覚えておくと便利な情報を一時保存するのに、これほど便利な方法はないだろう。

問題は、最近のディスプレイ装置の中には、デザイン上の都合なのか、画面周囲の「枠」の部分が狭くなっているものが多くなったことだ。あるいは、これもデザイン上の都合なのだろうが、表面が波型に整形されていたりして、付箋紙が貼り付けにくい形になっているものもある。町なかの電柱などで、宣伝ビラの貼り付け防止用にデコボコの加工をしているものがあるのを思い出す。まさかディスプレイ装置にそのような目的の加工をするはずもないだろうが、メモ用紙を貼り付けるような使い方は想定していないということなのだろうか。ともかく窮屈な話ではある。

## ファイルに「番号」を振る

連載のタイトルで「初級Linuxer〜」と銘打っておきながら、bashのforコマンドなどという、ちょっと高度な使い方を説明しているのは、筆者自身がこのコマンドをとても便利であると感じているからだ。特に、音楽CDからMP3ファイルを作成する場合には、なくてはならない機能だと思う。もちろん、このような用途に限定される話ではないが、Linuxの使い道の一例として見てもらいたい。

Linux上でバッチ的にMP3ファイルを作る場合、グラフィカルなインターフェイスを備えたツールを使うのもよいのだが、コマンドラインだけで手早く済ませたい場合などは、次のような操作をよく行っている（ちなみに、「cdparanoia」は音楽CDからWAV形式のファイルを作成するツール、「bladeenc」はWAV形式ファイルをMP3形式に変換するためのツールである）。

```
$ for track in 01 02 03 04 05
> do
> cdparanoia -q $track track$track.wav
> bladeenc -quiet -delete -quit
track$track.wav
> done
```

しかし、それ以上に便利、というより欠かせないのは、市販のMP3作成ソフトなどで作ったMP3ファイルを処理する場合だ。ソフトの設定を適切にしていれば問題はないのだが、うっかりすると、曲名をファイル名にしたMP3ファイルや、トラック番号の桁数がそろっていないMP3ファイルを作ってしまうことがある（**図1**）。このようなファイル名のなにが問題なのかというと、各種のMP3再生ソフトやポータブルプレーヤでこれらのファイルを再生しようとすると、曲順がめちゃくちゃになってしまうのだ（**画面1**）。

曲名がファイル名になってしまった場合は、とくにやっかいである。元の

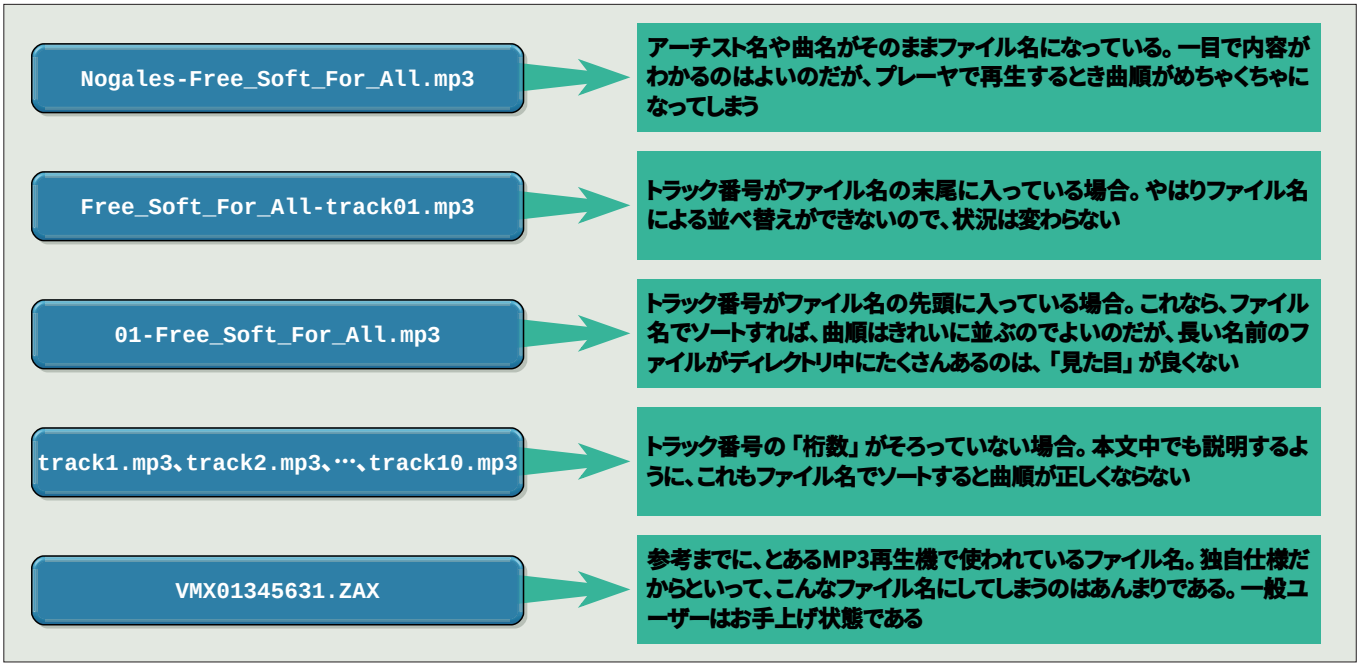



図1　困ったファイル名の例
このような名前のファイルだと、トラックの名前順（辞書順）でしか再生できないMP3プレーヤでは困ったことになってしまう

曲順はファイル名からはいっさい判別できないからだ。このような場合、CDのジャケットの曲名リストを見ながら、ファイル名をひとつずつ変更したり、あるいは再生順序を指定するプレイリストファイルを作成することになるのだが、なかなか面倒な作業である。

MP3ファイルの作成アプリケーションは、通常トラックの先頭から末尾に向けて、順番にファイルを作成するので、ファイルの更新日時を元にすれば、トラックの順番もわかる。lsコマンドには、ファイルの更新日時で並べ替えをするオプション（-t）があるので、逆順表示のオプション（-r）と組み合わせて、

```
$ ls -rt ./*.mp3 > playlist.m3u
```

などとすれば、XMMUなどで使えるプレイリストを作ることは可能である。しかし、筆者の所有しているMP3ポータブルプレーヤでは、プレイリストを使った再生には対応しておらず、**ファイル名でソートした順序**でしか再生してくれない。

そこでなんとかしてファイル名をトラック番号で表現したものに書き換えてやりたい。つまり、**ファイルに番号を振り直す**ような作業を行いたいわけだ。このようなときに、forコマンドが役に立つのだ。

ファイルに番号を振る場合、その番号をなんらかの方法で計算させなければならない。まずは、その方法について触れておこう。

| 演算子 | 意味 |
|---|---|
| ＋ | 加算 |
| － | 減算 |
| ＊ | 乗算 |
| ／ | 除算（結果は整数に丸められる） |
| % | 剰余 |

表1
bashの$((……))で利用できる演算子の例（実際にはこのほかにもビット演算や論理演算などの演算子が使える）。

### ●数値計算をさせる方法

前々回、bashのシェル変数には文字列や数値など**なんでも入れることができる**と書いたかと思う。数値を入れることができるのなら、**変数を使った数値計算**もできるのではと考えるのが自然だろう。実際、bashではコマンドラインで数値計算をさせて、その結果を変数に保存したり画面に表示させることができるのだ。

たとえば次のようなコマンドラインを実行させてみる。

```
$ echo 1+1
```

結果は明らかに失敗で、このコマンドラインはオウム返しに、

```
1+1
```

という**文字列**を表示するだけである。では、次のコマンドラインを実行してみるとどうだろう。

```
$ echo $((1+1))
```

すると、想定したとおりの答えが表示される。

```
2
```

この$((……))という記述方法は、bashに対して**かっこ内を数値計算してその答えを返せ**という指示を与えるものなのだ。これは、とても便利な機能なので、**付箋紙にメモしてディスプレイの脇に貼っておく**ことをお勧めする。Linuxでは、コマンドラインで計算を行うbcというコマンドもあるのだが、bashだけでも四則計算くらいはできることを覚えておいて、損はないだろう。$((……))で利用できる計算（演算子）の一覧を**表1**にまとめておくので、参考にしてほしい。

$((……))の中では、変数の値を呼び出して計算することができる。また、結果を変数に保存することもできる。

```
$ a=10; b=20
$ result=$(($a * $b))
$ echo $result
200
```

この機能とforコマンドを使えば、次のようなことが可能になる。


画面2
最近のMP3作成ツールなら、曲目のデータをファイル中に埋め込めるので、再生時に曲名はわかるようになっている。ファイル名を変えたとしても、困るようなことはないだろう。

```
$ count=0
$ for f in ./*
> do
> count=$(($count+1))
> done
$ echo $count
```

上のコマンドラインを実行すると、カレントディレクトリ中のファイルの数が表示される。もちろん、こんなまわりくどいことをしなくても、「ls | wc -l」を実行すれば一発じゃないかという意見もあるだろうが、これはあくまでシェル変数とforコマンドの使い方を示すための例である。

さて、上の例に手を加えれば、カレントディレクトリ中のmp3ファイルに、変更の日時順の番号を振ることができる。番号は、1から数え始めることにしよう。

```
$ count=1
$ for f in `ls -rt ./*.mp3`
> do
> mv $f file$count.mp3
> count=$(($count+1))
> done
```

ここで、`ls -rt ./*.mp3`となっている部分では、lsコマンドで.mp3という拡張子を持つファイルの一覧を、更新日時の逆順で出力させている。以前にも触れたかと思うが、`……`という引用符（バッククォート）は、その中身をコマンドラインとして実行して、その結果の文字列をその場所に記述したものとみなせ、という意味を持つ。これについては改めて解説したいと思うが、ここでは決まり文句のようなものと考えてもらいたい。

ともかく、上のforコマンドを使えば、曲名がファイル名になっていたmp3ファイルを、

```
track1.mp3、track2.mp3、……
```

というようなファイル名に変換することができる。ファイルの数が9個以内なら、これで当面の目的は達成できる（**画面2**）。

### ●番号の桁数をそろえる

番号を付けるファイルの数が10個を超えると、先ほどの方法では不十分である。次のような11個のファイルがあったとしよう。

```
track1.mp3
track2.mp3
    ⋮
    ⋮
track10.mp3
track11.mp2
```

これらのファイルを、ファイル名の辞書順にソートすると、次のような結果になってしまう。

```
track1.mp3
track10.mp3
track11.mp3
track2.mp3
    ⋮
    ⋮
```

track2よりも、track10やtrack11のほうが前にきてしまうのである。これでは、ファイル名の順に再生するMP3プレーヤでは都合が悪い（**画面3**）。

このような場合には、ファイル名に含まれる数字の桁数をそろえて、その桁数に満たない番号については、先頭に0を付けるのが一般的だ。つまり、次のようなファイル名を付けるわけだ。

```
track01.mp3
track02.mp3
    ⋮
    ⋮
track10.mp3
track11.mp3
```

これなら、ファイル名を名前順にソートしても、数字の順にきれいに並んでくれる。

このようなファイル名を付ける場合


**画面3**
桁数がそろっていない数字付きファイルを再生。10番目と11番目のトラックが2番目のトラックより先に再生されてしまう（ただし、最近のMP3再生ソフトでは、この点が改善されているものもあるようだ）。

にも、forコマンドを利用すればよいのだが、実際にはちょっと複雑なコマンドになってしまう。sedコマンドなどをうまく組み合わせて、数字の部分の桁数が同じになるようにしなければならない。これは、かなり面倒である。

そこで、筆者はこのような場合に、もう少し簡単な方法を使っている。**ファイルの番号を100から始める**のである。つまり、

```
track101.mp3
track102.mp3
    ⋮
    ⋮
track110.mp3
track111.mp3
```

というようなファイル名にすればよい。**ちょっとみっともない**かもしれないが、これでファイル名の順にソートしても、目的通りに並んでくれるし、MP3プレーヤなどでの再生も問題なくできるようになる。

具体的には、次のようにforコマンドを使う。

```
$ count=100
$ for f in `ls -lr *.mp3`
> do
> mv $f track$count.mp3
> count=$(($count+1))
> done
```

ファイルの数が100個以上ある場合、すなわち、ファイル番号の桁数が3桁になるような場合には、最初のcount変数の値を1000にしておけばよい。

## スペースが含まれたファイル名

さて、どんどん話が脱線するようだが、ついでなので、もっとやっかいなファイル名への対処方法を説明しておこう。筆者が利用しているWindows用のMP3変換ソフトの中には、曲のアーチスト名や曲名をファイル名にしてしまうだけでなく、**スペース文字を含んだファイル名**を作成してしまうものがある。ファイル名にスペースが含まれていても、別に**不法なファイル名**というわけではなく、WindowsマシンでもLinuxでも、ファイルへのアクセスはできるし、それぞれのOSの**仕様に背いているわけではない**。

しかし、コマンドラインでファイルを扱うとき、スペース文字がファイル名に含まれていると、**かなり面倒**なのである。たとえば、

```
This is my favor.txt
```

というようなファイルがあったとする。このファイルを、次のようなコマンドラインでコピーしようとしてみよう。

```
$ cp This is my favor.txt
my_favor.txt
```

すると、

```
cp: copying multiple files, but
last argument……
```

というような、長々としたエラーメッセージが出てしまう。ファイル名の中に含まれている空白文字と、コマンドの引数を区切る空白文字が**ごっちゃ**になってしまうからだ。

このような場合、bashなら、

```
$ cp "This is my favor.txt"
my_favor.txt
```

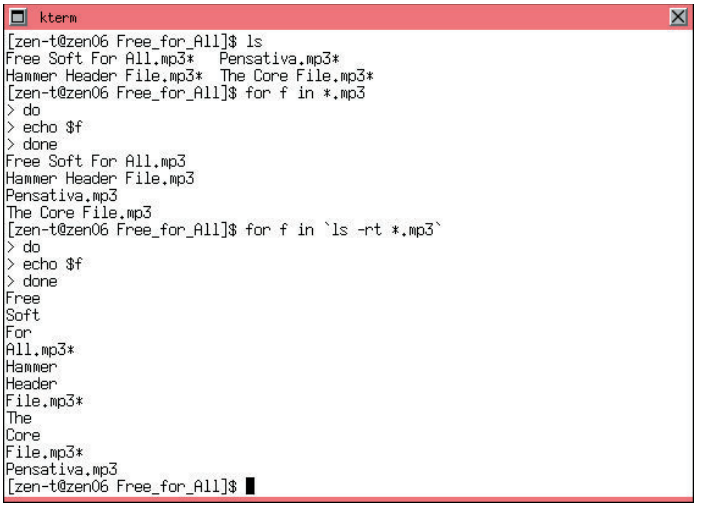
のように、"（**二重引用符**）でファイル名を囲んでやれば問題は解決する。あるいは、

```
$ cp This¥ is¥ my¥ favor.txt
my_favor.txt
```

のように、ファイル名に含まれるスペース文字の前に¥を付けてやっても同じである。ところが、forコマンドなどの中で、このようなファイルを複数個処理しようとするとやっかいである。

たとえば、次のような3つのMP3ファイルがあったとする。

画面4
forコマンドとlsコマンドでファイル名に空白を含むファイルを処理させてみる。forコマンド単体では問題ないのだが、lsコマンドでソートさせた結果を処理させてみると、問題が起こる。



```
Free Software for All.mp3
Magnetic Head.mp3
The Core File.mp3
```

これらを、先ほどと同じようにforコマンドを使って変更日時順に処理しようとすると、おかしなことになってしまう。

```
$ count=1
$ for f in `ls -rt *.mp3`
> do
> mv $f track_$count.mp3
> count=$(($count+1))
> done
mv: cannot stat `Free': No such file or directory
mv: cannot stat `Software': No such file or directory
    ⋮
    ⋮
```

lsにファイルをソートさせてforコマンドに与えると、どうも、それぞれの**単語**をファイル名と解釈してしまうようなのだ（**画面4**）。「for f in *.mp3」のように、直接ファイル名をワイルドカードで指定した場合は問題ないのでちょっと不思議だが、ここでは**そのようなもの**と納得しておいていただきたい。

この問題の解決方法にはいろいろあるだろう。もっと**スマートな解法**はいくらでもありそうだが、筆者がよく使っているのは次の方法だ。

まず、順番は気にせずに、元のファイル名を当たり障りのない番号付きファイルに一度変えてしまう（mvの行で、$fの周りを"……"で囲んでいる点に注意すること）。

```
$ count=1
$ for f in *.mp3
> do
> mv "$f" number$count.mp3
> count=$(($count+1))
> done
```

すると、ディレクトリ中のMP3ファイルはすべて

```
number1.mp3
number2.mp3
number3.mp3
    ⋮
    ⋮
```

というファイルに名前が変わってしまうが、ファイル名を変更しただけなので、**ファイルの更新日時の情報は元のまま**変わっていない。ここであらためて、先に説明した方法で、**更新日時順に番号を付け直す**のである。

```
$ count=1
$ for f in `ls -rt *.mp3`
> do
> mv $f track_$count.mp3
> count=$(($count+1))
> done
```

これでめでたく、

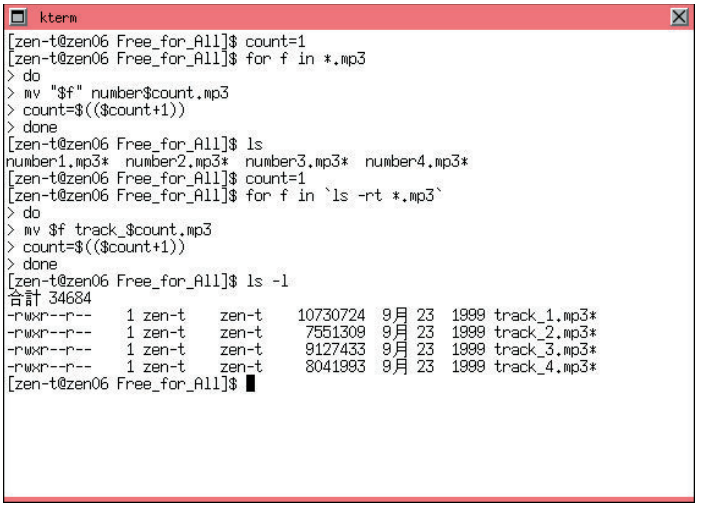
```
track_1.mp3
track_2.mp3
track_3.mp3
    ⋮
    ⋮
```

というように、ファイルの更新日時順に番号の付いたファイル名に変えることができる（**画面5**）。

このように、文章で説明してしまうとずいぶん回りくどく感じるかもしれないが、ファイル名をひとつひとつ付け直したり、MP3ファイルの作成からやり直したりするのに比べれば、ずっと楽な方法である。このようなforコマンドの使い方はいろいろと応用が利くと思うので、ひるまずに練習してみてほしい。

さて、今回はforコマンドの使い方の「応用編」といった内容になってしまったが、"……"や`……`といった**引用符**の使い分けがちょっと目についたかもしれない。シェルの解説書を読むと、この引用符の使い方が最初に説明されていることが多いのだが、初心者にはなかなか理解するのが難しいテーマでもあるので、いままではあえて詳しくは触れてこなかった。次回は、この引用符の使い方を見てみることにしよう。

**画面5**
ファイル名だけの変更なら更新日時は変わらないので、このようにすれば「変」な名前のファイルでも、日時順のファイル名に付け替えることができる。



# Oracle8iで作るWebサイト入門

**Webページにデータベースの内容を表示するためには、DBMSに接続して得られた結果を元にして動的なHTMLを生成する必要があります。本連載では、そのためにPHPというサーバサイドスクリプトを使用します。**

## 第2回　PHPを使う（前編）

文：おもてじゅんいち／かざぐるま
*Text : Junichi Omote/Kazaguruma*

企業システムにたずさわったことのある方ならご存じでしょうが、システムの多くはメインフレームと呼ばれる汎用コンピュータをホストにして、古くはダム端末というキーボードとディスプレイだけの端末をいくつもぶらさげて、その後はオフコンを端末として構成されていました。

そんな企業システムも、端末として――端末とは言わなくなりました。クライアントですね――PCを使うようになり、ホストもサーバという名のワークステーションが使われるようになりました。これがクライアント／サーバシステムですが、最近はもっと進んで、クライアント側に特別なソフトウェアを入れないで、Webブラウザだけで運用できる、Webアプリケーションの形態に変化しています。ソフトウェアの機能はすべてサーバ側に搭載することで、プログラムの修正やバージョンアップをサーバ側だけで済ませることができるのが大きなメリットです。しかしこれって昔のダム端末に戻ったような気もするのですが…。

とはいえ、Webベースですから、社内システムとしてイントラネット上で使うこともできますし、インターネットを介して遠方との業務や、インターネット上で一般に公開するなど、いろんな運用形態に合わせて利用することができますね。何よりも、規模によっては何百台～何千台もあるクライアントマシンのソフトウェアをメンテナンスする必要がないというところが、Webアプリケーションばやりの理由でしょう。

セキュリティの問題やデータベースの規模などを別にすれば、世の中で実際に使われているWebアプリケーションの原理や開発方法は、この連載で学ぶこととほとんど同じです。ここしばらくはWebアプリケーション全盛の時代になるでしょうから、皆さんもまずはこの記事でWebアプリケーションの入口に立ってみてください。

### データベースWebアプリケーション

業務を行うシステムはもちろん、ちょっとしたWebサイトでもデータベースを使うのは普通でしょう。ではWebアプリケーションでデータベースを使うには具体的にどうすればよいのでしょう。

Webアプリケーションというからには、Webサーバが動きます。クライアント側はWebサーバから送られてきたデータをブラウザに表示し、時にはWebサーバにデータを送り返すというやりとりです。データベースはサーバ側で動いていますから、サーバ側では、必要なデータをデータベースから取得して、それをブラウザで表示できるようにHTMLの形に加工してデータを送ります。

ですから、サーバ側ではデータベースアクセスを担当するデータベースサーバ、データベースサーバからのデータをHTMLデータに加工するプログラム、HTMLデータを送信するWebサーバの3つが必要になります（**図1**）。

本連載では、データベースサーバにOracle8iを、Webサーバに Apacheを使います。ApacheはMiracle Linux

インストール時にセットアップされていますから、すぐに使えます。あとは「HTMLに加工」の部分ですが、この部分のしくみとしてCGIとサーバサイドスクリプトの2つの方法があります。かつてはCGIがよく使われていたのですが、今回はサーバサイドスクリプトを使ってみます。

### ■CGIとサーバサイドスクリプト

CGI（Common Gateway Interface）とは、Webサーバがhtmlファイルの代わりに、比較的小さなプログラムを起動するしくみのことで、このプログラムがデータベースにアクセスしたり、HTMLを加工したりしてくれます。といってもそんなプログラムがはじめから用意されているわけではなく、自分の目的（画面）に合わせたプログラムをそのつど作る必要があります。このプログラムが実行結果として出力するデータをHTML形式にして、そのデータをWebサーバに送信させれば、クライアントのブラウザにはあたかもその場で作られたHTMLファイルが送られてきたように見えるのです。

CGIは独立したプログラムですから、プログラムとして動作可能であれば開発言語はなんでもよく、PerlやCなどがよく使われます。ただ、プログラムの動きと関係のない、固定したHTMLタグなども、

```
print("<BR>");
```

というようにプログラムの一部として書く必要がありますし、いくら小さいといってもそのつどプログラムを起動することになるため、OSの資源を消費することになり、パフォーマンスもあまり良くありません。

一方サーバサイドスクリプトは、CGIのように別プログラムが動くのではなく、Webサーバプロセス内でWebサーバ自身がスクリプトを解釈して実行します。別のプログラムのためのプロセスを生成することもないので、サーバ側の処理が軽くなります。

また、サーバサイドスクリプトは、普通にHTMLを記述しながら、必要な部分にスクリプトを埋め込むという方式なので、スクリプトと関係のない固定されたHTMLタグなどは、HTMLファイルを作成するときと同じように、そのまま記述することができるのです。

HTML部分とスクリプト部分の区別が明確ですから、修正などのメンテナンスも楽になります。

### ■PHPとは

本連載ではサーバサイドスクリプトのひとつであるPHPを使います。CGIに比べて、サーバサイドスクリプトはメリットが多いのですが、Webサーバ自身の機能になるため、WebサーバにPHP用のモジュールを組み込むという作業を行っておかなければなりません。これが結構難しかったりするために、サーバサイドスクリプトが敬遠されることがあるのですが、ありがたいことに、Miracle LinuxではインストールTIMEに、Webサーバ（Apache）へのPHPモジュール組み込みが行われているのです。すなわち、Miracle Linuxのインストールが完了すれば、すぐにPHPを使うことができるようになっています。

PHPはもともと、PHP/FI（Personal Home Page Tools/Form Interpreter）という名で開発されたもので、その後PHP/FI2.0をもとに、PHP3（Php Hyper Preprocessor）として生まれ変わりました。現在はPHP4がリリースされていますが、PHP3はしっかりした国際化対応が行われてきているため、日本語環境で使いたい場合はPHP3のほうがまだ適しています。PHP4には新しい機能も搭載されましたので、早くPHP4が国際化されて使えるようになるのを望むところです。PHPの国際化対応に興味のある方は、日本PHPユーザ会のWebサイト（http://www.php.gr.jp/）などに、PHP国際化プロジェクトの詳細があります。

### ■PHPの考え方

PHPは前述のようにHTMLの中に埋め込んで使います。ですから作業自体は、普通のHTMLファイルを作成してサーバに置いておくのと同じイメージです。ただし、PHPを記述したファイルは、スクリプトが含まれていることを

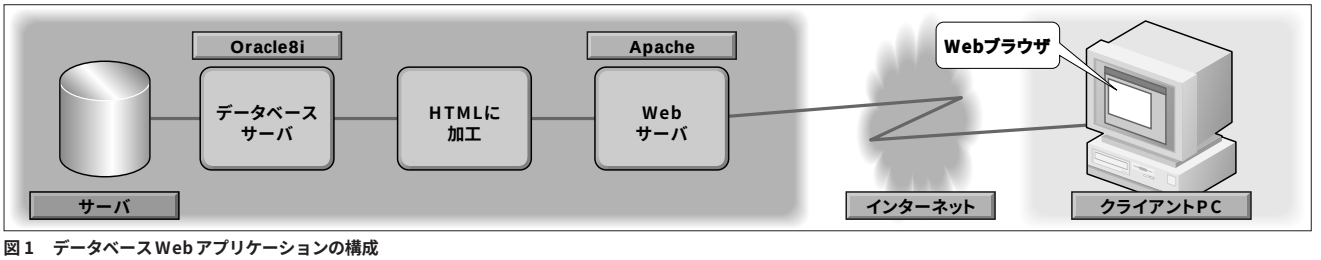



図1　データベースWebアプリケーションの構成

Webサーバに認識させるために、拡張子を.php3にしておきます（PHP4の場合は拡張子は.php）。

**図2**が、PHPスクリプトがHTMLに変換されて、クライアント側のブラウザで表示されるまでのしくみです。あらかじめ決まっていて、状況に合わせて変化させる必要のない部分はそのままHTMLで記述しておきます。データベースへアクセスする部分や、フォームで入力されたデータを処理する部分などをPHPスクリプトで記述しておき、Webサーバ内のPHPインタプリタがスクリプト部分を解釈して、結果をHTMLで出力してくれます。

クライアント側のブラウザが受け取る時点では、スクリプトの部分もその実行結果のHTMLに加工されていますので、ブラウザ側に特別な処理機能がなくても、単にHTMLを表示するだけで済むことになります。

### ■PHPの書き方

**リスト1**がPHPファイルの例です。HTML部となっている部分はHTMLをそのまま記述すればよい部分で、普通のHTMLファイルと何ら変わりはありません。PHPスクリプト部が特殊な部分で、スクリプトであることを明確にするために、<?phpと?>でくくります。この中はPHPの文法に従ってスクリプトを書かなければなりません。<?phpと?>でくくられた中にHTMLタグをダイレクトに書くとエラーになります。

<?phpと?>の代わりに、<?と?>でくくってもよいのですが、将来的にXMLなどとの関連を考えると、<?phpを使うことをお勧めします。

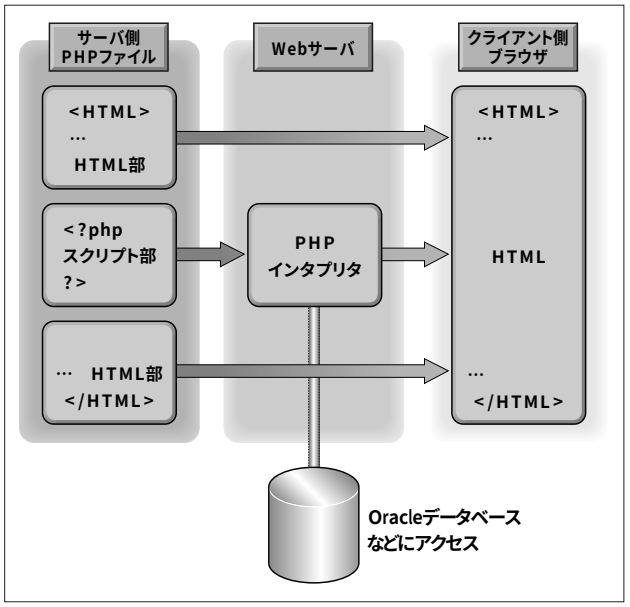



図2　HTMLへのPHP埋め込み

## PHPの文法

PHPの文法は、C言語にPerlを合わせたようなものになっています。ですから、CやPerlを知っている方ならすんなりとマスターできるでしょう。文法の基本はC言語、変数、文字列の扱いや正規表現のサポートなどはPerl、各種組み込み関数はCとPerlとPHP独自の便利な関数群といった雰囲気でしょうか。

### ■変数

PHPの変数は、変数名の先頭に$を付けます。Perlと同じですね。名前にはアルファベットと数字が使え、日本語は使えません。

```
$a、$A、$a1、$var、$kingaku、……
```

大文字、小文字は区別されるため、$aと$Aとは別のものになります。

変数に代入できるデータには、文字列（テキスト）、整数、実数などがありますが、それらを格納する変数の型に、文字列型とか数値型とかの区別はありません。変数の型は、その変数が使用されるときに自動的に決定されます。

たとえば、

```
$var = "50";
```

と、変数$varに"50"という文字列を代入していても、

```
$res = $var + 4;
```

▼リスト1　PHPファイルの例

```
<html>
<head></head>
<body>
ただいまの時刻<br>

<?php
  $ta = getdate();
  $h = $ta["hours"];
  $m = $ta["minutes"];
  $s = $ta["seconds"];
  echo "$h:$m:$s\n";
?>

</body>
</html>
```

HTML部

PHPスクリプト部

HTML部

と、＋を使って数字を加算すれば、"50"は自動的に数値として扱われ、$resの結果は54になります。

つまり式の状況に合わせて、文字列、整数、実数のいずれにでも扱いを変えてくれるのです。

```
$var = "50";    のとき
$res = $var . "円";      →（結果）$res = "50円"
(. は文字列と文字列をつなぐ演算子です。後述)
$res = $var + 4;         →（結果）$res = 54
$res = $var + 1.3;       →（結果）$res = 51.3
```

"50"のように、文字列の内容が数値のみの場合だけでなく、文字列の先頭が数字か－（マイナス）記号であれば、同じように数値として評価されます。たとえば、

```
$var = "80円のりんご";
```

のような場合、数値として評価される式で使用すると、$varは80という数値として扱われます。残りの文字列は無視されます。そして、PHPではC言語などのように変数の型宣言をする必要がありません。

### ■配列

これまでの変数は、$var=50のようにひとつの値（文字列）を記憶するものでした。これに対して、複数の値や文字列をひとまとめにして記憶する変数を配列と呼びます。

配列は通常、$a［0］、$a［1］……というように、変数名の後ろに［ ］をつけて、そこに配列内の順番を表す添字（そえじ）をつけます。添字の番号は0から始まることに注意してください（たとえば、配列の中の３番目の値は、［2］になります）。配列は、

```
$a[0] = "りんご";
$a[1] = "みかん";
$a[2] = "メロン";
```

というように使います。$a［0］などの、添字のついたものは通常の変数と同じように扱えますが、$aというのは配列全体をあらわすため、通常の変数とは違うものになっています。

配列の各要素に値を代入するときには、上記のほかに、

```
$a[] = "りんご";
$a[] = "みかん";
$a[] = "メロン";
```

というように、添字の数字を省略することもできます。この場合は、0から順番にまだ値が代入されていない要素が使われていきます。結果は、添字を指定した場合と同じです（$a［0］、$a［1］、$a［2］に代入される）。

あらかじめたくさんの値を代入しておく場合は、array（）関数というものを使って、

```
$a = array("りんご","みかん","メロン");
```

と書くこともできます。この場合だけは左辺が$a（添字なし）になりますので気をつけましょう。ようするに、配列全体として$aに代入しているというイメージです。

### ■多次元配列

テーブル（表）などを作成する場合、行と列でセルを表すことがよくあります。そういうとき、各セルに1対1で変数が対応していれば便利です。その場合は、行と列という2次元のインデックスをもつ多次元配列が使えます。

2次元といっても、添字が２つ並ぶだけです。

```
$cell[0][0] = 100;
$cell[0][1] = 200;
$cell[0][2] = 300;
$cell[1][0] = 400;
……
```

最初の添字を行番号、次の添字を列番号と見れば、各変数はセルに対応しますね。

多次元配列をarray（）関数で代入するには、array（）関数の中にarray（）関数を並べて、

```
$cell = array(
        array(100,200,300),
        array(400, …… )
);
```

というように行います。内側にあるarray（）のうち、最初が$cell［0］［0］、$cell［0］［1］、$cell［0］［2］……に、次のarray（）が、$cell［1］［0］、$cell［1］［1］、$cell［1］［2］……にあたります。

■文字列

PHPで扱うデータとして文字列があります。文字列は基本的に、

```
"あいうえお"
'ABCDE'
```

のように、""（ダブルクォート）や、''（シングルクォート）でくくります。

ダブルクォートの場合、その中に変数名が含まれていると、その変数の値が使われます。たとえば、

```
$var = 5000;
echo "価格は$var円です。";
```

の結果は、

```
価格は5000円です。
```

となります。逆に、値への展開を避けたい場合はシングルクォートでくくります。

```
$var = 5000;
echo '価格は$var円です。';
```

の結果は、

```
価格は$var円です。
```

となります。文字列の扱いもPerlに準じています。

■エスケープシーケンス

ダブルクォートで文字列をくくっているときに、「"」という文字を使いたい場合、

```
echo "二重引用符は"です。";
```

と書いてしまうと、「二重引用符は」の直後の"が、文字列の終わりと解釈されてしまい、思うような結果になりません。その場合は、単なる文字として扱いたい"の前に\を入れて、

```
echo "二重引用符は¥"です。";
```

とすれば、正しく表示されます。

つまり、変数を表す$やダブルクォート内での"などの意味を打ち消して（エスケープして）やるために、\を使います。その他にも、改行やタブなど、文字として表せないものを扱う場合にも\を使います（**表1**）。

日本語フォントでの\は、欧文フォントでは＼（バックスラッシュ）になります。エディタ等の環境によっては\と表示しないかもしれませんが、この2つの文字は同じものです。

■演算子

PHPの演算子には、CとPerlから受け継いだ、

```
算術演算子
加算子／減算子
代入演算子
比較演算子
論理演算子
文字列演算子
ビット演算子
実行演算子
エラー制御演算子
```

などがあります。よく使われるものを説明します。

■算術演算子

数値の計算をするための演算子です（**表2**）。

■加算子／減算子

CやPerlと同様に、1を加算したり減算したりするために、特別な演算子があります（**表3**）。

■代入演算子

算術演算子と同じ効果ですが、

```
$a = $a + 3;
```

| 記述 | 意味 |
|---|---|
| \n | 改行（LF） |
| \r | 復帰（CR） |
| \t | タブ |
| \\ | \という文字 |
| \$ | $という文字 |
| \" | "という文字 |

表1　エスケープシーケンス

| 式 | 意味 | 説明 |
|---|---|---|
| $a + $b | たし算 | |
| $a - $b | ひき算 | |
| $a * $b | かけ算 | |
| $a / $b | わり算 | |
| $a % $b | 余り | $a を $bで割った余り |

表2　算術演算子

| 式 | 意味 | 説明 |
|---|---|---|
| $a++ | 1を足す | $aの評価のあと1を足す |
| ++$a | 1を足す | 1を足してから評価する |
| $a-- | 1を引く | $aの評価のあと1を引く |
| --$a | 1を引く | 1を引いてから評価する |

表3　加算子／減算子

| 意味 | 式 | 算術演算子 | 説明 |
|---|---|---|---|
| たし算 | $a += 2 | $a = $a + 2と同じ | $a + 2を$aに |
| ひき算 | $a -= 2 | $a = $a - 2と同じ | $a - 2を$aに |
| かけ算 | $a *= 2 | $a = $a * 2と同じ | $a * 2を$aに |
| わり算 | $a /= 2 | $a = $a / 2と同じ | $a / 2を$aに |
| 余り | $a %= 3 | $a = $a % 3と同じ | $a を3で割った余りを$aに |

表4　代入演算子

| 式 | 意味 |
|---|---|
| == | 等しい場合に真 |
| != | 等しくない場合に真 |
| > | より大きい場合に真 |
| < | より小さい場合に真 |
| >= | より大きいか等しい場合に真 |
| <= | より小さいか等しい場合に真 |

表5　比較演算子

| 演算子 | 意味 | 説明 |
|---|---|---|
| ! | 否定 | 偽の場合に真 |
| \|\| | または | どちらかが真の場合に真（or） |
| && | かつ | 両方が真の場合に真（and） |

表6　論理演算子

といったように、計算結果をふたたび計算に用いた変数に戻す場合に使います。「変数$aの値を3増やす」などといった場合にわかりやすい記述です（**表4**）。

### ■比較演算子

あとで出てくるif文などで使われるものですが、PHPでも対象となる式が真か偽かで、条件判断を行います。比較演算子を含む式は、真か偽の値をとります（**表5**）。

また、比較演算子のない式や変数のみが評価される場合は、「0以外はすべて真」という規則にしたがって、数値ならば0のときに偽、それ以外は真になり、文字列ならば空文字列のときに偽、文字が1文字でもあれば真になります。

### ■論理演算子

条件判断のときに2つ以上の比較演算式がある場合は、それらの式を「または」もしくは「かつ」で結ばなければなりません。たとえば、$aが1から9までの数値であるかどうかを判断するには、

```
$a が1以上
$a が9以下
```

という2つの判断を「かつ」でむすんで、「$aが1以上かつ9以下」であるかどうかを評価することになります。式にすると、

```
($a >= 1) && ($a <= 9)
```

のようになります。この場合の&&のような演算子を論理演算子と呼びます（**表6**）。

### ■文字列演算子

$a = "50"、$b = "円"の場合、この2つをつなげて、"50円"という文字列を作ろうとして、

```
$var = $a + $b;
```

とやってしまうと、$varの結果は数値の50になるだけです。＋は数値のための演算子なので、$aも$bも数値として評価されてしまい。$a=50、$b=0となって、$varの結果は50になるのです。

あくまでも文字列としてつなぐ場合は、文字列演算子の . を使います。

```
$var = $a . $b;
```

とすれば、$varは"50円"になります。ダブルクォート内では変数が値に展開されることを利用して、

```
$var = "$a$b";
```

としてもOKです。\でエスケープする文字などがない場合などはこちらのほうがわかりやすいかもしれません。

## 処理の流れ

PHPには条件判断のif、ループのfor、while、ループを制御するbreak、continueなどが用意されています。基本的にはC言語に準じています（PHP4ではforeachが追加されています）。

### ■if文

if文は、「（条件が）・・・ならば・・・する」という、条件判断を行います。以下の例の〜の部分には比較演算子と論理演算子を含んだ比較式を記述します。

**・〜ならば……する**

```
if (〜) {
    ……
}
```

**・〜ならば……する、そうでなければ……する**

```
if (〜) {
    ……
}
else {
    ……
}
```

**・〜ならば……する、そうでなく〜ならば……する、そうでなければ……する**

```
if (〜) {
    ……
}
elseif (〜) {
    ……
}
else {
    ……
}
```

いずれも、〜が条件式で……が実行文になります。

実行文をくくる{}を使う代わりに、

```
if (〜) :
    ……;
elseif (〜) :
    ……;
else:
    ……;
endif;
```

と書くこともできます。このときはifやelseの後の:と、endif;を忘れないようにしてください。

### ■while文

while文は、whileに続く比較式が真であるあいだ、{}内の処理を繰り返し実行します。たとえば

```
while ($i>0) {
      echo "iが0になるまでループ";
      $i--;
}
```

のようになります。whileでも{}を使わずに記述することができ、その場合はendwhileを使います。

```
while ($i>0):
      echo "iが0になるまでループ";
      $i--;
endwhile;
```

### ■for文

for文は、決まった回数だけ繰り返しを行います。for文の書式は、

```
for ( 初期値; 条件式; 増減式 ) {……}
```

になり、具体例として、

```
for ( $i=0; $i<10; $i++ ) {……}
```

のように記述します。この場合は、「$iを最初に0とし、$iが10未満であるあいだ繰り返す。1回繰り返すごとに$iに1を加える」という意味になります。forを使って実際のPHPファイルを作ってみましょう。

Miracle Linuxがインストール済みなら、**リスト2**を

loop.php3というファイル名にして、/home/httpd/htmlディレクトリに置いてください。Webブラウザから、

```
http://（ホスト名またはIPアドレス）/loop.php3
```

というアドレスで呼び出すと、1から10までの数字が並んで表示されます。

うまく表示されたら、ぜひそのHTMLソースを見てください。**リスト3**のようになっているでしょう。PHPファイル（loop.php3）と、ブラウザ側のソース表示とを見比べることが、PHPマスターの早道なのです。つまり、PHPファイルのスクリプト部分を、WebサーバのPHPインタプリタで解釈された結果がブラウザのソースですから、スクリプトが思ったとおりに実行されているかどうかは、このソースが答になるのです。

ぜひとも、loop.php3をあれこれと書きかえて、そのつどブラウザ側ではどのようなHTMLソースになっているかを確認してみてください。一見ブラウザでは同じ表示になっていても、HTMLソースの異なる場合がよくあります。ただし、PHPファイルをローカルファイルとして開くと、スクリプト部分は解釈されませんので注意してください。あくまでもhttp:でWebサーバを経由しなければPHPスクリプトは実行されません。

### ■break

whileやforなどのループを実行中、途中でループを抜けたいときにはbreakを使います。

```
for (……) {
   if (〜) break;
}
```

という場合は、forループ内のif（〜）が真になったときにループを抜け出して、}の後を実行します。

PHPのbreakはC言語とはちがって、ネストした外側のループから抜け出すことが可能です。

```
for (……) {
  for (……) {
    if (〜) break;                    (1)
    if (〜) break 2;                  (2)
  }
}
```

という場合、（1）のbreakは内側のforループを抜けるだけですが、（2）のbreakは外側のforループを抜け出します。breakのパラメータとして、一番内側のループを1とした脱出ループレベル数を記述することができるのです。

### ■continue

continueはループ内の残りの処理をスキップして、ループの先頭にジャンプします。for文の場合、$i++などの増減式は1回実行されますから、繰り返しの次に進むというイメージになります。

```
for ( $i=0; $i<10; $i++ ) {
   if ( ($i % 2) == 0 ) continue;
   echo "$i<br>";
}
```

とすると、1、3、5、7、9が表示されます。理由は自分で考えてみましょう。

**▼リスト2　loop.php3**

```
<html>
<head></head>
<body>

<?php
    for ( $i=0; $i<10; $i++ ) {
         echo "$i<br>\n";
    }
?>

</body>
</html>
```

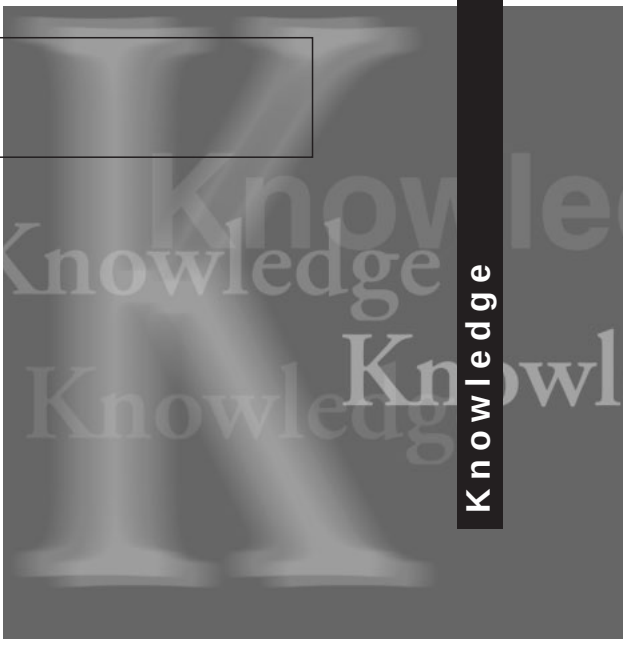
**▼リスト3　ブラウザで見たソース**

```
<html>
<head></head>
<body>

0<br>
1<br>
2<br>
3<br>
4<br>
5<br>
6<br>
7<br>
8<br>
9<br>

</body>
</html>
```

# RPMパッケージ開発講座




**今回からはRPMパッケージを実際に作る方法を解説していきたいと思います。まず、既存のソースパッケージから再構築する方法と、SPECファイルを使って生成する方法を見ていきます。**

## 第2回　RPMパッケージの作成方法

文：佐藤大輔(densuke@usi.dyndns.org)
アミュレット株式会社・開発部
*Text : Daisuke Sato*

### RPMパッケージを作成しよう

RPMパッケージを作れるようになると、自分のパッケージも管理下に置けるようになるため、なにかの都合でシステムを入れ換えた時も最小限の手間でカスタム環境の復旧も可能になります。この場合、RPMを作るための方法は、

- **既存のソースパッケージをそのまま再利用**
- **SPECファイルを作ってゼロから作成**

という方法があります。順に見ていきましょう。ただし、コンパイルが簡単に済むとは100％は言えません。その場合には、多少はソースコードを追いかけたり、makeに渡すフラグを注意しておかないといけません。

### ソースパッケージからの再構築（--rebuild）

多くのバイナリのRPMパッケージはRed Hat Linux向けに配布されていますが、みんながみんなRed Hat Linuxを使っているわけではありません。Red Hat系ディストリビューションは多様化されているため、Kondara MNU/LinuxやVine Linux、はたまたTurbolinuxの場合もあります。

最近は各ディストリビューションがFHSへの移行の過渡期にあるため、一部のディレクトリに微妙な差があったりもしますが、ほとんどの部分が共通です。Red Hat Linux用のRPMパッケージは提供されているが、自分の使っているディストリビューション用はないという場合に、ゼロからパッケージを作っていくのは面倒です。そこで、まずはソースパッケージを利用させてもらいましょう。

ソースパッケージは、各ディストリビューションのFTPサイトやCD-ROMのソースパッケージディレクトリから手に入れることができます。rpmfind（http://www.rpmfind.net/）を利用するのもいいでしょう。ソースパッケージには2つの種類があります。

- **src.rpm形式**
  **パッケージ作成の手順書となるSPECファイルと、作成に必要なソースコードとパッチがすべて収められています。**
- **nosrc.rpm形式**
  **一部（もしくはすべて）のソースコードやパッチが含まれていないソースパッケージです。**

nosrc.rpmの場合、欠けているソースコードやパッチを個別に取ってこないといけません。

ソースパッケージ（src.rpm）からインストールできるバイナリパッケージを作成するために使うのが--rebuildパラメータです。ソースパッケージを渡すことで、パッケー

ジに内包されているSPECファイルを使ってバイナリパッケージを生成してくれます。構築に必要なパッケージがそろっていれば、だいたいの場合うまくいきます。

ここでは、xalf（http://www.lysator.liu.se/~astrand/projects/xalf/）を実際に再構築してみましょう。Webサイトよりソースパッケージ（執筆時点での最新版はxalf-0.12-1.src.rpm）をダウンロードして、

```
# rpm --rebuild xalf-0.12-1.src.rpm
```

としてみましょう（**画面1**）。

最後のほうに表示される「書き込み中: /home/densuke/redhat/RPMS/i586/xalf-0.12-1.i386.rpm」という場所に、バイナリパッケージが作られます。できあがったファイルは、そのシステム用のバイナリパッケージですので、rpm -Uvhコマンドでインストールすることができます。

このように、すでにソースパッケージ化されているのであれば、一度はちゃんと作れているわけで、各自のLinux環境に問題がない限りだいたい成功します。

リビルド中に問題が起こった場合にはエラーメッセージが出力されますから、その指示を見て各自で判断しないといけません。このエラーがコンパイルやリンクのエラーの場合、「どこが悪いのか」を自分で追えないと先に進めません。いずれにしても一度src.rpm（もしくはnosrc.rpm）を展開して、どこでどうなるのかを確認してエラーを修正していきましょう。

## RPMの作成（作成ディレクトリとSPECファイル）

今度は、SPECファイルの書き方を見てから、SPECファイルを利用してRPMパッケージを作ってみましょう。ソースコードは通常、.tar.gzの形式か.tar.bz2の形式で配布されていると思います。これらのソースコードからRPMパッケージを生成させます。RPM化するからといって、あまり気を張る必要はありません。通常のソフトウェアのインストールを考えてみればいいわけです。Linux用のプログラムの多くは、

**1.ソースコードを展開する**
**2.configureしてmakeする（最近はこれで十分になってきています）**
**3.make installでインストールする**

```
[densuke@yuzu SRPMS]$ rpm --rebuild --target=i586 xalf-0.12-1.src.rpm
xalf-0.12-1.src.rpm をインストール中
作成中ターゲットプラットフォーム: i386
Building for target i586
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.60192
+ umask 022
+ cd /home/densuke/redhat/BUILD
 ⋮
Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.10190
+ umask 022
+ cd /home/densuke/redhat/BUILD
+ cd xalf-0.12
+ CFLAGS=-O2
+ ./configure --prefix=/usr
 ⋮
Provides: libxalflaunch.so.0
PreReq: /sbin/ldconfig
Requires: ld-linux.so.2 libICE.so.6 libIIOP.so.0 ...（長いので省略）
書き込み中: /home/densuke/redhat/RPMS/i586/xalf-0.12-1.i386.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.54966
 ⋮
+ rm -rf xalf-0.12
+ exit 0
```

画面1　xalfのソースパッケージから再構築する例

という順序ですが、これにプラスして、RPMパッケージ化するという部分が含まれてきます。これらの作業をまとめた手順書となるファイルが通称SPECファイルと呼ばれるものです。

### ■RPM作成ディレクトリの構成

RPMパッケージを作成するには、それ相応に準備されたディレクトリが必要です。通常は/usr/src/redhatとなっていますが、任意の場所にすることができます（方法は後述）。ここでは/usr/src/redhatとし、%_topdirと表記します（この名前は内部変数ですが重要なものでもあるため出しておきます）。このディレクトリは、**図1**のようなディレクトリ構成を伴っています。

それぞれ、

- **SPECS**
  **SPECファイルを置くディレクトリ。**
- **SOURCES**
  **ソースファイルやパッチなどを置くディレクトリ。**
- **SRPMS**
  **生成されたソースパッケージを置くディレクトリ。**
- **RPMS**
  **生成されたバイナリパッケージを置くディレクトリ、この下にアーキテクチャに対応した名前のディレクトリとnoarch（アーキテクチャに依存しないデータなどのパッケージ）のディレクトリが必要です。**
- **BUILD**
  **ソースコードを展開して実際にmakeするためのディレクトリ。**

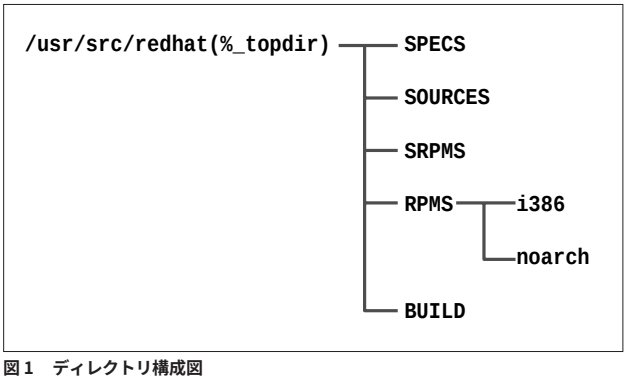



**図1　ディレクトリ構成図**

となっています。いくらなんでも本当にゼロからSPECファイルを書くのはきついので、たいていの場合、似たようなものを使って書き換えていくという方法がとられています。ここでは先に紹介したxalfのSPECファイルを読んで、内容を追っていくことにしてみましょう。実はソースパッケージをrpm -Uvhでインストールすると、SPECファイルやソースプログラムなどが%_topdir以下に適切に置かれます（**画面2**）。

### ■パッケージは誰の権限で作るのか

パッケージの作成をするのはrootでないといけないのか。実はここに重要な問題があります。一般ユーザーでもパッケージの作成ができていいはずです。普通にコンパイル→インストールというときも、コンパイルのためにrootになることはそうそうないはずです。そこで上記のディレクトリを変更する方法があります。

実は先にあげた%_topdirは、各ユーザーで個別に設定することができます。ここではユーザーdensukeが自分のホームディレクトリ内で作りたいとしましょう。まず、~/.rpmmacrosというファイルを作成し、その中で開発用ディレクトリを以下のように指定します。

```
%_topdir /home/densuke/redhat  ←%も必要です
```

あとは、先ほど説明したRPMS／SRPMS／SPECS／BUILD／SOURCESのディレクトリを、指定したディレクトリ以下で作成すればいいのです。

```
[densuke@yuzu densuke]$ mkdir ~/redhat
[densuke@yuzu densuke]$ cd ~/redhat
[densuke@yuzu redhat]$ mkdir RPMS SRPMS SPECS BUILD
SOURCES
[densuke@yuzu redhat]$ cd RPMS
[densuke@yuzu RPMS]$ mkdir i586 noarch
```

```
[root@yuzu SRPMS]# rpm -Uvh xalf-0.12-1.src.rpm
xalf                        ##################################################
[root@yuzu SRPMS]# find /usr/src/redhat -type f
/usr/src/redhat/SPECS/xalf.spec
/usr/src/redhat/SOURCES/xalf-0.12.tgz
```

**画面2　ソースパッケージをインストール**

なお、ここではKondaraによる作成なのでi586を作っていますが、たいていのディストリビューションではi586でなくi386です（IA32の場合）。LinuxPPCなどではppcですし、Alpha AXPではalphaとなります、この名前はrpmパッケージのアーキテクチャ名と同じです。

試しにxalfのソースパッケージを一般ユーザーで展開させてみましょう（**画面3**）。

## SPECファイルの詳細

では、実際にSPECファイル（xalf.spec）を読むことにしましょう。

### ■プレアンブル

プレアンブル（**リスト1**）には、パッケージのコンパイルなどには直接関係しているわけではありませんが、重要な情報が含まれています。ほとんどの部分は、

**パラメータ名: 値**

という形式となっています（最初の%defineや最後の%descriptionは別）。また、シャープ記号（#）はコメントです。この部分では、SummaryやName、VersionやReleaseといった、RPMパッケージの情報を得るコマンドを使用した時に見た文字列があると思いますので大筋理解していただけると思います。

通常のrpmコマンドで見ることのできないものとしては、Source指示子があります。これがソースファイルの名前となっていて、ここにあるファイルがSOURCESディレクトリにないといけません。そのほか、注意しておくべき指示子やコマンドとして以下のようなものがあります。

```
[densuke@yuzu densuke]$ pwd
/home/densuke
[densuke@yuzu densuke]$ cat ~/.rpmmacros
%_topdir /home/densuke/redhat
[densuke@yuzu densuke]$ rpm -Uvh xalf-0.12-1.src.rpm
xalf                        ##################################################
[densuke@yuzu densuke]$ find ~/redhat -type f
/home/densuke/redhat/SPECS/xalf.spec
/home/densuke/redhat/SOURCES/xalf-i18n-1.patch
/home/densuke/redhat/SOURCES/xalf-0.12.tgz
```

画面3　ユーザーの指定した開発用ディレクトリを使用

▼リスト1　プレアンブル

```
# Note that this is NOT a relocatable package
%define ver      0.12
%define rel      1
%define prefix   /usr

Summary: A utility to provide feedback when starting X11 applications.
Name: xalf
Version: %ver
Release: %rel
Copyright: GPL
Group: X11/Utilities
Source: xalf-%{ver}.tgz
BuildRoot: /var/tmp/%{name}-root
URL: http://www.lysator.liu.se/~astrand/projects/xalf

%description
This is a small utility to provide feedback when starting X11
applications.  Feedback can be given via four different indicators:
An invisible window (to be used in conjunction with a task pager like
Gnomes tasklist_applet or KDE Taskbar), an generic splashscreen, an
hourglass attached to the mouse cursor or an animated star.
```

- **変数定義**
  **変数は%define命令により作成できます。「%define 変数名 値」のように定義します。%変数名もしくは%{変数名}とすることで、定義された値に展開されます。**
- **%descriptionコマンド**
  **次の行から、%で始まるコマンド文字の前までをそのパッケージの説明文として記述します。**
- **BuildRoot指示子**
  **パッケージを作成した時に通常のディレクトリにインストールせず、ここに指定したディレクトリを仮想的なルートディレクトリとみなしてインストールすることを指示します。現在の開発においてはほぼ必須の機能で、詳しくは後述いたします。**

注意点として、プレアンブルで指定されたものは、自動的に変数に代入されるものがあります。今回の場合、

```
%define name xalf
%define version %{ver}
%define release %{rel}
```

が最低限含まれ、今後利用可能となります。

### ■前準備、ソース展開、コンパイル

**リスト2**の部分は、変数を展開したあとでシェルスクリプトとして動作します。

前準備を行う%prepは、ソースを展開する前に準備しておきたい作業がある場合に記述します。あまり必要性がないように見えますが、以前に同一パッケージを作った（作ろうとした）際のゴミを除去しておく目的で使われることがあります。

%setupは、ソースコードを展開します。この時、展開したソースコードは、「<ソフトウェア名>-<バージョン>」ディレクトリ以下に展開されているとみなされます（GNU autoconf／automakeを用いた場合は標準でこの形式ですし、ほとんどのものはこの形式にしています）。展開後にこのディレクトリが存在しないとエラーとなり停止しています。

しかしながら展開してもその形式にならない例外もあります。身近な例としてはLinuxのカーネルソースで、展開するとバージョンとは無関係にlinuxディレクトリが作成されてその下にファイルが置かれます。この時は、

```
%setup -n xalf
```

のように、-nオプションでディレクトリ名を明示することで対処できます。これ以外にも、**表1**のようなオプションがあります。

%buildの部分からは実際にファイルをコンパイルさせるのに必要なコマンドを記述します。ここではconfigureしてmakeするというお約束のみで済んでいますが、RPM_OPT_FLAGSという環境変数が展開されています。この変数には、あらかじめrpmコマンド側で用意されている最適化用のフラグが渡されます。

### ■仮想インストール

コンパイルが無事終了すると、続いて%installからのコマンドがシェルを通じて実行され、適当なディレクトリを起点としてインストールされます（**リスト3**）。

この中で登場する変数RPM_BUILD_ROOTには、プリアンブルのBuildRoot指示子で渡したディレクトリがそのまま入ります。GNU autoconf／automakeを使っているソフトウェアの場合のほとんどは、makeの変数prefixに渡すことで、指定したディレクトリ以下にファイルを配置

**▼リスト2　前準備、ソース展開、コンパイル指示部分**

```
%prep
%setup

%build
CFLAGS="$RPM_OPT_FLAGS" ./configure --prefix=%prefix
make
```

**▼リスト3　仮想インストール部分**

```
%install
rm -rf $RPM_BUILD_ROOT

make prefix=$RPM_BUILD_ROOT%{prefix} install-strip
```

| オプション | 説明 |
|---|---|
| -c | ディレクトリを1つ、-nで明示した名前もしくは<name>-<version>の形式で作成し、その中に展開させる |
| -q | 表示の抑制 |
| -a 数字、-b 数字 | ソースファイルが複数ある際に指定した番号のものを展開させる。-aの場合は最初のソースファイルを展開した後（after）で、-bの場合は展開する前に（before）のタイミングで実行される |

**表1　%setupのオプション指定（抜粋）**

してくれます。こうすることで、

- **すでに実際に使っているソフトウェアとかぶらないので、万一作成に失敗しても既存の環境を壊さない**
- **うまく使えば一般ユーザーでもパッケージ開発ができる**
- **もしトラブルで既存ディレクトリにファイルを置こうとしても、一般ユーザーで作成していれば権限がないのでエラーで停止してくれるから安心**

というメリットが得られます。後者を実現するには%filesの部分で、ひと手間が必要ですがそれだけの価値はあります。

実際のディレクトリにインストールするわけではないことから、仮想インストールと呼ばれています。もちろんインストールしたあとに設定が必要であれば任意のスクリプトを挿入してもかまいません。

### ■後始末

%cleanには、パッケージ作成作業が終了した時に実行される、下記のような後始末用のスクリプトを入れておきます。たいていの場合、仮想インストールの痕跡を消すだけで十分ですが、別の場所に作業ファイルを作ったという場合に削除してもいいでしょう。

```
%clean
rm -rf $RPM_BUILD_ROOT
```

### ■パッケージインストール時のスクリプト

次に書かれる%pre、%post、%preun、%postunは、パッケージ開発の際には実行されず、パッケージのインストールなどを行う際に実行されるスクリプトです。

**リスト4**のように、単一のコマンドを実行させるだけであれば、-pオプションに続けてコマンド名を書いて直接呼ばせるようにすることもできます。実行されるタイミングは、以下のようになっています。

**▼リスト4　パッケージインストール時のスクリプト**

```
%post -p /sbin/ldconfig
%postun -p /sbin/ldconfig
```

`%pre` **パッケージをインストールする前に実行します。特定ユーザーアカウントが必要な場合にあらかじめ作成する、などに用いられています。**

`%post` **パッケージをインストールしたあとに実行します。例のようにライブラリのキャッシュを作らせる（`ldconfig`）場合によく用います。**

`%preun` **パッケージをアンインストールする前に実行します。サービスが動いていないかを確認して止めておくタイミングにはいいでしょう。**

`%postun` **パッケージをアンインストールしたあとに実行します。例のようにライブラリのキャッシュを更新させて削除されたライブラリを外す場合によく用います。**

### ■パッケージに組み込むファイル指定

%filesから始まる**リスト5**の部分には、パッケージとして取り込みたいファイルを指定させます。BuildRoot指示子を用いている場合は、そのディレクトリを起点として見てくれますから、特に意識する必要はありません。

ここでの%{prefix}は、プレアンブルにて、

```
%define prefix   /usr
```

としているため、展開されて**リスト6**のようになります。

ワイルドカードも使えます。%doc指定は、ドキュメントとして扱うべきファイルを指定するもので、ソースコー

**▼リスト5　パッケージに組み込むファイル指定**

```
%files
%defattr(-, root, root)

%doc AUTHORS FAQ COPYING ChangeLog NEWS README INSTALL TODO BUGS extras
%{prefix}/lib/libxalflaunch.*
%{prefix}/bin/xalf
%{prefix}/bin/xalfoff
%{prefix}/bin/xalf-capplet
%{prefix}/share/pixmaps/hourglass-big.png
%{prefix}/share/pixmaps/hourglass-small.png
%{prefix}/share/control-center/Desktop/xalf-settings.desktop
%{prefix}/share/gnome/apps/Settings/Desktop/xalf-settings.desktop
```

ドのディレクトリ上のファイルを指定します。

■%filesによるファイルの指定

パッケージに含めたいファイルは、%filesセクションで指定します。ワイルドカードを用いて複数指定することもできます。ただし、root権限で作成しているのならともかく、一般ユーザー権限での作成をしている時には、owner／groupに問題が出ます。パッケージ化する時にはその時のファイル属性がそのまま使われてしまうからです。この問題に対処できるのが、%defattr（）プレフィックス、および%attr（）プレフィックスです。

```
%defattr(-, root, root)
```

とすることで、この指定よりあとのファイルには（%attr（）がない限り）アクセス権／owner／groupを強制することができます。書式は、

```
%defattr(アクセス権(8進数指定), owner, group)
```

で、変更の必要がなければ“－”で代用できます。

ただし、このまま全部強制されてしまうと困るという時に、個別に%attr（）を使うことができます。

（Squidパッケージでの例）

```
%attr(-, squid, squid) /var/log/squid
```

この場合、パッケージのインストールの際にユーザーsquidが存在しないと「そんなユーザーはいないのでrootにしておきます」ということになりますので、%pre文を使って、

```
/usr/sbin/useradd -u 23 -d /var/spool/squid -r -s
/dev/null squid
```

という形で作成させています。

■ログ（%changelog）

SPECファイルの更新履歴を記入していく場所です。

## RPMの作成（-bによるパッケージの作成）

適当なSPECファイルと必要なソースを用意できたら、あとは実際に作ってみるだけです。これには-bというパラメータを用います。実際には-bに作業内容を示すためのオプションを付けます（**表2**）。

それでは実際にパッケージを作成してみます。今回はバイナリまでで十分でしょう（**画面4**）。

## エラーが発生したら

エラーが発生した時、いったいなにが原因でエラーが発生したのか、これがわからないとどうしようもありません。

▼リスト6　prefix展開後はこうなる

```
%doc AUTHORS FAQ COPYING ChangeLog NEWS README INSTALL TODO BUGS extras
/usr/lib/libxalflaunch.*
/usr/bin/xalf
/usr/bin/xalfoff
/usr/bin/xalf-capplet
/usr/share/pixmaps/hourglass-big.png
/usr/share/pixmaps/hourglass-small.png
/usr/share/control-center/Desktop/xalf-settings.desktop
/usr/share/gnome/apps/Settings/Desktop/xalf-settings.desktop
```

| オプション | 説明 |
|---|---|
| -bp | %prep～%setup、（あれば）%patchまでを実行する |
| -bc | -bpに加え、%buildまで実行する |
| -bi | -bcに加え、%installまで実行する |
| -bb | -biを実行し、実際にバイナリパッケージを作成する |
| -ba | -bbに加え、ソースパッケージを作成する |
| --clean | -b系パラメータと別に指定する。作業が終了したあとでBUILDディレクトリ以下に展開した作業ディレクトリを削除する |
| --target=<arch> | 作成するアーキテクチャを指定する。通常は必要ないが、Kondara MNU/LinuxやSuSEのように、特定アーキテクチャに最適化を行ったパッケージにしたい場合に用いる（例：rpm -ba --target=i586。この場合RPMSディレクトリに対応するディレクトリが存在しないとエラーになるので、あらかじめ作成しておくといい |

表2　RPMパッケージを作成するオプション

多く発生するエラーとしては、

- **ヘッダ／ライブラリの不足**
  **configureの出力を見てみると、実はライブラリが不足している。リンクやヘッダの読み込みで失敗して止まっている場合はたいていこれです。不足しているものを探してインストールしてから再度rpm -bbしてみましょう。**

- **コンパイル時のエラー**
  **あなたのCのプログラミング能力に頼るしかありませんが、いったい何のエラーなのかはよく見ないといけません。**

- **%filesで指定したファイルが存在しない**
  **RPM_BUILD_ROOT以下で探してみて、実際にファイルがあるか確認してみましょう。**

こういう時に役立つかもしれないパラメータとして、--short-circuitというものがあります。rpmのオプションに指定すると、「やりたいこと」の前までは「すでに終わってる」ものとして飛ばしてくれる機能です。

```
[densuke@yuzu SPECS]$ rpm -bi --short-circuit --
target=i586 xalf.spec
```

とすると、「-bb（展開、コンパイル）までは終了している」とみなされ、いきなり%install以降を実行することができます。ただし-bb／-ba（パッケージ生成）のスイッチと--short-circuitは併用できません。

## 次回予告

ここまでで、たいていのパッケージの作り方が追えるようになると思います。しかし、エラーが出てきたときの対処として、本来のソースと違った処理が必要となったら、パッチを作って対処しないといけません。

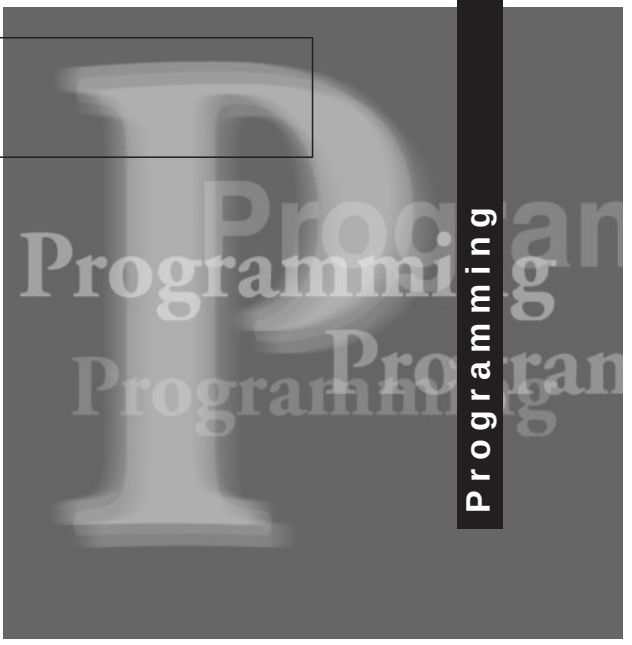
次回は、パッチの作成と適用方法（%patch命令）および、ちょっと特殊（？）なパッケージ生成について考えてみたいと思います。

```
[root@yuzu SPECS]# rpm -bb --clean xalf.spec
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.19708
+ umask 022
+ cd /usr/src/redhat/BUILD
+ cd /usr/src/redhat/BUILD
 ⋮
(省略)
 ⋮
Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.99519
+ umask 022
 ⋮
(省略)
 ⋮
+ CFLAGS=-O2 -m486 -fno-strength-reduce ← $RPM_OPT_FLAGS
 ⋮
(省略)
 ⋮
Executing(%install): /bin/sh -e /var/tmp/rpm-tmp.8002
+ umask 022
+ cd /usr/src/redhat/BUILD
+ cd xalf-0.12
+ rm -rf /var/tmp/xalf-root
+ make prefix=/var/tmp/xalf-root/usr install-strip
 ⋮
(省略)
 ⋮
書き込み中: /usr/src/redhat/RPMS/i386/xalf-0.12-1.i386.rpm
 ⋮
(省略)
 ⋮
+ rm -rf xalf-0.12
+ exit 0
```

画面4　SPECファイルを利用したRPMパッケージの作成

# プログラミング工房

**サーバ管理システム第3回目の今月は、ファイルサーバ管理部分について取り上げる。特に、共有ファイルを提供するうえで非常に重要なファイルサイズ制限について、詳しく説明したい。**




## 第20回　サーバ管理プログラム（3）

文：藤沢敏喜
*Text : Toshiki Fujisawa*

### どんどん増え続けるファイル

数年前に比べれば、ハードディスクの容量は飛躍的に増加した。筆者が最初に使ったハードディスクは10Mバイトくらいだった思うが、その時代にはGバイト単位の容量は「無限に広がる大宇宙」であり、とても使い切れるものではなかった。

ところが、人間というものはあればあるだけ使うもので、自宅にあるテレビ録画専用PCに内蔵された160Gバイトのハードディスクは、100時間分のドラマや映画ですでに満杯状態となっている。

また、少し前にADSLの工事が終わり、実測で1Mbps程度の常時接続環境となったため、600MバイトくらいのCDイメージが1時間半ほどでダウンロードできるようになった。こうなると、NetBSDやFreeBSDのCVSなどの膨大な容量のファイルを気軽にダウンロードするようになり、dfコマンドの実行結果は常に100%に近い状態となってしまっている。

### バックアップと管理可能なファイル容量

仕事や研究をする場合に、多くのメンバーでファイルを共有する場面が増えてきた。表計算プログラムのファイルやプレゼンテーション用のとてつもない大きなファイルなどが、ディスクスペースをどんどん占有していくのである。

現在は、80Gバイトのハードディスクが2万6000円程度で購入できるので、ディスクの容量を増やすことは簡単にできる。しかしながら、データ量が増えれば増えるほど、バックアップすることは指数関数的に難しくなってしまう。

仕事や研究で使ったデータは、長い間確実に保存しておかなければならないことが多いため、提供するデータ容量はバックアップのことを考えたうえで決定する必要がある。

サーバ管理を始めて間もない人は、RAIDによるミラーリングだけで安心することが多いようであるが、ミラーリングは誤ったデータまでミラーする。RAM不良など（結構ある）が発生した場合、両方のディスクのデータが化けてしまう。また、最も頻発するトラブルは、人間が誤ってファイルを消してしまうことなのである。

これを避けるためには、ハードディスクを複数用意して、定期的にRAIDの再構成をする必要がある。しかし、ディスクを何回も抜き差しすると接点不良が起きたり、抜き差しするときにディスクを落としてしまうといった事故が発生することもありえる。RAIDは連続運用を保証するために用いたり、個人が行うバックアップとしては、コストパフォーマンスに優れていると思うが、業務で（本来の意味の）バックアップとして用いるには不安も多い。

昔から、「データのバックアップはテープに限る」と言

われているが、今回想定している研究室や部門でのファイル共有のような用途では、現在でもやはりテープによるバックアップが最適だと筆者は感じている。

テープドライブは、リストア時にコピー元とコピー先を間違うようなミスが起きにくいのが第一の利点である。そして履歴が確実に残せるということが最も重要な点である。月火水木金、毎月第1週～第5週の月曜日、1月～12月の月初め、2001年～2005年の年初、というように27本のテープを使ってバックアップを行えば、必要なデータをほぼ確実にバックアップできる。さらに、ハードディスクと比べてカートリッジのサイズが小さく、衝撃に強いため、輸送が簡単な点も見逃せない。遠隔地に保存しておくことにより、火災や地震などでシステムが壊滅的な被害を受けた場合でも、復旧が可能なのである。

なお、1つのマシンのデータを複数のテープにバックアップすると、管理の面（人間のミス）から意外にトラブルとなることが多い。リストア時間を考えても、1本のテープに収まる程度の容量が現実的である（DLTのスピードでも、40Gバイトのリストア時間は非常に長い）。

したがって、1台のサーバで管理できるデータ量は、使用するテープの容量に制限されることになる。

## 無制限に増え続けるファイル

上記のように、バックアップ装置の制限から、1台のサーバできちんと管理できるデータ量は、数Gバイト～40Gバイト程度である。そのため、この容量を絶対に超えないようなサーバ運用をする必要がある。

ディスク領域を管理するために、各部署に一定容量のデータ領域を割り当て、その容量を超えないように管理する人を任命することがよく行われる。

しかし、筆者の経験によれば、ディレクトリ領域を使用する人が数十人を超えると、この方法は必ず破綻するようである。なぜなら、業務で使用する場合には、ファイルの所有者が明確でなくなることが多く、いらないファイルを消すということがきわめて難しくなるからである。特に、ディレクトリ領域の管理は、権限のない若いスタッフが担当させられることも多く、自分の判断だけでは整理ができないのである。

そのため、ファイルは無制限に増え続け、パーティションの切り方によっては、メールが受けられなくなったり、システムの動作にまで影響が出ることになる。

## 容量制限をかける方法

このように、容量の制限を人間が手動で管理することはたいへん難しく、自動的に確実な制限がかかるように設定する必要がある。

制限をかける最も簡単な方法は、提供するディレクトリ単位でパーティションを分ける方法である。この方法はシンプルなので、管理も楽でありお勧めできる。ただし、この方法はあとからパーティションのサイズを変更することが難しく、パーティションを変更する際に、誤ってシステムを破壊するという危険性がある。そのため、サーバの運用開始時には、将来を見据えた綿密な設計が必要になる。

なお、パーティション方式のバリエーションとしては、ループバックデバイスを用いる方法もある。ループバックデバイスとは、1つのファイルをあたかも1つのパーティションに見せかけるデバイスで、iso9660のCDイメージを作成する際などによく使われる。ループバックデバイスを使えば、パーティションのサイズ変更は比較的簡単であるが、速度的な面では不利となる。

さて、容量制限をかける方法の本命が「ディスククオータ」である。クオータを使うと、各ユーザーごとあるいは各グループごとに総使用ファイルサイズの制限をかけることができる。それぞれの制限容量は、edquotaコマンドにより簡単に変更することができ、quotacheckコマンドにより使用量と制限容量の一覧を得ることができる（詳しくは、http://www.linux.or.jp/JF/JFdocs/Quota.htmlを参照のこと）。

しかしながら、ディスククオータはあくまでも、ユーザーID単位やグループID単位でしか制限をかけることができない。したがって、あるディレクトリ下に多数のユーザーが書き込みを行うような状況で、その「ディレクトリ単位」での容量制限をかけたい場合は工夫が必要となる。

## クオータを使った共有単位での容量制限

sambaを使ったファイル共有を行う場合、容量制限を実現するために、Windowsの「共有」単位でグループクオータをかける方法がよく紹介される。sambaの設定ファイルであるsmb.confのオプションには、force groupという設定がある。これを指定すると、その共有名に対して、接続したすべてのアクセスにおいて指定されたグループIDの権限でアクセスが行われるようになるのである。

したがって、共有名ごとに専用のグループを作成して、/etc/groupに登録しておき、そのグループに対してグループクオータをかけることにより、共有単位での容量制限を実現することができる。

しかしながら、この方法では、あくまでもWindowsの共有単位でしか容量制限をかけることができないため、それぞれのディレクトリをドライブとしてマウントする必要がある。共有する数が少ない場合はこれでもいいが、数が増えた場合には、どのデータがどのドライブにあるかを把握することが難しくなるし、ドライブレター数（e:～z:）以上のディレクトリを使用することはできない。

また、Netwareなどから移行する場合には、ドライブ単位の共有に拒否反応を示すユーザーも多いようである。

## クオータを使ったディレクトリ単位での容量制限

あるディレクトリ下の総ファイルサイズに容量制限をかける方法として、BSD方式のディレクトリ所有グループ継承を用いる方法がある。

BSD系のOSでは、ファイルが作成される際に、親ディ

## Column

### テープドライブ

今のところ、テープドライブとしてはDAT、AIT、そしてDLTがよく使われているようである。DATは比較的小さな4mm幅のテープを使うものであり、最新のDDS-4では非圧縮で20Gバイトを実現している（図1）。

筆者は90メートルテープを使うDDS-1の時代から、十数台のDATドライブを使用したが、世代が進むにつれテープが薄くなったせいか、DDS-3ではテープが切断するという事故を何回か経験した。信頼性という面では後述のDLTには及ばないため、予備ドライブを買っておくなどの運用上の工夫が必要だが、値段が手頃なのは魅力である。

DATより大きなテープを使うAITは、非圧縮で35Gバイトの容量が記録できる（画面1）。DDSより高速で安心感もあるが、回転ヘッドを使用しているためローディングしているようすを見ていると、多少弱々しい印象を受ける。今までトラブルを経験したことはないが、AITはあまり多くの台数を使用したことがないせいかもしれない。

さて、テープドライブの最高峰はなんといってもDLTである。非圧縮時の容量は40Gバイトとなっている。DATやAITに比べるとそのドライブは大きく重いが、その重さに信頼性を感じる。また、DLTの魅力はデータ転送速度が非常に速い点である。テープからリストアしているときは気が動転している場合が多く、何度もリストアする羽目になることもしばしばある。DATだと徹夜作業になる場合でも、DLTでは終電に間に合うのである。また、テープカートリッジも大きく、剛性が高い。DLTはDATやAITのようなヘリカルスキャンによる回転ヘッドではなく、メカ構造的にシンプルな固定ヘッドである点も重要なポイントのように思える。さらに、ローディングレバーの「ガッシャン」という操作感など、実際に使用してみると非常に安心感がある。

DLTの欠点は値段が高いことで、ドライブの定価が80万円で、一緒にテープを50本買うと130万円コースとなる。しかしながら、業務や研究データを失うことに比べれば安いものである。重要なデータであればあるほど、DLTを導入すべきであろう。ちなみに、DLTは重いので、移動には台車が必要である。ネットワーク経由でバックアップはできても、リストア時には、サーバが設置されているサーバルームまで、DLTを運ばなくてはならないこともある。高価ではあるが、予備という意味も含めて、必要な場所にあらかじめ設置しておけるだけの台数を購入したいものである。

なお、最近はDLT1という安価なドライブもある（図2）。こちらのほうは、まだ実際には使用したことがないのでなんともいえないが、スペックと値段は魅力的である。

図1　SONY SDT-11000のカタログ

画面1　SONY SDX-400CのWebページ


図2　HP surestore dlt1

レクトリの所有グループが継承される。たとえばFreeBSDでは、max50mbという名称のグループを/etc/groupに登録し、ディレクトリのグループをmax50mbに変更することにより、そのディレクトリ中に作成されるファイルをすべてmax50mbというグループにすることができる。したがって、max50mbグループに対して、50Mバイトのグループクオータをかけることにより、特定ディレクトリ下の容量制限を行うことが可能になるのである。

このページのコラムで説明しているように、設定によってLinuxでもBSD方式のグループ継承を使えるようにできるので、ディレクトリ単位での容量制限も可能であるはずだ。

ちなみに、ディレクトリ単位でグループパーミッション権限を適切に設定すると、セキュリティも向上する。今回のシステムのように、シンボリックリンクを用いたアクセス許可のみでは、各ユーザーのFTPやtelnetアクセスを禁止する必要があるが、グループアクセス権限を適切に設定することにより、それらをユーザーに解放することも可能になる。

なお、このようなグループクオータを使う場合には、いくつか注意すべき点がある。まず、数百人が使用するサーバでは、共有ディレクトリもかなりの数になり、/etc/groupが巨大になるという問題がある。そのため、行の文字数に制限があるエディタでは/etc/groupの編集が行えなくなるので、usermodなど専用のコマンドを用いる必要がある。

そして、1人が所属できるグループに制限があることにも注意が必要である。たとえばFreeBSDでは、

```
include/sys/syslimits.h
```

にある、NGROUPS_MAXが16になっているので、1人のユーザーが16を超えるグループに所属するこはできない。そのため、所属するグループをもっと多くしたい場合は、この値を変更し、カーネルとすべてのコマンドを再構築（FreeBSDでは、cd /usr/src ; make world）する必要がある。

一方、Linuxでは、

## Column

### 親ディレクトリのグループ継承

いわゆるUNIX環境では、ファイルを作成する際に、そのファイルの所有者情報と所有グループ情報が付加される。しかしながら、この所有者情報と所有グループ情報は、オペレーティングシステムによってその振る舞いが違う。FreeBSDなどのBSD系OSでは、親ディレクトリのグループを引き継ぐのに対し、Linuxでは自分の所属するグループになってしまうのである。

たとえば、max50mbというグループ名を/etc/groupに登録し、

```
# mkdir tmp
# chown root:max50mb tmp
```

として、

```
# ls -ld tmp
```

を実行すると、tmpディレクトリの所有ユーザーと所有グループは、

```
drwxrwxr-x 2 root max50mb 1024 May 25 23:58 tmp/
```

となる。ここで、

```
# cd tmp
# touch foo
# ls -l
```

を実行すると、Linuxでは、

```
-rw-rw-r-- 1 root root 0 May 25 23:58 foo
```

となり、/etc/passwdで登録された自分のグループが使用されてしまう。一方、FreeBSDでは、

```
-rw-rw-r-- 1 root max50mb  0 May 25 23:58 foo
```

となり、親ディレクトリのグループを引き継ぐことができ、たいへん便利である。

しかしながら、Linuxでも、

```
mount -o bsdgropus /dev/XXX /mnt
```

としてマウントしたり、/etc/fstabでbsdgroupsオプションすると、BSDと同じ振る舞いにすることができる。

なお、筆者には、Linuxのデフォルト動作が有利になる場面が想定できず、なぜこのような仕様になっているかという疑問を感じた。そこで識者に質問してみたところ、

「おそらくLinuxの振る舞いがUNIXのもともとの仕様だったのだが、不便だったのでBSDでは仕様を変えたのだろう。それが便利だったので、System V系も互換性のためにオプションという形でこれを導入したのではないだろうか」

という回答を得た。

ちなみに、上記のbsdgroupsオプションは、Vine Linux2.0で正しく動作することを実際に確認した。しかし、FreeBSDと違い標準仕様ではないので、Linuxで大規模に使う場合には、実験してから実際の運用をするほうがよさそうである。

```
include/linux/limits.h
```

で、NGROUPS_MAXが32に設定されている。Linuxではシステム全体の再構築が難しいディストリビューションが多いように思われるので、デフォルトである32の範囲内で使用するのが無難ではないだろうか。

## 親ディレクトリのグループを継承できない場合

上記のように、親ディレクトリのグループを継承することにより、特定のディレクトリ下のすべてのファイルを特定の所有グループIDとすることが可能になる。

しかしながら、このやり方でも問題が起きる場合が存在する。それは、ディレクトリを「移動」してしまった場合である。Windowsでは、マウスでドラッグすることによりディレクトリを簡単にすることが移動できてしまう。

現在のsambaの実装では、このようにしてディレクトリを移動してしまった場合、内部で、mvコマンドが実行されるのと同様の動作が行われる。すなわち、renameシステムコールが発行されることになる。

そのため、たとえば設計グループディレクトリの中にあるディレクトリを、生産グループのディレクトリにマウスでドラッグした場合、そのディレクトリ下のファイルの所有者が生産グループに変更されず、設計グループの所有者のままになってしまうという問題が発生する（**図3**）。

この問題を避けるには、Ctrlキーを押しながらフォルダをドラッグしていったんコピーして、コピー後に元のファイルを消すように、ユーザーを教育するという方法がある。しかし、これをWindowsユーザーに徹底するのはほとんど不可能であり、結果としてディレクトリの所有グループがめちゃめちゃになり、容量制限が適切に適用されていないというクレームが発生することになるのは明らかであろう。

## sambaのソースコード変更

フォルダを移動しても、所有グループが正しく継承されるようにする方法として、sambaのソースコードを変更するという技がある。ソースコードが公開されていないシステムでは、このように柔軟な対応をとることはできないが、オープンソースのプログラムなら自由自在に変更が可能なのである。

samba-2.0.5aの場合、移動を実現しているソースコードは、lib/doscalls.cの中にある、

```
int dos_rename(char *from, char *to)
```

という関数である。

上記の関数では、fromとtoが移動するパス名を示している。このfromとtoは、

```
pstrcpy (zfrom, dos_to_unix (from, False));
pstrcpy (zto, dos_to_unix (to, False));
```

というようにして、dosのファイル名からUNIXのファイル名に変換され、zfromとztoという名前の変数に代入される。そして、

```
rcode = rename (zfrom, zto);
```


図3　フォルダをマウスで移動した場合にグループが継承されない例

というようにして、renameシステムコールが発行されることになる。

今回の目的を達成するためには、このrenameを実行する行の前に、

```
+ #if 1 /* By Toshiki Fujisawa  */
+     if( ! is_same_group_rename(zfrom, zto ) ){
+         /* 'COPY' data insted of 'RENAME' */
+         rcode = copy_same_group(zfrom, zto);
+         return rcode;
+     }
+ #endif
```

というコードを追加すればよい。

上記のis_same_group_renameは**リスト1**で定義される関数で、zfromとztoの所有グループが同じかどうかを判断する。すなわち、所有グループが異なっている場合は、renameの代わりにcopy_same_group関数が呼ばれることになる。また、copy_same_group関数は**リスト2**に示す関数で、zfromからztoへファイルをコピーする関数である。

ここで、**リスト1**と**リスト2**は、dos_rename関数の前に追加すればよい。最新のsamba-2.2.0でも、この部分のソースコードは変わっていないので、同様に適用できるはずだ。なお、この変更は、sambaの漢字コーディングがHEXであることを前提としており、2バイト目にスラッシュが存在するSJISなどには対応していない。

ちなみに、筆者の経験では、この変更をする以前はマウスのドラッグで一瞬にしてディレクトリが移動してしまうため、数百Mバイトのフォルダが（無意識のうちに）どこかにいってしまうという事故がよく起こっていた。しかし、このコードの追加によって、移動に時間がかかるようになり、誤ったフォルダ移動を行うとユーザーがすぐに気がつくようになったため、そのような問題は激減した。

上記の実装以外に、renameで移動してから所有者を変える方法もあるが、ディスククオータをかけている場合には、今回のようにファイルを1つずつコピーするようにして、制限容量いっぱいまで適切な処理が行われるようにし

**▼リスト1　s_same_group_rename関数**

```
/* Copyright (c) 2000,2001 Toshiki Fujisawa
License: GPL2 */

static int
get_group_id(char *path)
{
    struct stat info;

    if( stat(path, &info ) == -1 ){
        return -1;
    }
    return info.st_gid;
}

static char *
alloc_dirname( char *path )
{
    char  *p;
    char  *last = NULL;
    char  *buf;
    int    len;

    for( p=path; *p ; ++p ){
        if( *p == '/'  ){ last = p; }
    }
    if( last == NULL ){
        return NULL;
    }
    len = last - path;
    buf = malloc( len + 1);
    memcpy(buf, path, len);
    buf[len] = '\0';
    return buf;
}
```

```
static int
is_same_group_rename(char *src_arg, char *dst_arg)
{
    char  *dst_dir = alloc_dirname(dst_arg);
    int    dst_gid, src_gid;

    if( dst_dir == NULL ){
        DEBUG(0,("not found dir part=[%s]\n",
            dst_arg));
        return 0;
    }
    if( (dst_gid = get_group_id(dst_dir)) == -1 ){
        DEBUG(0,("can not get gid [%s]\n",
            dst_dir));
        free(dst_dir);
        return 0;
    }
    free(dst_dir);

    if( (src_gid = get_group_id(src_arg)) == -1 ){
        DEBUG(0,("can not get gid [%s]\n",
            src_arg));
        return 0;
    }

    if( src_gid == dst_gid ){
        /* DEBUG(0,("RENAME OK from=[%s]
            to=[%s]\n", src_arg, dst_arg)); */
        return 1;
    }else{
        /* DEBUG(0,("GROUP RENAME NG from=[%s]
            to=[%s]\n", src_arg, dst_arg)); */
        return 0;
    }
}
```

たほうが有利である。

なお、このコードを追加したsamba-2.0.5aを使用したFreeBSDマシン十数台で、半年ほど大規模に運用してみたが、特に問題は発生しなかった。

## ファイルサーバ管理のポイント

サーバを安定運用するためには、提供するデータ容量をできる限り少なくすることが望ましい。現在のバックアップデバイスやディスクの転送速度を考えると、各部署に与えるディレクトリは、各部署のディレクトリ管理担当者がCD-RやMOに保存できる600Mバイト以内とするのが無難であろう。

また、毎日変更される可能性のあるデータ領域は、なるべく小さくしておくべきである。一方、リードオンリーのデータの管理は易しいので、定期的にCD-Rやリードオンリーのフォルダに移動するといった運用面の工夫も必要である。提供するデータ容量を増やすのは簡単だが、数年に渡ってそのデータを確実に保守していくことはきわめて難しいのである。

▼リスト2　copy_same_group関数

```
static int
copy_same_group(char *source, const char *dest)
{
    SMB_STRUCT_STAT source_stats;
    int ifd;
    int ofd;
    char *buf;
    int len;    /* Number of bytes read into `buf'.
*/

    sys_lstat (source, &source_stats);
    if (!S_ISREG (source_stats.st_mode))
        return 1;

    if (unlink (dest) && errno != ENOENT)
        return 1;

    if((ifd = sys_open (source, O_RDONLY, 0)) < 0)
        return 1;

    if((ofd = sys_open (dest, O_WRONLY | O_CREAT |
                        O_TRUNC, 0600)) < 0 )
    {
        close (ifd);
        return 1;
    }

    if((buf = malloc( COPYBUF_SIZE )) == NULL)
    {
        close (ifd);
        close (ofd);
        unlink (dest);
        return 1;
    }

    while ((len = read(ifd, buf, COPYBUF_SIZE)) > 0)
    {
    if (write_data(ofd, buf, len) < 0)
    {
      close (ifd);
      close (ofd);
      unlink (dest);
      free(buf);
      return 1;
    }
    }
    free(buf);
    if (len < 0)
    {
        close (ifd);
        close (ofd);
        unlink (dest);
        return 1;
    }

    if (close (ifd) < 0)
    {
        close (ofd);
        return 1;
    }
    if (close (ofd) < 0)
        return 1;

    /* chown turns off set[ug]id bits for non-root,
        so do the chmod last.  */

    /* Try to copy the old file's modtime and
                                access time.  */
    {
        struct utimbuf tv;

        tv.actime = source_stats.st_atime;
        tv.modtime = source_stats.st_mtime;
        if (utime (dest, &tv))
            return 1;
    }

    /* Try to preserve ownership.  For non-root
                        it might fail, but that's ok.
       But root probably wants to know, e.g.
                        if NFS disallows it.  */
    #if 0
    if (chown (dest, source_stats.st_uid,
        source_stats.st_gid) && (errno != EPERM))
        return 1;
    #endif

    if (chmod (dest, source_stats.st_mode & 07777))
        return 1;

    unlink (source);
    return 0;
}
```

特別講座

# ステップアップC言語

## ファイル情報を得るstatシステムコール

■ファイルの情報

いわゆるUNIX環境のファイルシステムでは、1つのファイルは次のようにさまざまな情報を持っている。

- i-node番号
- 許可モード
- ハードリンクの数
- 所有者ID
- 所有グループID
- 作成時間
- 最終アクセス時間
- 最終変更時間
- ファイルのバイト数

これらの情報を調べるには、statシステムコールを使用する。statシステムコールは、次のような関数として実装されている。

```
int
stat(const char *path, struct stat *sb)
```

ここで、pathは調べるファイルのパス名であり、sbはstat型の構造体で、そのメンバは、sys/stat.hで定義されている。

■statシステムコールの使用例

statシステムコールは、たとえば次のようにして使用される。

```
#include <sys/types.h>
#include <sys/stat.h>

int
main(void)
{
    char *path = "file.txt";
    struct stat sb;

    stat(path, &sb);

    printf("size=%d\n",sb.st_size);
    printf("mode=%o\n",sb.st_mode);
    printf("uid =%d\n",sb.st_uid );
    printf("gid =%d\n",sb.st_gid );
    return 0;
}
```

ここで、

```
# echo abcd >     file.txt
# chown 1234.5678 file.txt
# chmod 4755      file.txt
```

として、file.txtを用意して、上記のプログラムを実行すると、

```
size=5              ※ファイルサイズ
mode=104755         ※許可モード（8進数）
uid=1234            ※ユーザーID
gid=5678            ※グループID
```

という結果が得られる。ここでmodeは8進数で表されるので、printfのフォーマットにオクタルを示す%oを指定している。modeの上位3桁の104は、ヘッダファイルで、

```
#define S_IFREG 0100000
#define S_ISUID 0004000
```

として定義されるビットの論理和で、このファイルが通常ファイル（FREG）であり、セットユーザーIDビット（SUID）が指定されていることがわかる。

もし、pathで指定されたファイルがディレクトリだった場合は、S_IFREGの代わりにディレクトリを意味する、

```
#define S_IFDIR  0040000
```

というビットが立つことになる。

■lstatシステムコール

上記のfile.txtがシンボリックリンクだった場合を考えてみよう。

試しに、上記で作成したfile.txtを、

```
# mv file.txt real-file.txt
# ln -s real-file.txt file.txt
```

というようにして、file.txtがreal.txtを指し示すシンボリックリンクに変更してみる。この状態で、先ほどのプログラムを実行しても、先ほどと同じ結果が得られるはずである。

次に、上記のプログラムの、

```
stat(path, &sb);
```

の行を、

```
lstat(path, &sb);
```

と変更して実行すると、

```
size=13
mode=120755
gid =6104
uid =0
```

という結果が表示される。この場合、サイズが13バイトになっているが、これは、そのシンボリックリンク自体のサイズ、つまり次のように、real-file.txtの文字数を示している。

```
r e a l - f i l e . t x t
1 2 3 4 5 6 7 8 9 0 1 2 3
```

すなわち、lstatではstatと違いシンボリックリンク自体の情報を得ることができるのである。

なお、シンボリックリンクの内容を得るには、readlink関数を用いることになる。詳しくは、先月号のsedusr.cを読んでみてほしい。

# Rubyで作るWeb日記システム

**前回はWeb日記の更新を行うCGIを作成しました。今回は日記の参照機能を作成し、Web日記システムを完成させます。**


## 日記更新システムの作成（後編）

文：ただただし
*Text : TADA Tadashi*

### ユーザー認証

日記の更新ができるようになりましたから、あとは読むほうの参照機能さえできれば日記としての体裁は整います。さっそく実装、といきたいところですが、その前に考えておかなければならないことがあります。

前回作った更新機能は、日記のオーナーであるあなただけがアクセスできるべきです。でないと、何者かによって身に覚えのない日記を書かれてしまいかねません。まぁ、それはそれで面白いかもしれませんが。

そこで、参照と更新のCGIスクリプトを分けて、更新用のスクリプトにはパスワードでロックをかけることにします。スクリプト自身にその機能を持たせてもよいのですが、ここはApacheの持つ認証機能を利用しましょう。まず、Apache付属のhtpasswdコマンドを使ってパスワードファイルを作ります。

```
$ htpasswd -c /home/hoge/.htpasswd hoge
New password: （←パスワードを入力）
Re-type new password: （←確認のためにもう一度）
Adding password for user hoge
```

ここでは、ユーザー「hoge」のパスワードを、/home/hoge/.htpasswdというファイルに保存しています。オプションの-cは新たにファイルを生成するという意味です。通常このファイルはWWW経由でアクセスできない場所に作成します。ISPによってはシェルが使えない代わりに、CGIを使ってこのファイルを生成できるようなサービスを提供しているところもあるようです。

パスワードの設定ができたら、それを使って認証を行うようにします。前回作ったindex.rbは参照用に使うことにして、update.rbに名前を変えて更新用のスクリプトにしましょう。さらに前回作った、.htaccessを**リスト1**のように書き換えます。

「<Files update.rb>～</Files>」で囲まれた部分が追加されています。これはupdate.rbに対するアクセス時に以下のような動作を指定しています。

**・認証時のプロンプトとして「tDiary」を表示（AuthName）**

▼リスト1　認証用の設定をした.htaccess

```
Options ExecCGI
AddHandler cgi-script .rb
DirectoryIndex index.rb

<Files update.rb>
AuthName      tDiary
AuthType      Basic
AuthUserFile  /home/hoge/.htpasswd
Require user  hoge
</Files>
```

- **認証方式は「Basic」（AuthType）**
- **パスワードファイルは「/home/hoge/.htpasswd」（AuthUserFile）**
- **利用できるユーザーは「hoge」のみ（Require user）**

これでupdate.rbにアクセスすると、**画面1**のようなダイアログボックスが出て、正しいユーザー名とパスワードを入力しないと先に進めなくなります。

## eRuby

前回、更新結果を表示するために、スクリプト中にヒアドキュメントとしてHTMLを埋め込みました。日記参照スクリプトにも同じ手が使えますが、ちょっと待ってください。

通常の日記システムでは、デフォルト表示で最新の数日分の日記が読めます。これは、HTMLの生成にループが入るということを意味します。さらに、場合によってはさまざまな条件分岐が入ってくるでしょう。ヒアドキュメント中でこのようなループや条件分岐を記述するのはけっこう骨です。かといってスクリプト側の都合に合わせてヒアドキュメントを分割すると、可読性をかなり低下させてしまいます。

これは主に、本質であるHTMLがスクリプトに従属している、主従逆転の結果です。本来あるべきHTMLをベースにして、その中にスクリプトを埋め込むことができれば、この問題はかなりすっきり解決します。

インターネット上にCGIによるサービスが増えてこの問題が顕在化してくると、おなじみのPHPやASP、JSPなど、さまざまな「埋め込み」テクノロジーが登場してきました。RubyにもeRubyというテクノロジーがあって、複雑なCGIの構築に大きく役立っています。今回は、日記の参照システムをこのeRubyを使って実現しましょう。


**画面1　ユーザ認証ダイアログ**

eRubyは、上で説明したようにテキストにRubyスクリプト（の断片）を埋め込んだものです。埋め込む先は別にHTMLである必要はありません。たとえば以下のような内容のテキストをeRubyに通すと、

```
私のユーザー名は<%= ENV['USER'] %>です。
```

結果は以下のようになります。

```
私のユーザー名はhogeです。
```

テキスト中に埋め込んである、HTMLタグを拡張したような特殊なタグがポイントです。<%= 〜%> というタグで囲まれたRubyスクリプトは評価された結果が文字列化されてテキスト内に埋め込まれます。また、スクリプトを評価はするが結果を埋め込まない、<%〜%> というタグもあります。この2種類のタグだけを使ってテキスト中にRubyスクリプトを埋め込むことで、動的に内容が変化するテキストファイルが作れるわけです。eRubyに関する解説は最近よく見られるようになってきたので、詳しくはあとで実際に使いながら見ていくことにしましょう。

eRubyの実装には、前田修吾さんによるオリジナルのerubyと、咲さんによるerbがあります。erubyはC言語で書かれていて、mod_rubyと組み合わせた高度な運用もできるのが特徴です。erbはRubyだけで書かれていて、コマンドだけでなくライブラリとして使いやすくなっています（最近はerubyも拡張ライブラリとしての使い方ができるようになりました）。

今回は「Rubyの入っているISP上でCGIを動かす」という前提で話を進めていますので、コンパイルの必要なerubyよりも、コピーするだけで使えるerbのほうが使いやすそうです。さらにerbのライブラリをCGIスクリプトの中から直接使って、柔軟な利用を考えてみましょう。

erbは以下にあります。最新版は1.3.3です。

```
http://www2a.biglobe.ne.jp/~seki/ruby/erb.html
```

もしrootになれる環境であれば、付属のインストーラを使ってsite_rubyにインストールしてしまうのがいいでしょう。パッケージを適当なディレクトリで展開して、**画面2**のように実行します。

しかし、ISPの自分のホームディレクトリでerbを使いたい場合は、こうはいきません。そんな時は、CGIスクリ

```
$ su
# ruby install.rb
lib/erb/strio.rb -> /usr/local/lib/ruby/site_ruby/1.6/erb/strio.rb
lib/erb/main.rb -> /usr/local/lib/ruby/site_ruby/1.6/erb/main.rb
lib/erb/erbu.rb -> /usr/local/lib/ruby/site_ruby/1.6/erb/erbu.rb
lib/erb/erbl.rb -> /usr/local/lib/ruby/site_ruby/1.6/erb/erbl.rb
lib/erb/erbcgi.rb -> /usr/local/lib/ruby/site_ruby/1.6/erb/erbcgi.rb
lib/erb/erb.rb -> /usr/local/lib/ruby/site_ruby/1.6/erb/erb.rb
lib/erb/compile.rb -> /usr/local/lib/ruby/site_ruby/1.6/erb/compile.rb
```

画面2　付属のインストーラを使ってsite_rubyにインストール

プトを置いたディレクトリに、展開したerbのパッケージに含まれる、lib/erbというディレクトリを丸ごとコピーしてしまえば使えます。Rubyが、カレントディレクトリをライブラリの検索パスに持っているのが利用できるわけです（**図1**）。

## 参照機能

それではerbを使って参照機能を作ります。とりあえず最新5日分の日記を逆順に並べることにしましょう。また、Web日記のスタンダードである、段落アンカーもここで埋め込みます。段落アンカーは、以下のようなURLで表現することにしましょう。

```
./?date=年月日#p段落番号
```

「./」はCGIスクリプトindex.rbを指します。続いてCGIのパラメータdateの値として年月日を示します。そして段落番号は「p」に続けた数値で、ページ内のアンカーとして表現しましょう。たとえば2001年7月8日の最初の段落は、「./?date=20010708#p01」になります。

参照機能の実装として、まずはeRubyタグを埋め込んだHTMLファイル（rhtmlファイル）、latest.rhtmlを作ります（**リスト2**）。埋め込まれたコードのほうが多いため読みづらくなっていますが、ベースになっているのはまぎれもなくHTMLです。わかりやすくするために、コード部分を抜き出して注釈を加えた**リスト3**を見てください。

最新5日分の日記を取り出すために、イテレータlatestを使っています。そのブロック中では、順に取り出したDiaryクラスのインスタンスdiaryに対して、日付とタイトルを表示したあと、前回定義した段落を順に取り出すイテレータ、each_paragraphを呼び出しています。そしてeach_paragraphのブロック内では、サブタイトルがあるかどうかで条件分岐をして、アンカーの付け方などに違いを持たせています。

以上、2つのメソッドをイテレータとして使っていますが、latestメソッドのほうはまだ定義していません。先にrhtmlファイルを作って、これから作るメソッドの要求を定めたということになります。あとはこれに従ってコードを書けばいいわけです。またlatestメソッドが定義されるべきクラスは、複数日付にわたる日記データを管理している必要がありますから、TDiaryクラス（もしくはその派生クラス）に実装されるべきであることがわかります。

そこで、続いてはこのlatestメソッドを実装します。前回、更新用のクラスTDiaryAppendをTDiaryの派生クラスとしたように、今回も参照用のクラスとしてTDiaryViewを定義しました（**リスト4**）。initializeメソッドでは、TDiary#transactionに現在時刻を与えて呼び出すことで、最新のデータを読み込んでいます。これで日記データは@diariesに読み込まれているはずです。

そこでイテレータであるlatestメソッドでは、@diariesのキー（日付を文字列化したもの）を逆順にソートし、与えられたlimitを超えない範囲で数え上げています。「最新のn日分の日記」を取り出す処理がこれだけの行数で、しかもわかりやすく書けるのは、Rubyの記述力の高さがあってこそですね。

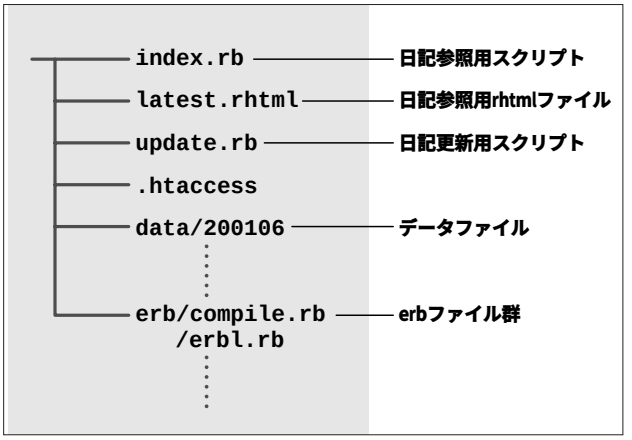



図1　tDiaryのディレクトリ構造

あとは実行するだけ、と言いたいところですが、肝心な処理がまだです。それは、rhtmlファイルを読み込んで、erbに解釈させる処理です。これを実現するために、erb付属のERbLightクラスを使いましょう。ERbLightクラスは、値の出力を<%= 〜%>のみに限定したクラスです。通常のeRubyでは<%〜%>の中でもprint関数などの標準出力への出力を呼び出すことでも出力を生成できますが、ERbLightではこれができません。しかし実用上問題がないうえに軽量なので、今回はこちらを使います。

ERbLightクラスは、以下のように使います。

```
require 'erb/erbl'
```

▼リスト2　最新5日分の日記を表示するlatest.rhtml

```
<html>
<head>
  <title>tDiary</title>
</head>
<body>
<h1>tDiary</h1>
<%
latest( 5 ) do |diary|
%>
  <h2>
    <a href="./?date=<%= diary.date.strftime( '%Y%m%d' ) %>">
      <%= diary.date.strftime( '%Y-%m-%d' ) %>
    </a>
    <%= diary.title %>
  </h2>
<%idx = 0
  diary.each_paragraph do |paragraph|
    if paragraph.subtitle then %>
      <p>
      <a name="p<%= '%02d' % idx += 1 %>"
      href="./?date=<%= diary.date.strftime( '%Y%m%d' ) %>#p<%= '%02d' % idx %>">■</a>
      <%= paragraph.subtitle %>
      </p>
      <p>
<%  else%>
      <p>
      <a name="p<%= '%02d' % idx += 1 %>"
      href="./?date=<%= diary.date.strftime( '%Y%m%d' ) %>#p<%= '%02d' % idx %>">■</a>
<%  end%>
    <%= paragraph.body.collect{|l|l.chomp}.join( "</p>\n<p>" ) %>
    </p>
<%end%>
  <hr>
<%
end
%>
</body>
</html>
```

▼リスト3　latest.rhtmlのコード部分を抜き出したもの

```
latest( 5 ) do |diary| #最新5日分の日記を順に取り出す
  #日記の日付、タイトルを表示

  diary.each_paragraph do |paragraph| #日記中の段落を順に取り出す
    if paragraph.subtitle then #サブタイトルがある段落
      #サブタイトルにアンカーを付けて表示
    else #サブタイトルがない段落
      #最初の段落用のアンカーを生成
    end

    #すべての段落の本文を表示
end
```

```
erbl = ERbLight::new( eRubyスクリプト )
html = erbl.result( binding )
```

まず、erb/erblをrequireします。これでERbLightクラスが使えるようになります。ファイルなどから読み込んだeRubyスクリプトを与えて、ERbLightインスタンスを作ります。続いて、bindingを与えてresultメソッドを呼ぶことで、eRubyタグの中身が解釈された文字列が得られます。このbindingというのは、現在のインスタンスの状態を保持したオブジェクトで、binding関数を呼び出すと得られます。詳しい説明は省きますが、このことで、eRubyスクリプト中で使っているメソッドやインスタンス変数は、すべて呼び出し側のインスタンスのものが使えることになるのです。

それではこれを、TDiaryViewの基底クラス、TDiaryに実装しましょう（**リスト5**）。File::readlinesを使って与えられたファイル名fileの中身をすべて読み込み、ERbLightインスタンスを生成しています。現在のbindingを与えて得られたHTMLを、文字列として返します。

最後に、このeval_rhtmlを呼び出すCGIスクリプトindex.rbを書けば、参照機能は完成です。'cgi'および'erb/erbl'は、tdiary.rbの中でrequireされていることにしました（**リスト6**）。前回作成した更新用CGIスクリプトに比べて、ずいぶんすっきりしましたね。eRubyの威力といえましょう。また、今回は省略しましたが、もしエラーが発生しても、エラーメッセージ用のrhtmlファイルを用意しておけば、結果を返すのも簡単になります。

参照機能にはこのほかに、段落アンカーをクリックされた場合の対処が必要です。CGIにはdateパラメータとして日付が指定されますから、それに応じて単独の日付の日記を表示するような仕組みも必要です。この機能は、宿題ということにしましょう。最新表示から、ループを1つ減らすだけですから、簡単ですね。

## アンテナ対策

以上でいちおう日記として必須の機能はできたわけですが、一歩進んで「日記コミュニケーション」をサポートしようと思った場合、欠かせない重要な機能があります。アンテナ対策です。

アンテナというのは、指定したWebページの更新状況を定期的に調べてくれるソフトウェアです。自分が欠かさず読んでいる日記がいつ更新されたのかを、タイムリーに知ることができてとても便利です。アンテナはWeb上の各地に設置されていますし、もちろん自分で運用することもできます。Rubyで書かれた「たまてばこ」というアンテナもあります（http://www.wakaba.toyonaka.osaka.jp/~ikemo/soft/tama/）し、そもそもRubyホー

▼リスト4　TDiaryViewクラス

```
class TDiaryView < TDiary
  def initialize( cgi )
    super
    transaction( Time::now ) {}
  end

protected
  def latest( limit )
    @diaries.keys.sort.reverse.each_with_index do |date,idx|
      break if idx >= limit
      yield @diaries[date]
    end
  end
end
```

▼リスト5　TDiary#eval_rhtmlメソッド

```
class TDiary
  :
  def eval_rhtml( file )
    rhtml = File::readlines( file ).join
    ERbLight::new( rhtml ).result( binding )
  end
  :
end
```

▼リスト6　参照機能を実装したindex.rb

```
#!/usr/bin/ruby -Ke

require 'tdiary'

@cgi = CGI::new
tdiary = TDiaryView::new( @cgi )

print @cgi.header( 'type' => 'text/html', 'charset' => 'EUC-JP' )
print tdiary.eval_rhtml( 'latest.rhtml' )
```

ムページにあるHotLinksがアンテナそのものです(http://www.ruby-lang.org/ja/hotlinks.html)。

このアンテナでtDiaryを捕捉してもらうためには、CGIが「Last-Modified」というHTTPヘッダを返す必要があります。これは、そのページがいつ更新されたのかを示す情報です。これがないとアンテナは更新時刻を取得できず、その日記はいつまでたっても更新されていないことになってしまいます。Last-Modified対応は、Web日記コミュニケーションには必須の機能といえるでしょう。

日記に限らず、CGIで動的にページを生成する場合にページの更新状況を知らしめるのは良いことです。それでは、Last-Modifiedを返す機能を、tDiaryに組み込みましょう。

まずは、日記が更新されたときにその時刻を保存しなければなりません。更新時刻は日付ごとに持つべきですから、Diaryクラスがその値を保持すべきでしょう。そこで、Diaryクラスのappendメソッドを改造します(**リスト7**)。日記の内容が変更されるappendメソッドで、@last_modifiedというインスタンス変数に現在時刻を設定しています。また、attr_readerでその値を外部から取り出せるようにしました。

続いてTDiaryViewに、表示するすべての日記のうち、最新のものを取り出すメソッドlast_modifiedを作成します(**リスト8**)。latestメソッドを利用して最新日記limit日分を取り出し、最新のlast_modifiedを計算しています。

最後にindex.rbのHTTPヘッダ出力部分を書き換えて完了です(**リスト9**)。ここで使っているのが、CGI::rfc1123_dateメソッドです。CGIクラスのメソッドは、与えられたTimeオブジェクトからRFC1123で定められた時刻表記に変換してくれます。これで正しいLast-Modifiedヘッダを作成することが可能です(ただし、Ruby 1.6.2以

**▼リスト7　更新時刻を保存するDiary#append**

```
class Diary
     ⋮
  def append( body )
    body.gsub( "\r", '' ).split( /\n\n+/ ).each do |fragment|
      paragraph = Paragraph::new( fragment )
      @paragraphs << paragraph if paragraph
    end
    @last_modified = Time::now #現在時刻を保存
    self
  end

  attr_reader :last_modified
     ⋮
end
```

**▼リスト8　更新時刻を返すTDiaryView#last_modified**

```
class TDiaryView < TDiary
     ⋮
  def last_modified( limit )
    last_modified = Time::at( 0 )
    latest( limit ) do |diary|
      if diary.last_modified and diary.last_modified > last_modified
        last_modified = diary.last_modified
      end
    end
    last_modified
  end
     ⋮
end
```

**▼リスト9　Last-Modified対応index.rb(一部)**

```
print @cgi.header(
  'type' => 'text/html',
  'charset' => 'EUC-JP',
  'Last-Modified' => CGI::rfc1123_date( tdiary.last_modified( 5 ) )
)
print tdiary.eval_rhtml( 'latest.rhtml' ) if /HEAD/i !~ @cgi.request_method
```

前のrfc1123_dateメソッドにはバグがあって正しい表記にならないので、1.6.3以降が必要です)。

また、この新しいindex.rbではボディ部分の出力を、CGI#request_methodが「HEAD」以外の時のみに絞っています。これは、アンテナやWebブラウザが更新時刻を取得するときに、まずHTTPのHEADコマンドを使うためです。HEADコマンドへの応答としてボディまで返してしまうので無駄なのでこのように抑制し、帯域を節約しています。

## Refererとツッコミ

さて、段落アンカーとアンテナ対策をすれば、Web日記としての体裁はおおむね整ったといえます。しかしこれだけでは、日記コミュニケーションはアンテナ経由で人から読まれるだけの一方通行しか成立しません。というのは、日記間のツッコミ合いはリンクで行われるので、自分の日記がどこからリンクされたのかを感知できないとコミュニケーションが成立しないからです。

HTTPにはこの「どこからリンクされたか」をある程度知るための手掛かりである、Refererというヘッダがあります。あるページにあなたの日記へのリンクがあって、誰かがそのリンクをたどってやってくると、リンク元ページのURLがこのRefererに設定されるのです。通常、この情報はHTTPサーバのログを解析するなどして得るものですが、CGI内部で取得することも可能です。RubyではCGI#refererメソッドを使って簡単に得ることができます。tDiaryではこのReferer情報を取得して、今日の日記がどこからどれだけ参照されているかを直接見られるようにしてしまいましょう。

さらにもうひとつ、付け加えたい機能があります。日記コミュニケーションは、もっぱら日記をつけている人どうしの間でのみ成立します。そうでない人、つまり自分では日記をつけずにもっぱら読む専門の人は、この輪の中に入れません。そういう人たちを救済するために、ツッコミ専用の簡易掲示板を付けます。そう、tDiaryの「t」には「ツッコミ」の意味が込められていたのです(どうしょうもないですね)。

が、紙面も足らなくなってきたので、ここではReferer対応の説明だけしましょう。まず、その日のRefererを保存するためにDiaryクラスを拡張し、add_refererとeach_refererというメソッドを追加します(**リスト10**)。

インスタンス変数@refererはHashです。キーにはRefererのURLを使い、値には参照回数とRefererのURLを要素に持つArrayを入れます。同一のRefererについて、アクセス数をカウントできるようになっています。add_refererの中で、キーになるURLにはCGI::unescapeを施している点に注目してください。同じページからやってきても、ブラウザの違いなどの要因によって、字面上は異なるURLになることがあります。「~」が「%7d」になっているような場合です。CGI::unescapeを通しておくことで、この違いを正規化することができます。Webサイトを訪れるユーザーの環境は多様ですから、それらの違いはできるだけ吸収してやる必要があります。

似たような注意として、ツッコミ掲示板を実装する時に投稿の文字コードを統一したり、HTMLタグを無効にす

**▼リスト10　Referer対応Diaryクラス(一部)**

```
class Diary
   ⋮
  def add_referer( ref )
    @referer = {} if not @referer
    uref = CGI::unescape( ref )
    if a = @referer[uref] then
      @referer[uref] = [a[0]+1, a[1]] #参照回数を増加
    else
      @referer[uref] = [1,ref]
    end
  end

  def each_referer( limit )
    @referer = {} if not @referer
    @referer.collect{|k,v| v}.sort.reverse.each_with_index do |ary,idx|
      break if idx >= limit
      yield ary
    end
  end
   ⋮
end
```

ることも大切です。ブラウザによってはこちらの意図しない文字コードを送り込んでくることがありますから、フォームからの入力はすべて、以下のような感じで文字コードを統一しておくべきです。ここではNKF拡張モジュールを使って文字コードをEUCに変換しています。

```
require 'nkf'
euc_string = NKF::nkf( '-e', @cgi['text'][0] )
```

また、不特定多数から入力される可能性があるデータからは、HTMLタグを無効にしておくのも大切です。自分の日記に不愉快な写真を貼られたり、勝手に文字属性を変えられたりされたらたまりません。これには、CGI::escapeHTMLが使えます。このメソッドは、「<」を「<」に置換するなど、HTMLの特殊文字をエスケープしてくれるので、タグを無効化する効果があります。

```
escaped = CGI::escapeHTML( @cgi['text'][0] )
```

脱線してしまいましたが、Referer対応に戻りましょう。each_refererでは、参照回数の多い順に数え上げるために少しややこしいことをしていますが、あとは普通のイテレータです。パラメータlimitで、数え上げる数を制限できます。

さらにTdiaryViewクラスには、最新の日記にRefererを追加する処理を加えます（**リスト11**）。この処理はファイルに保存されなければいけませんから、transactionの実行中に行わなければいけません。そこで、initializeメソッドにその処理を加えています。latestメソッドのパラメータに1を与えると最新の日記が1つだけ取得できるのを利用しています。

これでRefererの保存ができたので、あとは利用するだけです。日記を表示するrhtmlファイルに、以下のような文を挿入すれば、参照回数が多い順にRefererへのリンクが並ぶことになります（**画面3**）。

```
<p>本日のReferer:
<% diary.each_referer( 10 ) do |i,url| %>
<a href="<%= url %>"><%= i %></a> |
<% end %></p>
```

**▼リスト11　Referer対応TDiaryViewクラス(一部)**

```
class TDiaryView < TDiary
  def initialize( cgi )
    super
    transaction( Time::now ) do
      if @cgi.referer then
        diary = nil
        latest( 1 ) {|d| diary = d } #最新の日記を取り出す
        diary.add_referer( @cgi.referer ) if diary
        $stderr.puts "initialize: #{diary}"
      end
    end
  end
  ⋮
end
```

## おわりに

最後は少し駆け足になってしまいましたが、RubyでCGIスクリプトを書く雰囲気はつかんでもらえたでしょうか？　これまでの説明を読めばわかるように、Rubyでは機能を追加しても影響範囲が少なく、少しずつプログラムを書き進めていくのがとても楽です。オブジェクト指向スクリプト言語ならではの部分ではないかと思います。アイデアを少しずつ形にするのが好きな筆者は、おかげで助かっています:-)

今回はtDiaryのエッセンスを解説しましたが、tDiary自体はフリーソフトウェアとして公開しています。柔軟なカスタマイズやiモード対応など、今回は説明しなかった機能も加わっています。なお、本誌付録CD-ROMのDisc 2に、tDiary 0.9.9（tdiary-0.9.9.tar.gz）を収録しています。ぜひ使ってみてください。

```
http://www.spc.gr.jp/tdiary/
```


**画面3　完成したtDiary**

# Perlスクリプト入門
## すぐにできる実践Perl

**手のひらに乗るLinuxマシンをラジコンカーにのせたら……。ネットワークも無線があるし。そうなると、Perlで簡単に制御できてしまうのではないでしょうか。ついにPerlもロボット制御言語になるのかと発想（妄想？）は広がるばかりです。**


## 第6回　ファイル入出力と外部コマンド

文：おもてじゅんいち／かざぐるま
*Text : Junich Omote/Kazaguruma*

本連載の第2回で、ファイルハンドルとは「ファイルを指定するための名前」と説明しました。正確には、「ファイル、デバイス、ソケット、パイプにつけた名前のことで、そのうちどれを表すか指定するためのものである」ということになります。つまり、ファイルハンドルとはいっても、それが指し示すものはファイルだけとは限らないのです。ちょっとしたツール代わりのスクリプトや、Webで使われるCGIなどでも、ファイルのほかにパイプなどを使うことがあります（CGIについては次回以降に解説します）。

つまり、入出力を行う際に入出力先がファイルであるか他デバイスであるかなどをまったく意識することなく、すべてファイルハンドルとして扱うことができる仕組みになっているのです。

ファイルハンドル名についての注意点は、英数字であれば変数名や関数名と同じでもかまいませんが、混同しないように、できるだけ別の名前をつけておくほうがよく、Perlの予約語などと間違えないように、すべて大文字を使用するように推奨されています。

### ■ファイルのオープンとクローズ

ファイルを使用するときは必ずファイルをオープンする必要があります。ファイルオープンにはいくつかのモードがあり、たとえばファイル「test.txt」をオープンする場合、**表1**のように、ファイル名の前の記号によって読み／書きや上書き／追加書き込みのモードが決まります。入出力共用というモードは一見便利そうですが、予想しなかった書き込みが行われたり、スクリプトの中でのそのファイルの用途が不明確になったりしますので、どうしても必要でない限りは、読み込み専用か上書き／追加書き込みモードとしてオープンするようにしましょう。

また、ファイルをオープンしたら必ずクローズするのを忘れないようにしましょう。

```
close FH;
```

### ■STDIN、STDOUT、STDERR

Perlには特別なファイルハンドルとして、STDIN、STDOUT、STDERRというものが用意されています。

STDINは標準入力というもので、パイプやリダイレクトのときにデータを受け取る入口で、CGIのPOSTメソッドでも標準入力にデータが送られてきます。

STDOUTは標準出力で、同じくパイプやリダイレクトでのデータの出口です。CGIのHTMLデータも標準出力

| 指定方法 | 意味 |
|---|---|
| open FH,"test.txt";<br>open FH,"<test.txt"; | 読み込み専用 |
| open FH,">test.txt";" | 上書き出力 |
| open FH,">>test.txt";" | 追加書き込み出力 |
| open FH,"+<test.txt";" | 入出力共用 |
| open FH,"+>test.txt";" | 入出力共用（上書き） |

表1　ファイルオープンのモード（FHは任意のファイルハンドル）

から出力します。コマンドラインで実行したときのデフォルトはコンソール（画面）につながっています。

STDERRは標準エラー出力と呼ばれるもので、標準出力に似ていますが、パイプやリダイレクトを行ってもコンソールに出力されます。CGIの場合は通常、ログに出力されます。

STDIN、STDOUT、STDERRはすでにファイルハンドルとしてオープンされていますから、STDINから1行のデータを読み込むには、

```
$line = <STDIN>;
```

とすればよく、標準出力にデータを出力するには、

```
print STDOUT "……";
```

でよいのですが、標準というだけあって、print文でファイルハンドルが省略されている場合は、標準出力に出力するという決まりがあるので、これは、

```
print "……";
```

でよいわけです。今まで出てきた普通のprint文は、標準出力に出力するということだったのですね。

同じく標準入力も同様に省略できますので、先ほどの「$line = <STDIN>;」は、

```
$line = <>;
```

でもかまいません。<>の1行入力でファイルハンドルが省略されたときは、標準入力からのデータを受け取ります。

### ■die関数

ファイルをオープンしようとして失敗することはよくあります。指定したファイル名が間違っている、ファイルが存在しない、アクセス権限がないなど……。

気をつけなければいけないのは、そのような原因によってファイルのオープンに失敗しても、スクリプトはエラーにならず、処理が続行されるということです。普通はファイルがオープンできたことを前提に、その後のスクリプトが書かれているはずですから、オープンの失敗は重大な問題になりかねません。

Perlでは、ファイルのオープンに失敗したときに処理を中断させたい場合はdie関数を使うことが通例になっています。スクリプトは、

```
open FH,"test.txt" or die;
```

というものです。

openとdieのあいだにあるorは||と同じで、「どちらかが真であれば真を返す」という意味ですが、このとき、前半が真だとわかった時点で後半の評価をせずに次に進みます。後半はあらためて真か偽かを評価しなくても、全体は真だと決まっているわけですから、効率から考えて後半部分の処理をスキップするのです。つまりこのような場合、前半が偽であった場合にのみ後半部分を処理するのです。

open関数はファイルのオープンに成功すると真を、失敗すると偽を返してきますので、オープン成功のときは後半のdieは実行されません。オープンに失敗したときだけdieが実行されます。

dieはその名の通り、スクリプトを強制的に終了する命令で、終了時にメッセージを出力することもできます。

```
open FH,"test.txt" or die "オープン失敗¥n";
```

| 演算子 | 意味 |
|---|---|
| -e | ファイルやディレクトリが存在する |
| -r | ファイルやディレクトリが読み出し可 |
| -w | ファイルやディレクトリが書き込み可 |
| -x | ファイルやディレクトリが実行可 |
| -o | ファイルやディレクトリの所有者である |
| -R | ファイルやディレクトリが実UID／GIDで読み出し可 |
| -W | ファイルやディレクトリが実UID／GIDで書き込み可 |
| -X | ファイルやディレクトリが実UID／GIDで実行可 |
| -O | ファイルやディレクトリが実UIDの所有である |
| -z | ファイルの大きさがゼロ |
| -s | ファイルやディレクトリの大きさを返す |
| -f | ファイルは通常ファイル |
| -d | ファイルはディレクトリ |
| -l | ファイルはシンボリックリンク |
| -p | ファイルは名前付きパイプ |
| -S | ファイルはソケット |
| -b | ファイルはブロック特殊ファイル（ディスク等） |
| -c | ファイルはキャラクタ特殊ファイル |
| -t | ファイルハンドルはttyにオープンされている |
| -u | setuidビットがセットされている |
| -g | setgidビットがセットされている |
| -k | stickyビットがセットされている |
| -T | ファイルはテキストファイル |
| -B | ファイルはバイナリファイル（-Tの反対） |
| -M | スクリプト実行開始時のファイル修正時からの日数 |
| -A | 同様にアクセス時からの日数 |
| -C | 同様にinode変更時からの日数 |

表2　ファイルテスト演算子

dieのメッセージは標準エラー出力に出力されます。

orの働きは前述のとおりですが、英語を読むように、「ファイルをオープンしろ、さもなければ（or）強制終了しろ」と理解すればよいでしょう。

ただし、CGIでは標準エラー出力は通常、ログに記録されるだけなので、ブラウザからは状況がわかりませんし、スクリプトが何のメッセージもなく強制終了しても困りますから、dieはあまり使われません。

### ■ファイルテスト演算子

ファイルオープンの失敗に対するもうひとつの対策として、事前にファイルの存在を確かめておくという方法があります。事前にファイルの存在を確認しておけば、ファイルのオープンに失敗することがなくなるというわけです。Perlにはファイルをオープンせずに、そのファイルについての状態を調べる演算子があります。これをファイルテスト演算子と呼びます（**表2**）。使い方は、演算子の後にファイル（ディレクトリ）名かファイルハンドルを書きます。

たとえば、

```
-e "test.txt"
```

とすれば、この式はファイル「test.txt」が存在しているかどうかを返しますので、

```
if ( -e "test.txt" ) { ……
```

のように、if文などの条件式として使えます。

ファイル名は文字列であればよいので、変数にファイル名を格納して、

```
-e $filename
```

などと書くこともできます。

ここで、デフォルト変数$_を思い出してください。そうです、ファイル名部分を省略した場合は、$_の内容をファイル名として処理します。

### ■ファイルの削除

ファイルを削除するときは、unlinkを使います。

たとえば、

```
unlink "test.txt";
```

とすれば、「test.txt」というファイルが削除されます。

unlinkのあとにはリストを書くことができるので、

```
unlink "test1.txt","test2.txt";
```

と複数のファイル名を並べて書いたり、

```
@files = ("test1.txt","test2.txt");
unlink @files;
```

というように、配列にファイル名を格納して、それらを一度に削除することができます。

unlinkは実行結果として削除できたファイル数を返しますので、ひとつも削除できなかったときは0が返ることになり、偽になります。前述のorを使えば、

```
unlink "test.txt" or print "削除できませんでした。";
```

という使い方もできます。

### ■ファイル名の変更

ファイル名を変更するときは、renameを使います。renameの書式は、

```
rename 旧ファイル名，新ファイル名
```

になります。たとえば、「test.txt」を「finish.txt」に変更するには、

```
rename "test.txt","finish.txt";
```

となります。

renameはLinuxのmvコマンドと同じように、ファイル名を変えるのではなく、パス名を変更することでファイルを移動させることもできます。/tmp/にあった「test.txt」を、/var/www/html/に移動するときは、

```
rename "/tmp/test.txt","/var/www/html/test.txt";
```

とします。ただし、お互いのディレクトリが別のファイルシステム（パーティション）にある場合はエラーになります。その場合は、コピーと削除を組み合わせたスクリプトを書く必要があります。

■アクセス権（パーミッション）の変更

本誌の読者の方々は、Linuxのファイルアクセス権についてご存じだとは思いますが、簡単に説明しておきます。

ファイルには、1つ1つにオーナー（所有者）とアクセス権を設定することができます。アクセス権には、「読み込み可能」「書き込み可能」「実行可能」の3種類があり、そのファイルを使うユーザーの種類によって、異なったアクセス権を設定できます。ユーザーの種類には、「所有者」「グループ内」「全員」があります。

つまり、この3つのユーザー種類と、3つのアクセス権を自由に組み合わせることができるのです。たとえば、「このファイルは、所有者は読み書きしてもよいが、グループ内は読むだけ、そのほかの人は読むこともできない」などといった制約を、ファイルごとに設定することができるのです。

ls -lとしてファイルの属性を表示させてみると、

```
rwxrwxrwx   jun   users   1024     test.txt
```

のように表示されます。最初の「rwxrwxrwx」がアクセス権を表し、順に所有者名、グループ名、サイズ、ファイル名です。アクセス権は、「rwx」が1つのかたまりで、左から、「所有者」「グループ内」「全員」のアクセス権を表します。「rwx」の「r」が「読み込み可能」、「w」が「書き込み可能」、「x」が「実行可能」を表します。

これらの「rxw」をいろいろと組み合わせてやれば、さまざまなアクセス権が設定できます。

具体的には、テキストファイルなどの実行しないファイルには、「rw-r--r--」というアクセス権を、Perlスクリプトなど、読み書きだけでなく実行するものに関しては、「rwxr-xr-x」というアクセス権を与えます。

アクセス権の設定は、この「rwx……」を8進数で表したものを使用します。各権限は8進数では次のように表されます。

```
- ＝ 0   x ＝ 1   w ＝ 2   r ＝ 4
```

所有者やユーザーそれぞれのアクセス権はこれらを合算した形となります。たとえば、「rw-r--r--」の場合、所有者に対する部分が「rw-」ですから、4＋2＋0＝6、グループ内、全員に対してはいずれも「r--」なので4＋0＋0＝4です。つまり「rw-r--r--」の8進数表記は、0644になります（8進数表記は先頭に0をつける）。同様に、「rwxr-xr-x」の場合の8進数表記は0755です。それをファイルのアクセス権として設定するには、

```
chmod 0644,"test.txt";
```

のように行います。

■ファイル所有者の変更

ファイルの所有者の変更も、Linuxのchownコマンドと同じように行えますが、ユーザー名を使って設定することはできないため、ユーザーIDが必要になります。また、chownという関数で、所有者とグループを同時に設定できます。なお、chgrpという関数はありません。

今、ユーザーjunのユーザーIDが100、グループIDが50として、ファイル「test.txt」の所有者とグループをjunにする場合は、

```
chown 100,50,"test.txt";
```

となります。もし、所有者の設定だけでよく、グループを変更しないのであれば、グループIDの部分に－1を指定します。

```
chown 100,-1,"test.txt";
```

■複数ファイルに対する処理

Linuxのコマンドには、複数のファイルをまとめて扱う、＊や＊.txtといったワイルドカードがあります。Perlスクリプトの処理でも、このように複数ファイルをまとめて処理したいことがよくあります。Perlでもワイルドカードを使うことができます。たとえば、カレントディレクトリの全ファイルのファイル名を取得するには、

```
@files = <＊>;
```

と書きます。<>は、オープンされたファイルハンドルからデータを1行ずつ読み込むものとして学習してきたのですが、実はこの<>は、1行データ入力の<>とはまったく違う演算子なのです。

この場合、@files配列には、カレントディレクトリの全ファイル名が格納されます。

＜＊＞の<>はファイル名グロブ演算子と呼ばれ、<>内をワイルドカードとみなし、それにマッチするファイル名

をリストとして返します。1行データ入力との違いですが、Perlは、<>内にスペースや「.」、「*」など、ファイルハンドルかスカラー変数名以外のものがあるとファイル名グロブ演算子であると判断します。

たとえば、

```
<*>
<*.html>
<test*>
```

などはファイル名グロブ演算子で、

```
<TEST>
<test>
<$filename>
```

などは、1行データ入力になります。

このときに困るのが、ワイルドカード表記の"*.txt"などを変数に格納している場合です。

```
$wild = "*.txt";
```

のときに、ファイル名グロブ演算子を使おうとして、

```
<$wild>
```

とすることはできません。1行データ入力と解釈されてしまいます。ファイル名グロブ演算子として使いたい場合は、

```
<${wild}>
```

と、変数名の部分を{}でくくります。

非常にややこしいため、Perlバージョン5ではファイル名グロブ演算子の代わりに、glob関数が使えるようになりました。たとえば、<*.txt>は、

```
@files = glob("*.txt");
```

と書くことができます。ワイルドカードが変数に代入されている場合でも、

```
@files = glob($wild);
```

とすればよいわけです。混乱を防いで見やすいスクリプトにするためにも、glob()を使うことをお勧めします。Perlバージョン4の場合は残念ながら、<>を使う方法しかありません。

## ディレクトリの操作

処理したいのはカレントディレクトリばかりではありません。あるディレクトリを決めて、その中にあるファイルを処理したい場合もあるでしょう。単にワイルドカードでそれらのファイル群を指定できるのなら、前述のファイル名グロブ演算子や、glob()でもかまわないのですが、もう少し細かくファイルを選別したり、ディレクトリに含まれるファイルを順次処理していく方法として、ディレクトリをオープンすることができます。

ディレクトリをオープンする場合も、ファイルと同じくディレクトリハンドルを使用します。ディレクトリハンドルの命名ルールはファイルハンドルと同じですが、ディレクトリハンドルは読み込み専用になります。

ディレクトリハンドルのオープンは、

```
opendir ディレクトリハンドル, ディレクトリ名
```

です。/tmpディレクトリをオープンする場合は、

```
opendir TMP,"/tmp";
```

となります。処理が終わったら、closedir関数でクローズします。

ディレクトリハンドルをオープンしたら、readdirでそのディレクトリ内のファイル名を読み出します。readdirはスカラーコンテキストでは順に1つずつファイル名を返します。

```
opendir TMP,"/tmp";
while( $file = readdir TMP ) {
    print $file;
}
closedir TMP;
```

リストコンテキストではすべてのファイル名が一度に読み込まれます。

```
opendir TMP,"/tmp";
@files = readdir TMP;
foreach $f (@files) {
    print $f;
}
closedir TMP;
```

いずれの場合も、カレントディレクトリを表す「.」と上位ディレクトリを表す「..」も返されてきます。<＊>を使う場合には返されてきません。

■ディレクトリの作成

ディレクトリの作成は、Linuxのコマンドと同じくmkdirを使います。ただし、ディレクトリ名だけでなく、そのディレクトリのアクセス権も指定しなければなりません。アクセス権はchmodの時と同様に8進数表記で指定します。

```
mkdir "mydir",0777;
```

mkdirがディレクトリの作成に失敗したときは偽を返します。

■カレントディレクトリの変更

カレントディレクトリを変更する場合はchdirを使います。Linuxコマンドのcdにあたります。

```
chdir "/usr/local/bin";
```

■ディレクトリの削除

ディレクトリを削除する場合はrmdirを使います。その場合はディレクトリが空でなければなりません。

```
rmdir "mydir";
```

ディレクトリの削除に成功したときは真を、失敗したときは偽を返します。

■暗号化

実際に使うことがあるかどうかわかりませんが、Perlではいろいろなデータをファイルなどに書き込んでおくとき、簡単に暗号化することができます。ちょっとおもしろい機能なので紹介しておきます。最近はネットワーク利用が当たり前になってきたので、ちょっとしたセキュリティ対策として役に立つかもしれません。

cryptは、文字列データを暗号化するものです。ただしこれは元々Linuxのパスワードファイルを暗号化するためのものなので、暗号化される前の文字列は8文字までが有効で、それ以上の文字列を暗号化しても結果の暗号は同じものになってしまいます。

また、cryptは暗号化するだけで、暗号を元の文字列に複合する関数は用意されていません。つまり、使い方としては、合っているかどうかを調査したい文字列も暗号化して、元の暗号と一致するかどうかを調べることになります。

たとえば、パスワードなど元になる文字列をあらかじめ暗号化しておいて、変数$cryptedに格納しておき、新たに入力された文字列を$entryに入れて、

```
$ent_crypt = crypt $entry,"ky";
if ( $ent_crypt eq $crypted ) {
    ……                #暗号一致
}
```

というように使います。cryptは、

```
crypt 元の文字列, キー
```

という書式で使います。キーは2文字の英数字です。ちなみに、

```
$res = crypt "ABCDEFGH","ky";
print $res;
```

と実行してみると、筆者のシステムでは、

```
kyEdarpFN3iS6
```

という暗号が表示されました。先頭の2文字はキーがそのまま出力されるようです。解読（正確には再暗号化による一致検査）をする側でもキーは必要になりますから、キー部分は暗号化されずに付加されているのですね。一度試してみてください。

## 外部コマンドの実行

Linuxにはさまざまなコマンドやツールプログラムがあ

ります。ファイルを圧縮したり、システムを設定したり、メールを送ったりなどなど……。Perlにはスクリプトから外部コマンドを実行する機能が用意されています。せっかくですから、これらのコマンド機能をPerlで制御したいものです。

### ■system()

system()は外部コマンドを起動します。スクリプトはそのコマンドが終了するまで待っています。たとえば、「ls /etc」というコマンドを実行する場合は、

```
system("ls /etc");
```

と書きます。この場合、Perlはシェルを起動して、シェルに"ls /etc/＊"というコマンド文字列を渡します。

一方、

```
system("ls","/etc");
```

と書いたり、

```
@comm = ("ls","/etc");
system(@comm);
```

のようにコマンド文字列を配列に格納してsystem()の引数にした場合は、Perlはシェルを起動せずに直接lsコマンドを実行してくれます。

シェルを起動する場合としない場合とでは、起動されるプロセスの数が違いますから、シェルを起動しないほうがパフォーマンスは良くなります。

しかし、コマンド文字列に、ワイルドカードの＊やリダイレクトの<、>、パイプの｜などが含まれる場合は、シェルを起動しなければ解釈されません。

たとえば、

```
system("ls /etc/host＊");
```

はシェルが起動されるためワイルドカードが正しく解釈されますが、

```
system("ls","/etc/host＊");
```

は、文字どおり「host＊」という名前のファイルとして解釈されるため、何も表示されないことになります。特にコマンド文字列を配列に格納する場合に気をつけてください。

### ■バッククォート演算子

system()で実行されたコマンドの出力は、Perlスクリプトの標準出力（コンソール画面など）に出力されます。ですから、system()ではコマンドの出力データをスクリプト内で利用することができません。コマンドの出力データをスクリプト内で利用したい場合には、バッククォート演算子（`）を使います。

コマンドをバッククォートで囲むと、コマンドが実行されてその出力が返されます。たとえば、

```
$now = `date`;
```

を実行すると、$nowにはdateコマンドが出力する文字列が格納されます。

バッククォート演算子内ではダブルクォートと同じように変数展開やメタ文字の処理が行われます。たとえば、

```
$dir = "/tmp";
$res = `ls $dir`;
```

とすれば、$dirが展開されて「ls /tmp」というコマンドが実行されます。このとき、$resにはコマンドの結果が格納されます。

バッククォート演算子で囲んだコマンドの実行結果は、スカラーコンテキストではコマンドの出力を1つの文字列にして返します。リストコンテキストでは、コマンドの出力を各行ごとのリストにして返します。ですから、

```
$dir = "/tmp";
@files = `ls $dir`;
```

の場合は、@filesの各要素にはlsの出力が1行ずつ、すなわちファイル名が1つずつ入ります。

### ■コマンドをオープンする

実は外部コマンドもファイルと同じようにオープンすることができます。そうすることにより、コマンドに対して入出力を行うことができます。

コマンドのオープンの実体はパイプです。コマンドの出力を受け取るだけならバッククォート演算子でも可能でし

たが、コマンドにデータを送る場合はオープンする必要があります。

コマンドにデータを送る場合は、

```
open COMM,"|sort";
```

と書くことにより、ファイルハンドルCOMMに出力したデータがsortコマンドの標準入力に送られます。データを送るには、

```
print COMM "……";
```

とすればよいだけです。

逆にコマンドから出力を受け取るためには、

```
open COMM,"ls * |";
```

とすればよいことになります。

ただし、入力と出力を同時に行うことはできません。Linuxのパイプでは可能ですが、

```
open COMM,"|sort|";
```

という使い方はできません。

### ■メール送信スクリプト

それではコマンドのオープンを使って、メールを送信するスクリプトを作ってみましょう。

メールを送信するには、sendmailコマンドを使います。Perlでメールを送信するといっても、結局のところsendmailコマンドに送信先やメールの内容などのデータを渡してメールを送信してもらうだけですから、まずはsendmailコマンドでメールを送る方法を調べます。こういうスクリプトがうまく動かない原因の多くは、コマンドの使い方が間違っている場合ですから、必ず手動でコマンドを実行してから、その手順をスクリプトで再現するようにします。

sendmailコマンドでメールを送信する場合は、まず宛先を指定して

```
/usr/sbin/sendmail toaddr@domain.co.jp
```

とタイプし、次にSubjectとFromを入力します。

```
Subject: Mail Test
From: from@domain.co.jp
```

次に空行を1行入れて、メールの本体を入力します。最後に、sendmailコマンドに入力完了を知らせるため「.」を入力します。

```
This is a test mail.
送信テスト。
これはテストのメールです。
.
```

これでsendmailは直ちにメールの送信を行ってくれます。

スクリプトを書く前にぜひ、手動でのメール送信を行ってみてください。ただし、sendmailがインストールされていて（PostfixでもOKです）、SMTPサーバが動いている必要があります。

手動でのメール送信が成功したら、その手順をスクリプトにするだけです（**リスト**）。

ファイルハンドルをMAILとして、sendmailへのパイプをオープンしています。sendmailコマンドの実行パスは環境によって変わる可能性があるため、$mailpathとしていったん変数に代入します。宛先のメールアドレスも同様です。

「Subject」と「From」は最低限ないと、相手に到着したときにタイトルや送信元のない失礼なメールになってしまいます。ほかに、「Reply-To」なども指定できますから、詳しくはsendmailのマニュアルも参照してください。後半2行のprint文はメッセージ本体です。この部分は自由に変数を使うなり、ヒアドキュメントを使うなりしてください。

sendmailに入力の完了を伝える方法ですが、手動のときは「.」だけの行を入力して完了を知らせました。スクリプトでオープンしているときはその必要がなく、ファイルハンドルをクローズすれば、sendmailは入力完了だということがわかり、メールを送信してくれます。

**▼リスト　メール送信スクリプト**

```
$mailpath = '/usr/sbin/sendmail';
$toaddr = 'dareka@domain.co.jp';
open MAIL,"| $mailpath $toaddr";
    print MAIL "Subject: Mail Test \n";
    print MAIL "From: from@domain.co.jp \n\n";
    print MAIL "This is a test mail\n";
    print MAIL "from Perl script !! \n";
close MAIL;
```

# [超]入門シェルスクリプト


**bashのシェルスクリプトについて学ぶ本連載。今回は、スクリプト実行中のキーボード入力の扱い方や、複数の選択肢の中からユーザーに1つだけ選んでもらう処理など、対話的なシェルスクリプトを作成する方法について説明し、終了させるプロセスを一覧から番号で選択できるスクリプトを作成する。**

## 第10回　対話的な処理を行うスクリプト

文：大池浩一
*Text:Koichi Oike*

Linuxを含むUNIX系OSでコマンドやスクリプトの動作を変えるには、起動時のコマンドラインでオプションを指定する方法が一般的だ。実行時にいちいちユーザーの入力を求める対話的な方法では、処理を自動化（無人化）することが難しいからだ。

しかし、起動時のオプション指定によっては、対話的な処理を行うコマンドも存在する。典型的な例がcpやmv、rmなどで使われる-iオプションだ。-iオプションを付けてcpやmvを実行すると、コピー／移動先に同名のファイルがすでに存在する場合に、上書きしてもよいかユーザーに確認を求めてくる。

同様に、rmも-iオプションを付けることで、コマンドラインで指定したファイルひとつひとつに対して、削除してもよいか確認を求めてくるようになる。コマンド実行中にユーザーが入力した内容（「yes」か「no」か）によってその後の処理が変化するわけだ。

こうした対話的なオプションは、処理した結果が大きな影響を及ぼす場合に使われることが多い。

たとえば、Red Hat系ディストリビューションのスーパーユーザー（root）は、cp、mv、rmコマンドが常に-iオプション付きで実行されるようなエイリアス（別名）が標準で設定されている。不用意な操作で設定ファイルなどの大切なファイルが上書きされたり、削除されることを防いでいるわけだ。もちろん、自動処理の妨げになることがないように、問答無用で上書きや削除を行うオプション（-f）も用意されている。

### シェル関数の定義と利用法

シェルスクリプトで対話的な処理を実現するには、キーボードからの入力をシェル変数に格納する組み込みコマンドreadを利用し、入力された内容に応じてif／case制御構造による分岐処理を行えばいい。

さらに、「yes」か「no」かといった二者択一ではなく、3つ以上の選択肢の中から1つだけユーザーに選ばせるような処理も行える。当然のことながら、スクリプトは二者択一の場合よりも複雑になり、for／while制御構造による繰り返し処理や、前回説明したシェル関数、引数をコマンドラインとして評価する組み込みコマンドevalなどを組み合わせる必要がある。

もっとも、bashの場合は一連の処理を代行するselect制御構造が用意されているので、わずか数行の記述で同様の処理を実現可能だ。

以下では、

- **スクリプト中でのキー入力の扱い方**
- **ユーザーに二者択一させる方法**
- **複数の選択肢の中からユーザーに選択させる方法**
- **選択肢の表示や入力処理を代行するselect制御構造**

について説明したのち、終了させるプロセスを一覧から番号で選択できるスクリプトを作成する。

### ●キー入力した内容をシェル変数に設定する

シェルスクリプトでキーボードからの入力を扱うには、組み込みコマンドreadを利用する。標準入力（通常はキーボードからの入力）の内容を1行分読み込んで、シェル変数に格納するコマンドだ。readの書式は次のようになっている。

```
read 変数名…
```

引数には、入力内容を格納するシェル変数名を記述する（$は付かない）。複数のシェル変数をスペースで区切って並べることも可能だ。

キーボードから入力した内容は、組み込み変数IFSの内容（初期値はスペース、タブ、改行）を区切り文字とするワードに分解され、対応する位置のシェル変数に格納される。なお、シェル変数の数がワード数より少ない場合は、残りの内容は最後の変数にまとめて格納される。

たとえば、コマンドラインで、

```
$ read hoge foo bar
This is a test.
```

とすると、シェル変数hogeには「This」、fooには「is」、barには残りの入力内容をまとめた「a test.」がそれぞれ格納される（**図-1**）。

これに対し、

```
$ read hoge
This is a test.
```

のように、変数を1つだけ指定すると、hogeに「This is a test.」がすべて格納されるわけだ（**図-2**）。

ここで注意すべき点は、入力内容の先頭に並んだ区切り文字が自動的に取り除かれることだ。

たとえば、

```
$ read hoge
    This is a test.
```

と先頭にスペースやタブを並べても、hogeにはそれを取り除いた「This is a test.」だけが格納される。


図　組み込みコマンドreadによるキー入力

なお、行頭の空白文字も含めて、入力内容をすべてシェル変数に格納したいなら、readを実行する前に組み込み変数IFSの内容を空文字列に変更すればいい。ただし、IFSの変更はさまざまな部分に影響するので、readを実行後に元の内容（スペース、タブ、改行）に戻しておこう。

要点をまとめると、

- **シェルスクリプト実行中にキーボードから入力するには組み込みコマンドのreadを利用する。**
- **readの引数で指定したシェル変数（複数可）に、ワード分解された入力内容が順番に格納される。**
- **ワード数がシェル変数の数より多い場合は、残りの内容が最後のシェル変数にまとめて格納される。**

となる。

### ●スクリプト実行時に二者択一を行う

シェルスクリプトでユーザーに二者択一させるには、入力を促す「プロンプト」を端末画面に表示し、前節で説明したreadを使ってキーボードからの入力をシェル変数に取り込み、if制御構文などを利用して変数の内容に応じた分岐を行えばいい。

たとえば、ユーザーが「y」を入力した場合に処理を続行するスクリプトは次のようにする。

```
echo -n "処理を続けますか? (y/n) " > /dev/stderr
read ans
if [ "$ans" != y ]; then
  exit
fi
＜yを入力した場合の処理＞
    ⋮
```

まず、echoにより「処理を続けますか? (y/n)」というプロンプトを表示する。このとき、改行を抑制する-nオプションを付けて、readでキー入力する文字がプロンプトの直後に表示されるようにするといい。また、標準エラー出力（/dev/stderr）にリダイレクトすることで、スクリプト実行時のリダイレクトに関係なく、このプロンプトが確実に端末画面に表示される。

続いてreadが実行され、キー入力した内容がシェル変数ansに格納される。組み込み変数IFSを変更していないので、行頭の空白文字は自動的に取り除かれてansには含まれないが、この場合は何の問題もない。

if制御構造による分岐処理では、ansの内容を「y」と比較し、「y」とマッチしない場合は組み込みコマンドのexitを実行してスクリプトを終了する。ここで使われる「"$ans" ！＝ y」という条件式は、ansが「y」以外なら必ずマッチすることに注意されたい。つまり、「n」を入力した場合はもちろん、（「y」を除く）任意の文字列を入力した場合にもスクリプトが終了する。対話的な処理は結果が大きな影響を及ぼすような場合に使われるので、より安全な側に振っているわけだ。

もっとも、上の例だと「Y」「yes」「Yes」「YES」といったキー入力に対してもスクリプトを終了してしまうので、少しばかり都合が悪い。こうした入力に対しても処理を続行するように改良したバージョンは以下のとおりだ。

```
echo -n "処理を続けますか? (y/n) " > /dev/stderr
read ans
case "$ans" in
y*|Y* ) ;;
    * ) exit
esac
＜yやYで始まる文字列を入力した場合の処理＞
                ⋮
```

今度は、if制御構造の代わりにcase制御構造を利用することで、複数の入力文字列に対する処理を1つにまとめている。

まず、シェル変数ansの内容が、

- **yで始まる文字列…「y」や「yes」など**
- **Yで始まる文字列…「Y」や「Yes」、「YES」など**

のいずれかの場合は、最初の「y＊｜Y＊」というパターンにマッチするので、何もコマンドを実行せずに「；；」でcase制御構造を脱出する。

一方、最初のパターンにマッチしなかった場合は、2番目の「＊」というパターン、すなわち「任意の文字列」に必ずマッチするので、exitが実行されてスクリプトが終了する。

なお、プロンプトに対してEnterキーだけ押すと、シェル変数ansには空文字列（長さ0の文字列）が設定される。空文字列は「y＊｜Y＊」というパターンにはマッチしないため、上の例では「n」を入力した場合と同様にスクリ

プトが終了してしまう。もし、Enterキーだけが押された場合も処理を続行したいなら、空文字列を表わす「''」（引用符2つ）を最初のパターンに「｜」とともに追加して、「y＊｜Y＊｜''」と書けばいい。

この節での要点をまとめると、

- **ユーザーに二者択一させる場合は、プロンプトの表示→readによるキー入力→if/case制御構造による分岐処理という流れでスクリプトを作成する。**
- **プロンプト用のechoは、-nオプションで改行を抑制し、標準エラー出力（/dev/stderr）にリダイレクトする。**
- **「y」や「Y」などいくつかの文字列以外はすべて「n」と見なして処理したほうが安全。**
- **Enterキーを「n」と見なす場合は「y＊｜Y＊」、「y」と見なす場合は「y＊｜Y＊｜''」というパターンを使う。**

となる。

### ●選択肢を一覧表示して選択させるには

シェルスクリプトを実行中に、多数の選択肢の中からユーザーに1つだけ選んでもらう処理を考えてみる。選択肢の数が多いことを考慮すると、選択肢の表示はプロンプトとは独立させたほうがいい。また、選択肢の内容をキー入力して選ぶのは面倒なので、選択肢を表示する際に通し番号を付け、ユーザーにはその番号だけを入力してもらうことにしよう。

処理の流れは以下のようになる。

**(1) 複数の選択肢を通し番号付きで一覧表示する。**
**(2) プロンプトを表示して番号の入力を促す。**
**(3) ユーザーの入力した番号を取り込む。**
**(4) 入力した番号が有効かどうかで分岐する。**
**(b) 有効な場合は、その番号の選択肢の内容をシェル変数に格納して繰り返し処理から抜ける。**
**(a) 無効な場合は、警告メッセージを表示後、(1)に戻って処理を繰り返す。**

こうしたスクリプトの具体例として、3つの選択肢「foo」「bar」「baz」の中から1つだけ選択するスクリプトsel3が**リスト1**だ。

このスクリプトを実行すると、

```
1) foo
2) bar
3) baz
選択は? (1-3)
```

と選択肢が通し番号付きで一覧表示され、プロンプト「選択は? (1-3) 」が表示される。

有効な番号（1から3）のいずれか、たとえば2を入力してEnterキーを押すと、

```
選択したのはbarです。
```

と、対応する選択肢を含む文を表示して終了する。

それ以外の文字を入力した場合は、「不正な番号です」という警告メッセージが表示され、再び選択肢の一覧とプロンプトが表示される。処理を中断するにはCtrl-Cキーを押せばいい。

スクリプトの内容を簡単に説明しよう。処理の中心となるのは3～17行目のwhile制御構造で、シェル変数ansが空文字列の間ずっと繰り返し実行される。

4～8行目が選択肢の一覧を表示する処理にあたる。通し番号が格納されるシェル変数numに初期値0を設定したあと、for制御構造のリストに指定された選択肢「foo」「bar」「baz」を、通し番号とともに1つずつ表示する。6行目のコマンド置換と外部コマンドexprを組み合わせた数値演算については、本連載の第6回（2001年4月号）を参照されたい。bashでのみ実行するなら、この部分は「num=$((num+1))」と$((…))構文を使って書いてもいい。

9行目ではプロンプトを表示する。範囲の表示には、for

**▼リスト1　3つの選択肢から1つを選ぶスクリプトsel3**

```
 1: #!/bin/sh
 2: ans=''
 3: while [ -z $ans ]; do
 4:   num=0
 5:   for f in foo bar baz; do
 6:     num=`expr $num + 1`
 7:     echo "$num) $f" > /dev/stderr
 8:   done
 9:   echo -n "選択は? (1-$num) " > /dev/stderr
10:   read reply
11:   case "$reply" in
12:   1) ans=foo;;
13:   2) ans=bar;;
14:   3) ans=baz;;
15:   *) echo "不正な番号です" > /dev/stderr
16:   esac
17: done
18: echo "選択したのは$ansです。"
```

制御構造の繰り返しが終了した時点で、シェル変数numに選択肢の総数が格納されていることを利用している。

10行目でreadを使ってキー入力（番号）をシェル変数replyに格納し、11～16行目のcase制御構造で分岐する。入力した番号が有効な場合（1から3）、対応する選択肢をシェル変数ansに格納する。これで、次回のwhile制御構造の繰り返し判定では条件式「-z $ans」が不成立となり、18行目の処理にスキップする。

一方、入力した番号が無効な場合は「不正な番号です」という警告メッセージを表示する。ansの内容は空文字列のままなので、次回も繰り返し判定の条件式「-z $ans」が成立し、4行目以降の処理が繰り返される。

続いては、選択肢の数や内容がスクリプト実行時に決まるような場合でも使える、汎用性のあるスクリプトに改良してみよう。具体例として、スクリプト実行時にコマンドラインで指定した複数の選択肢の中からユーザーに1つだけ選んでもらうケースを考える。

たとえば、

```
$ selarg hoge fuga hogehoge fugafuga
```

として実行すると、

```
(1) hoge
(2) fuga
(3) hogehoge
(4) fugafuga
選択は? (1-4)
```

のように、引数で指定した選択肢が通し番号付きで一覧表示されるわけだ。

リスト1のスクリプトsel3では、選択肢の一覧を表示するfor制御構文のリストや、入力した番号に基づいてシェル変数ansに選択肢の内容を格納する部分に、選択肢が直接記述されている。これを、コマンドラインで指定した引数の並び「"$@"」や、それぞれの引数の内容（$1、$2、…）で書き換えればいい。

これに従って改良したのがスクリプトselarg（**リスト2**）だ。ほかのスクリプトにも簡単に流用できるように、「選択肢を通し番号付きで一覧表示して、ユーザーに1つだけ選んでもらう」という処理は、シェル関数sellistとして定義している（2～19行目）。シェル関数についての詳細は、前回（2001年7月号）の本連載を参照されたい。

シェル関数sellistのおおまかな流れはリスト1とほぼ同じだ。違うのは、7行目のfor制御構造のリストに、シェル関数の引数の並びを表わす「"$@"」を指定している点と、入力した番号に基づく分岐処理（13～17行目）が、case制御構造ではなくif制御構造で行われる点だ。

入力した番号が有効かどうかは、「"$reply" -ge 1 -a "$reply" -le $#」という条件式で判定される。これは、「replyの値が1以上で、かつ引数の数以下である」場合に成立する（「$#」はシェル関数の引数の数を表わす）。

また、入力した番号に対応する選択肢をansに設定する14行目では、引数をコマンドラインとして評価する組み込みコマンドevalが使われている。このevalについては次回詳しく取り上げることにしよう。

関数の定義が終わったら、あとはコマンドライン引数の並び「"$@"」をsellistの引数に指定して実行し（20行目）、シェル変数ansに返された結果を含む文字列を表示するだけだ（21行目）。

この節の要点をまとめると、

- **複数の選択肢の中からユーザーに1つだけ選んでもらう場合は、選択肢の一覧表示→プロンプトの表示→readによるキー入力→if/case制御構造による分岐処理という流れでスクリプトを作成する。**
- **選択肢には通し番号を付け、ユーザーにはその番号だけを入力してもらう。**

**▼リスト2　コマンドライン引数で指定した選択肢から1つ選ぶスクリプトselarg**

```
 1: #!/bin/sh
 2: function sellist
 3: {
 4:   ans=''
 5:   while [ -z $ans ]; do
 6:     num=0
 7:     for f in "$@"; do
 8:       num=`expr $num + 1`
 9:       echo "$num) $f" > /dev/stderr
10:     done
11:     echo -n "選択は? (1-$num) " > /dev/stderr
12:     read reply
13:     if [ "$reply" -ge 1 -a "$reply" -le $# ] 2>
    /dev/null; then
14:       eval "ans=\$$reply"
15:     else
16:       echo "不正な番号です" > /dev/stderr
17:     fi
18:   done
19: }
20: sellist "$@"
21: echo "選択したのは$ansです。"
```

**・選択肢の数や内容がスクリプト実行時に決まるような場合でも選択処理は可能。**

となる。

●選択肢の表示や入力処理を代行するselect制御構造

bashに用意されたselect制御構造を利用すると、前節とほぼ同じ処理を、わずか数行の記述で実現できる。その理由は、以下に示す処理をselect制御構造が自動的に行ってくれるからだ。

**(1) 選択肢を通し番号付きで一覧表示する。**
**(2) プロンプトを表示して番号の入力を促す。**
**(3) ユーザーの入力した番号を取り込む。**
**(4) 番号に対応する選択肢（番号が無効な場合は空文字列）をシェル変数に格納する。**
**(5) (1)に戻って処理を繰り返す。**

スクリプトの作成者は、「選ばれた選択肢の内容に応じて何らかの処理を行い、繰り返し処理から脱出する」という部分だけ記述すればいい。

select制御構造の書式は以下のとおりだ。

```
select 変数名 in リスト; do
  選択後の処理
done
```

「変数名」（$は付けない）に記述したシェル変数には、ユーザーが入力した番号に対応する選択肢が格納される。「リスト」には、複数の選択肢を空白文字で区切って書けばいい。この書式は、繰り返し処理を行うfor制御構造とよく似ている。

具体例として、前節で作成したselargを、select制御構造を利用して書き換えたスクリプトselarg2を**リスト3**に示す。21行もあったスクリプトと同じ処理が、半分以下の9行で実現されている。なお、select制御構造を使ったスクリプトはbashでしか動作しないため、1行目の記述は「#!/bin/bash」としている。

2行目で設定しているPS3は、select制御構造のプロンプトを格納する組み込み変数だ。PS3が未設定の場合、「#?」という不親切なプロンプトが表示されるので、「選択は?」という文字列を設定している。

続く3～9行では、select制御構造による繰り返し処理が行われる。選択肢のリストには、コマンドライン引数の並び「"$@"」が設定され、シェル変数ansに選択結果が格納される。

たとえば、このスクリプトを、

```
$ selarg2 hoge fuga hogehoge fugafuga
```

として実行すると、

```
(1) hoge
(2) fuga
(3) hogehoge
(4) fugafuga
選択は?
```

のように、コマンドライン引数で指定した選択肢が通し番号付きで一覧表示される。

4～8行目は、ユーザーが番号を入力した後で実行される処理だ。この時点で、シェル変数ansには、ユーザーが入力した有効な番号に対応する選択肢か、無効な番号に対応する空文字列が設定されている。ansが空文字列ではない場合は、5行目のechoでansの内容を含む文を表示し、6行目のbreakでselect制御構造から抜ける。一方、ansが空文字列の場合は、8行目で警告メッセージを表示して、再度選択肢の一覧が表示される。

この節での要点をまとめると、

**・複数の選択肢の中からユーザーに1つだけ選んでもらう処理は、bashのselect制御構造で簡単に実現できる。**
**・スクリプトに記述するのは、「選ばれた選択肢の内容に応じて何らかの処理を行い、繰り返し処理から脱出する」という部分だけ。**

ということになる。

**▼リスト3　select制御構造を使ったスクリプトselarg2**

```
1: #!/bin/bash
2: PS3="選択は? "
3: select ans in "$@"; do
4:   if [ -n "$ans" ]; then
5:     echo "選択したのは$ansです。"
6:     break
7:   fi
8:   echo "不正な番号です" > /dev/stderr
9: done
```

## 今月のスクリプト

後半は、与えられたテーマを実現するスクリプトの作成手順を説明する。

今月は、

**・指定した文字列をコマンドラインに含むプロセスを一覧表示し、選択したプロセスだけを終了させる「killsel」**

というスクリプトを作成する。

プロセスIDやジョブ番号を指定する必要のあるkillと違って、コマンド名の一部を指定するだけでいい。また、プロセス情報の一覧が表示されるので、同じコマンドを複数実行している場合も判別が容易だ。

killselの基本的な処理の流れは、

**(1) プロセス情報をpsで出力する**
**(2) 指定した文字列を含むものだけにgrepで絞り込む**
**(3) 得られたプロセス情報を一覧表示して選択させる**
**(4) 選択されたプロセスをkillで終了する**

となる。以下では、まず(1)(2)の処理について説明し、続いて(3)(4)の処理を追加してスクリプトkillselを完成させることにしよう。

### ●psの出力をgrepで絞り込む

プロセス情報を表示するpsには、表示する情報の種類をユーザーがカスタマイズできるoオプションが用意されている。

たとえば、

```
$ ps o pid,command
```

とすると、プロセスIDとコマンドラインの内容だけが表示される。「pid」などの項目を示すコードについては、psのマニュアルを参照されたい。

今回は、ユーザーの所有するすべてのプロセスに対して、プロセスID、制御端末名、コマンド名の3つを表示することにしよう。

対応するコマンドラインは、

```
$ ps xho pid,tty,comm
```

となる。なお、xオプションは他の制御端末で実行中のプロセスやデーモンなどのプロセスも表示するオプション、hオプションは先頭に出力されるヘッダ行を抑制するオプションだ。

psの出力を特定の文字列を含むものだけに絞り込むには、文字列検索コマンドgrepを利用する。

たとえば、

```
$ ps xho pid,tty,comm | grep 'gnome'
```

とすると、「gnome」という文字列を含むプロセス情報だけが表示される。

ただし、今回のpsの出力にはプロセスIDと制御端末名も含まれるため、このままではそれらと一致してしまう場合もある。たとえば「tty」を含むコマンドを検索しようとしても、制御端末を持つ全プロセスが表示されてしまうのだ。この問題を解決するには、プロセスIDと制御端末名の表示に使われる行頭から15文字分の文字列を読み飛ばせばいい。

具体的には、

```
$ ps xho pid,tty,comm | grep -E '^.{15,}tty'
```

とする。

検索パターンに含まれる正規表現「^.{15,}」は、行の先頭から15文字以上の任意の文字列にマッチする。このため、その後の「tty」は16文字目以降の文字列に対してしかマッチしないのだ。なお、こうした{ }による文字数の指定は「拡張正規表現」と呼ばれ、grepで使う場合は-Eオプションを指定する必要がある。

以上のことを踏まえると、スクリプトの前半部分のプロトタイプは以下のようになる。

```
pslist=`ps xho pid,tty,comm | grep -E "^.{15,}$1"`
```

まず、psで出力されたプロセス情報が、スクリプトの最初の引数（$1で参照）をコマンド名部分に含むものだけにgrepで絞り込まれ、コマンド置換によりシェル変数pslistに格納される。

なお、スクリプト実行時にコマンドライン引数を1つも指定しなかった場合は、検索文字列が「^.{15,}」になり、すべてのプロセス情報がマッチする。

### ●選択肢の一覧表示とキー入力はselect制御構造で

シェル変数pslistに格納されたプロセス情報を番号付きで一覧表示し、ユーザーに番号を入力させるという処理には、今回登場したselect制御構造がうってつけだ。

後半部分のプロトタイプは以下のようになる。

```
PS3='which process? '
select ans in $pslist; do
  if [ -n "$ans" ]; then
    echo "killing $ans"
    kill `echo $ans | cut -b 1-5`
    break
  fi
  echo "Invalid Number" > /dev/stderr
done
```

まず、入力を促すプロンプト「which process?」を組み込み変数PS3に設定し、select制御構造を実行する。psの出力を取り込んだシェル変数pslistの内容をリストに指定しているので、それぞれのプロセス情報が通し番号付きで一覧表示される。

ユーザーが番号を入力すると、do～done間の内容が実行され、再び一覧表示に戻る。基本的にはこの繰り返しだ（中断はCtrl-Cキー）。

do～done間では、if制御構造の「-n "$ans"」という条件式で、シェル変数ansの内容が1文字以上ある（空文字列ではない）ことを判定している。有効な範囲の番号を入力した場合、ansには選択した番号に対応するプロセス情報が格納されているので、この判定は成立する。そこで、echoでプロセス情報を含む文字列を表示したのち、プロセス情報の先頭に書かれた「プロセスID」（1～5文字目）だけを抜き出してkillの引数に指定し、breakでselect制御構造から脱出している。

一方、入力した内容が適切な範囲の番号でない場合は、シェル変数ansは空文字列になるのでif制御構造の条件式は成立しない。そこで、標準エラー出力に警告メッセージを出力し、再びプロセス情報の一覧表示に戻る。

**▼リスト4　スクリプトkillsel**

```
 1: #!/bin/bash
 2: IFS='
 3: '
 4: pslist=`ps xho pid,tty,comm | tac | tail +7 | grep
    -E "^.{15,}$1"`
 5: PS3='which process? '
 6: select ans in $pslist; do
 7:   if [ -n "$ans" ]; then
 8:     echo "killing $ans"
 9:     IFS=' '
10:     kill `echo $ans | cut -d' ' -f1`
11:     break
12:   fi
13:   echo "Invalid Number" > /dev/stderr
14: done
```

### ●スクリプトに関するプロセスを一覧から取り除く

前半と後半のプロトタイプを組み合わせると、スクリプトは一応完成するが、2つほど問題がある。問題を修正したスクリプトkillsel（**リスト4**）を見てほしい。

まず、2～3行目で、組み込み変数IFSの内容（初期値はスペース・タブ・改行）を改行のみに変更している。6行目でシェル変数pslistの内容をワード分解する際、IFSにスペースが含まれていると、プロセスID、制御端末名、コマンド名がそれぞれ別のワードに分解されてしまって都合が悪いのだ。一方、10行目で実行されるkillは、IFSにスペースが含まれていないとうまく動作しないので、9行目でIFSの内容をスペースに変更している。

もう1つの問題は、このスクリプトに関するいくつかのプロセス情報が、選択肢の中に含まれてしまうことだ。具体的には、スクリプト本体（killsel）、コマンド置換で起動されるサブシェル、サブシェル内で実行されるpsとgrepの4つのプロセスが該当する。

これらを選択肢から除外するには、psで出力されるプロセス情報から末尾の4行を取り除けばいい。修正個所は、4行目のコマンド置換内部で、psとgrepの間に、入力行を逆順に並び替えて出力する「tac」と、入力行の7行目以降のみを出力する「tail +7」を挿入する。

実は、「入力の末尾N行を取り除く」コマンドは用意されていないので、「入力行を逆順に並び替えて、先頭のN行を取り除く」ことで代用しているのだ。追加したtacとtailのプロセスも同様に選択肢に含めないようにする必要があるため、tailの引数は先頭4行を取り除く「+5」ではなく、先頭6行を取り除く「+7」になる。tacによりプロセスIDの大きいものから順に表示されるようになるが、これについては特に問題はないと思われる。

リスト4のスクリプトは、「-KILL」などのオプション指定により任意のシグナルを送信するといった機能が欠けている。腕に自信のある方は、この機能を組み込んでみよう。また、select制御構造の代わりに、今回の前半で作成したシェル関数sellistを利用して、Bourneシェルでも動作するようにスクリプトを書き換えてみるのもよいだろう。

# Linux日記

## 第23回　メール配送（10）

**エイリアスという機能を使うと、メールアドレスの別名を定義できます。今回は、このエイリアス機能について説明をしましょう。**

illustration ; Aki

文：榊 正憲
*Text : Masanori Sakaki*

前回予告した眠り姫状態のWindows 2000マシンの復旧とか、新しいサーバの構築などは、実はまだやっていない。仕事の締め切りとか、1年間動かさなかった原チャリの復活とかで、何となく先送りされている。ついでに、PlayStation 2で都バスを運転してみたり、衝動買いしたゲームボーイアドバンスを評価してみたりと、何かと忙しかったのである。

都バスの運転は、「東京バス案内」というゲームである。レース系ゲームと異なり、ひたすら安全運転、尊法運転が求められる。A級ライセンスとニ種免許の違いだ。進路変更や右左折ではきちんとウインカーを出さなければならないし、交差点内での立ち往生などもってのほかである。加えて、止まるバス停、止まらないバス停の判断、車内アナウンス、ドア操作、安全確認など、レースとは違う忙しさがある。

このゲームの難点は、ビューである。運転系ゲームでは、車の背後の視点で運転することが多いが、バスだと車体が大きすぎ、前がよく見えない。信号待ちで、前に止まっている車がまるで見えないのだ。運転席ビューにすると、左右方向の視角が狭い。バックミラーはウインカー操作時しか表示されないので、側方、後方はわからない。広い視野角とバックミラーは、大型車運転時の必須項目なので、ちと残念だ（筆者は一応大型免許と牽引免許を持っている。残念ながらどちらも一種免許なので、バスの運転手にはなれない）。ちなみに、本物の大型車シミュレータには、視野を表示するために3面のディスプレイを使っているものもある。もちろん、バックミラーも常時表示されている。

公共輸送機関の運転といえば、しばらく前に「電車でGo! 3」も買った。実家で暮らしていたころにさんざんお世話になった総武線の秋葉原までの運転があって、妙にうれしかった。さすがに乗り慣れていた区間だけあって、ほぼ一発でクリアできた。おまけに付いていたデモ版の山陽新幹線もやってみた。こちらはATC車上信号にしたがって運転すれば、あとは定点停車だけなので、はっきりいって通勤電車よりはるかに簡単である。新幹線を運転しながら、あらたな機能に気が付いた。カーブで世界が傾くようになったのだ。「そうだよなぁ、新幹線のカーブには、やっぱカントがなきゃね」と素朴に感動したのであった。

ゲームボーイアドバンスは、とりあえず買ったものの、スーパーマリオはもっぱら子供がやっていて、所有者である筆者は、ゲームボーイカラー用のウィザードリィをやっている。老眼が始まっている筆者の場合、あの画面サイズで文字表示が多用されるゲームは辛いというのが実感である。大画面液晶を搭載した「ゲームボーイシニア」の発売が望まれる。

さて、sendmailの話の続きだ。

### ■エイリアス

sendmailにはエイリアスという機能がある。sendmailに限らず、メール受

信を行えるたいていのMTAは同等の機能を持っている。エイリアス（Alias）というのはメールアドレスに別名を定義する機能である。

MTAが受信する有効なメールアドレス（ユーザー名）は、何らかの形でそのMTAホスト上に登録されている（中継処理を行う際は、メールアドレスが有効であるかどうかは関係ない）。UNIXの場合なら、そのUNIXホストに登録されているユーザー名が有効なメールアドレスとなる。システムによっては、ホストのユーザーアカウントとは別に、メール専用のユーザーアカウントを、MTAのレベルで登録するものもある。いずれにせよ、これらのメール用アカウントは、実在のユーザー、つまり実在のメールボックスを表すものであり、これらを宛て先アドレスとするメールはMTAで受信され、メールボックスにスプールされる。受信したメールについて、該当する宛て先がない場合は、宛て先不明というエラーになり、発信者にエラー通知が返送されることになる。

エイリアスでは、このような「実在の」メールアドレスとは別に、有効なメールアドレスを定義することができる。メールの送信時には、実在のユーザーのメールアドレスと同じように、宛て先にエイリアス名を指定することができる。

エイリアス機能は、エイリアス名と、それに割り当てる別のメールアドレス類の表という形で実現される。メールを受信した際に、宛て先アドレスがエイリアス名であった場合、そのエイリアス名に対応するメールアドレスにメールが配信される。

通常、エイリアスに対しては実在のメールアドレスを指定するが、別のエイリアスを指定することもできる。この場合、多重定義されたエイリアスが最終的に解決されるまで、何度もエイリアス評価を繰り返す。また、1つのエイリアスに対して、複数の名前（メールアドレスや別のエイリアス）を指定することもできる。この場合、そのエイリアスを宛て先として届いた1通のメールは、複数の宛て先に転送される。

エイリアスのまったく別の用途として、受信したメールをプログラムで処理するというものがある。通常、メールはユーザーのメールボックスにスプールされるが、ユーザーに送り届けるのではなく、何からのプログラムに対する入力とするのである。また、受信したメールを直接ファイルに記録することもできる。

エイリアスの用途をいくつか紹介しよう。

### ■実在しないユーザーのメールアドレスの登録

ユーザーアカウントの存在しないメールアドレスをエイリアスで定義し、それを実在のユーザーに配信することができる。メールシステムを運用する場合、postmasterというユーザー名でメールを受信できるようにしなければならないが、実際にpostmasterというユーザーアカウントを用意する必要はない。postmasterというエイリアスを定義し、postmaster宛のメールが、たとえばrootに配信されるように設定することができる。これにより、postmaster宛のメールはユーザー不明のエラーとはならず、root宛に届くようになる。

また人事異動などで、メールアドレスが変わったときなどにもエイリアスを使える。たとえば、masa@tech.ascii.co.jpというメールアドレスを使っていたユーザーが、異動によりmasa@pub.ascii.co.jpというアドレスに変わり、それに伴って、古いアドレスを管轄していたMTA上でmasaというユーザーアカウントが抹消されたとしよう。この場合、古いアドレス宛のメールは、宛て先不明エラーとなってしまう。これでは不便なので、古いアドレスを管轄するMTA上に、masaというエイリアスを登録し、masa@pub.ascii.co.jpというメールアドレスに転送するように設定しておけばよい。これにより、古いアドレス宛でも、新しいアドレス宛でも、新しいアドレスにメールが届くようになる。もちろん、このような転送は、社内だけに限られるわけではない。社外への出向など、違うドメインでも何の問題もなく転送できる。

### ■グループを表すメールアドレスを登録

エイリアスを使って、複数のメールアドレスから構成されるグループを代表して表すメールアドレスを登録することができる。たとえばsales@ascii.co.jpというエイリアスを定義し、営業部全員のメールアドレスを登録すれば、sales@ascii.co.jpという1つのメールアドレスを指定して、営業部全員にメールを送ることができる。rootになれる権限を持つ全ユーザーをadminというエイリアスに登録すれば、管理関係の連絡を円滑に行えるようになるだろう。

このようなグループを代表するメールアドレスは、組織内でも便利に使えるが、対外的に公開することもできる。よく使われているのがwebmasterといったアドレスである。Webページの管理グループのメンバーをこのエイリアスに登録しておけば、外部からのWebページに関する問い合わせのメールが管理グループに届くようになる。

このような登録方法は、組織内の人事異動などに際しても便利である。担当者が変わったときには、エイリアスを変更するだけでよい。対外的に公表しているアドレスを変更する必要はない。

### ■実在するユーザー宛に届いたメールを転送する

エイリアスは、実在するメールアドレスについても定義できる。この場合、そのアドレスを宛て先とするメールは、エイリアスで指定された宛て先に転送される（元のアドレスには配信されなくなる）。たとえば、rootは実在するユーザーであるが、root宛のメールを読むためにrootとしてログインしなければならないというのは不便である。そこで、root宛のメールを、システム管理者の個人メールアドレスに転送するように、rootというエイリアスを定義し、管理者の個人アドレスを指定する。これにより、root宛のメールが管理者の個人環境宛に届くようになる。管理者が複数いる場合は、adminといった名前の管理者グループ用のエイリアスを定義し、それに送るようにすればいいだろう。

また、会社のMTAに届いたメールを、個人で契約しているプロバイダに転送するといった設定も可能である。このような設定を許すかどうかは、組織の管理ポリシー次第であるが、不可能なことではない。もっとも、個人レベルの転送であれば、後述するフォワード機能を使うほうが簡単である。

### ■届いたメールをプログラムなどで処理

エイリアスの用途はメールの転送だけではない。エイリアスに対して、メールアドレスではなく、プログラムや各種スクリプトを指定することもできる。これにより、受信したメールをプログラムで処理することが可能になる。メーリングリストサーバ、自動応答メールサービスなどは、エイリアスのこの機能を使って実現できる。

### ■メールをファイルに保存

エイリアスの右辺に、スラッシュ（／）で始まるファイルのパス名を指定することで、受信したメールを直接ファイルに保存することができる。ローカル配信エージェントによる配信も、受信したメールをファイルに保存するが、この場合は、ディレクトリやユーザー名は変更できない。エイリアスによるファイル保存指定は、任意のファイルを指定できる。

### ■/etc/aliases

sendmailを使う場合、エイリアスの定義は、/etc/aliasesというテキストファイルに記述する。エイリアス名にはコロンを付け、そのあとに、別のアドレスなどを列挙するという簡単な書式である（**リスト1**）。

MAILER-DAEMONは、sendmailなどのMTAが発信したメールのFrom：行に使われるユーザー名である。つまり、通常のユーザーではなく、MTAプログラムから送られたということを明示するためのアドレスである。たとえば、宛て先不明のメールを発信者に送り返すといったエラー通知メールなどに、From： MAILER-DAEMONといったヘッダが使われることがある（postmasterとなっていることもある）。MAILER-DAEMONはもちろん実在のユーザーアカウントではないので、なんからの理由でMAILER-DAEMON宛のメールが届いた場合は（エラー通知メールがエラーになったなど）、人間の管理者を表すpostmasterに送るように定義している。

前に触れたように、postmasterは必要なアドレスであるが、実在のユーザーではない。そのためpostmasterもエイリアス名であり、postmaster宛のメールは、実在するユーザーであるrootに送られる。また、UNIXの管理上用意されている各種擬似ユーザーについても、同じような処理をしている。これらのアカウントは、ファイルのパーミッション設定などのために用意されたものであるが、これらの権限で動作するプログラムには、何らかのメールを送信するものもある。したがって、これらを宛て先とするメールが送られることもあるのだ。これらの擬似ユーザーは、実在するユーザーアカウントであるが、人間がこのアカウントを使ってログインすることはない（通常はログイン禁止になっている）。そのため、エイリアスによって、rootに配信するようにしている。

rootは実在するユーザーなので、ログインしてメールの送受信を行えるが、rootとしてのログインはあまり行わないし、制限も多い。また、複数のユーザーがrootになれる場合、行き違いなどが発生してしまうかもしれない。そこで、root宛のメールは、adminという宛て先に転送するようにしている。adminはやはりエイリアスで、admin宛のメールは、masa、tetsuという実

**▼リスト1　aliasesの例**

```
MAILER-DAEMON: postmaster
postmaster:    root
bin:           root
daemon:        root
news:          root
nobody:        root
operator:      root
root:          admin
admin:         masa@takobeya.com,
               tetsu@takobeya.com
webmaster:     tetsu@takobeya.com,
               shun@takobeya.com
```

在のユーザーに送られる。

webmasterもadminと同じような仕組みで、外部からこのアドレス宛に届いたメールは、tetsuとshunというユーザーに送られる。組織内で担当が変わった場合でも、外部に公開したアドレスを変更する必要はない。単にエイリアス定義を変更するだけだ。

aliasesファイルは、sendmail.cfと同じように、常に行頭から開始する。行頭から始まっていない行は、前の行の継続行となる。**リスト1**の例では、webmasterとadminの定義に継続行を使っている。

### ■エイリアスループ

エイリアスを複雑に定義していくと、ループになってしまうことがある。簡単な例としては次のようなものがある。masaがnoriになり、noriがmasaになるので、無限ループになってしまう。

```
masa:   nori
nori:   masa
```

sendmailはこのようなエラーをメールの配信時に検出する。つまり、エラーが発生するのは、実際にエイリアスの評価を行い、指定されたアドレスがループした時点である。newaliasesなどでエイリアスデータベースを再構築しても、ループは検出されない。

### ■プログラムによる受信

エイリアスの重要な機能として、受信したメールをプログラムに処理させるというものがある。これを行うには、エイリアス定義の右辺に、｜（パイプ）記号に続けてプログラム名を記述すればよい（**リスト2**）。

これにより、junk-mlというアドレスに送られたメールは、/usr/local/bin/mlserverに送られる。必要なら、引数も指定できる。

受信メールをプログラムに渡すという処理は、sendmail.cf中で、配信エージェントの形で定義されている。progという配信エージェントは、受信したメールをプログラムに渡すためのもので、通常、**リスト3**のように定義されているはずだ。

つまり、配信エージェントプログラムは/bin/shであり、実行時には「sh -c $u」という形で実行される。$uは宛て先アドレスだが、この場合はプログラムのパス名と引数が渡されることになる。結果として、シェルを介して指定したプログラムが実行され、受信メールは標準入力でプログラムに渡されることになる。

### ■外部ファイルのインクルード

エイリアスを使って簡単なメーリングリストを作ったり、あるいはメールマガジンの配信を行うといった作業を行う場合、1つのエイリアス名に非常に多くのアドレスを登録することになる。これを直接aliasesファイルに記述すると、ファイルの可読性は低下するし、管理も面倒なので、別ファイルにしたくなるだろう。これを行うには、：include：を使う。

**リスト4**のようにすると、junk-mlを宛て先とするメールは、/home/junk/membersというファイルに列挙されたユーザーに配信される。

### ■メーリングリスト

受信したメールをプログラムで処理する例として、メーリングリストの運用がある。

メーリングリスト（ML）というのは、メールシステムを使ったグループ内の情報交換システムである。メーリングリスト用に登録されたあるアドレスに対してメールを送ると、そのメールが、メーリングリストに登録された全ユーザーに対して配信される。配信されたメールに対してメンバーが返信すると、その返信もすべてのメンバーに配信される。つまり、一定のメンバーの間で自由に連絡、討論、よもやま話をできるシステムである。

もっとも単純なメーリングリストは、メンバーのメールアドレスを登録したエイリアスで実現できる。エイリアスに登録されたアドレスにメールを送れば、メンバー全員にメールを送れるという仕組みである。しかしこの方法は、返信がうまくいかない。返信アドレスを明示的にエイリアスのアドレスにしなければならないからだ。

実際にメーリングリストを運用する場合は、メーリングリストサーバというソフトウェアを使用することになる。エイリアス機能には、特定の宛て先アドレスに送られたメールをプログラム

**▼リスト2　メールをプログラムで処理する場合**

```
junk-ml:        |"/usr/local/bin/mlserver junk-ml"
```

**▼リスト3　メールをプログラムに渡すsendmail.cfの定義**

```
Mprog,    P=/bin/sh,    F=lsDFMoqeuP,   S=10, R=20/21,
          T=X-Unix, D=$z:/,
          A=sh -c $u
```

**▼リスト4　外部のエイリアスファイルをインクルードする**

```
junk-ml:        :include: /home/junk/members
```

で処理するという機能があるので、ここでメーリングリストサーバを指定すればいい。メールを受け取ったMLサーバは、別に用意されているメンバーリストに基づいて、そのメールを多数の宛て先に再送信する。メーリングリストサーバが持つ機能について簡単にまとめておこう。

### ●MLアドレス

メンバー全員にメールを送るためのアドレスである。このアドレスをエイリアスで定義し、MLサーバソフトウェアがメールを受け取るようにする。

### ●メール配信

MLサーバは、メールを受信すると、登録されているメンバーリストに基づいて、受信したメールをメンバーに配信する。このとき、ヘッダの書き換え、配信したメールをログに記録するといった付帯処理も行う。メンバーに配信されるメール中のFrom：行は、通常はオリジナルの発信者のままとすることが多いが、実際の発信者はMLサーバプログラムとなる。

メーリングリストを運用する場合、メールを投稿できるユーザーを制限するのが普通だ。つまり、配信メンバーとして登録されているアドレスから送られたメールはメンバーに配信するが、それ以外のアドレスから送られたメールは排除するという仕組みである。この判定も、MLサーバが行うことになる。

### ●返信の処理

MLに配信されたメールに対する返信は、基本的にはMLに送ることになるが、MUAを使ったメールの返信はFrom：行宛となるので、MLではなく、発信者個人に送られてしまう。そのため、返信もMLに送るために、MLサーバは配信メール中にReply-To：ヘッダを追加し、そこにMLアドレスを指定する。これにより、返信もMLに送られるようになる。これを行わないと、手作業で宛て先アドレスをML向けに書き換えなければならない。

### ●エラー処理

ユーザーアカウントの登録ミスや抹消により、メンバーに対する配信がエラーとなることがある。このようなエラーが発生した場合、受信側MTAはエラー通知を発信側に返すことになるが、これをReply-To：で指定されたアドレスに返すと、そのエラー通知がMLに配信されてしまう。このメールはさらにエラー通知を生成してしまうので、エラー通知メールの無限連鎖が発生してしまう。しかも、参加メンバー全員に送られてしまうのである。これを防ぐために、MLサーバはErrors-To：ヘッダを用意し、エラー通知をMLに送らず、管理者などに送るようにする。

### ●Subject：の書き換え

MLで送られたメールは、MLから送られたことを明示するために、Subject：にMLの名称、メールの通し番号を表示することが多い。たとえば、「ASR－33の印字ドラムありませんか?」というSubject：を指定してMLにメールを送信すると、配信されたメールのSubject：は**リスト5**のようになる。

［ と ］で囲まれた部分は、MLの名称とメールの通し番号である。通常のMUAを使ってこのメールに対する返信を行うと、Subject：は**リスト6**のようになる。

しかし実際にメンバーに配信されるメールでは、**リスト7**のようになるのが普通だ。

常にML情報が最初にあり、「Re：」は、実際のSubject：文字列の前に付くのである。このようなSubject：の書き換えは、MLサーバが行っている。具体的には、受信メールから［ と ］で囲まれた文字列を削除し、新規にML情報を追加するという形になる。

### ●管理機能

MLサーバは、各種の管理機能も持っている。以下のメンバー情報の変更などは、MLサーバのレベルで処理を行える。MLサーバソフトウェアは、各種設定、メンバーリストなど、関連する設定ファイルを必要とする。管理者がこれらを手作業で変更すれば、各種の管理作業を行うことができるが、それとは別に、メールを使った管理インターフェイスを備えているものが多い。MLサーバは、ML配信用のメールアドレスとは別に、管理用メールアドレスを持っている。この宛て先に対して、定型書式のメールを送ることで、設定の変更など、一部の管理作業を行

▼リスト5　MLの名前や通し番号が付けられたSubject：

```
Subject: [Junk-ML 12345] ASR-33の印字ドラムありませんか？
```

▼リスト6　MLのメールに対する返信のSubject：

```
Subject: Re: [Junk-ML 12345] ASR-33の印字ドラムありませんか？
```

▼リスト7　配信されるメールのSubject：

```
Subject: [Junk-ML 12346] Re: ASR-33の印字ドラムありませんか？
```

えるのである。

### ●メンバーの登録／抹消

MLメンバーの登録、抹消は管理者が行うことになるが、自動化、半自動化することも可能だ。配信先アドレスを含む一定の書式で、管理用メールアドレスにメールを送ることで、それを受信したMLサーバが、そのメールアドレスをメンバーリストに登録するのである。また、完全自動登録ではなく、登録に際して管理者の許可を求めるようにすることもできる。抹消も同じように行うことができる。

### ●配信の停止／再開

メンバーの登録、抹消とは別に、一時的な配信の停止、再開を行う機能である。特に、メンバー登録／抹消に際して管理者の許可が必要なMLの場合、配信の停止／再開にいちいち管理者の手をわずらわせるのは効率的ではない。メールアドレスの抹消や再登録を伴わない配信の停止、再開は、管理メールアドレスに適当な形式のメールを送る形で実現できると便利だ。

### ●アーカイブサービス

過去にMLで配信されたメールを保存しておき、あとから参照するためのサービスである。これは、Webサーバなど、別のサービスを使って過去のメールを公開するといった方法もあるし、あるいは、管理アドレス宛に、過去ログの取り寄せのコマンドを記述したメールを送るという方法もある。MLサーバは、このようなメールを受信すると、発信アドレスに対して、求められた過去のメールを送る。

### ■自動応答メールサービス

メーリングリストと並んでよく使われるメールのプログラム処理として、各種の自動応答サービスがある。これは基本的に、MLサーバの過去ログのアーカイブサービスと似たようなものである。たとえば、RFCなど、大量にある公開文書を提供するサービスとして使うことができる。このような用途には、現在ではWebなどの対話サービスを使うのが普通であるが、かつてこのような機能がまだ自由に使えなかった頃（常時接続でないとか、回線が細いなど）は、広く使われていた。

もちろん、自動応答サービスは、アーカイブ検索サービスだけではない。要は、何らかのサービスを行うためのインターフェイスとして電子メールを使うことができるのか、そしてプログラムによる自動処理が可能なのかといった点を考えるということだ。

### ■フォワード機能

エイリアスを設定することによって、あるアドレス宛に送られたメールを別のアドレスに転送することが可能になる。しかし、エイリアスデータベースはシステムワイドなものであり、管理権限を持たない個々のユーザーが手を加えることはできない。かつては、aliasesファイルをユーザーが自由に変更できるという運用形態もあったが、セキュリティの面で、現在では現実的ではない。それでも、各ユーザーのレベルで、自身のメールアドレスに届いたメールを別のアドレスに転送できれば便利である。これを可能にするのがフォワード機能である。

UNIXの場合、各ユーザーのホームディレクトリに.forwardというファイルを作成し、その中に転送先アドレスを書いておくことで、メールの自動転送が可能になる。また、エイリアスと同様に、受信メールをファイルに保存したり、プログラムで処理するように記述することもできる。

プロバイダのメールサーバを使うなど、MTAホスト上に正式なユーザーアカウントを持っていなくても、フォワード機能はサポートされていることが多い。Webブラウザなどを使って、個人のメール設定を変更できるシステムの場合、設定画面中で、受信メールを別アドレスにフォワードする機能を指定できる。

フォワード機能を使う場合も、ループに気を付けなければならない。たとえば、会社宛に来たメールを個人契約しているプロバイダのメールアドレスにフォワードすれば、会社宛のメールを自宅や出先で読めるようになる。これだけなら問題はない。さらに、個人宛（プロバイダのメールアドレス宛）のメールも会社で読めるようにと、プロバイダ側のアドレス宛のメールを会社のアドレスにフォワードしてしまうと、ループになってしまう。どちらに届いたメールも、会社とプロバイダの間を何度も往復することになる。

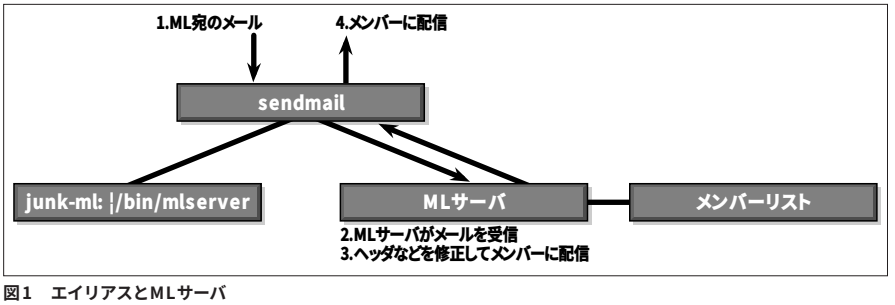
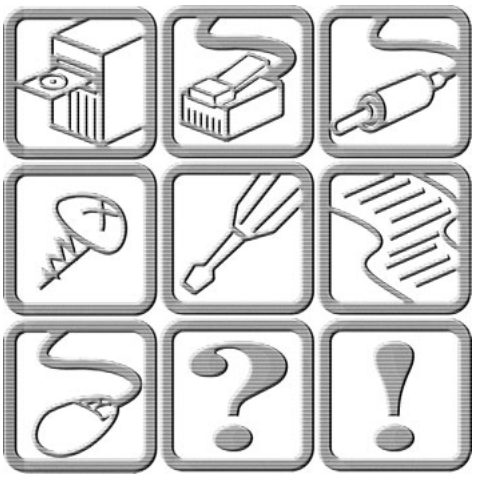



図1　エイリアスとMLサーバ

# Try & [trái] Try



## 情報化の流れの中で思うこと

文：政久忠由
*Text : Tadayoshi Masahisa*

情報なるものは、インターネットに限ったことではないけれど、日々ものすごい勢いで増え続け、僕が必要としている、知りたい、また将来的に有益であるかどうかに関係なく、グッピーのごとくに増殖し、さまざまなチャネルから洪水となって降り注いでいる。昔なら得ようと思ってもなかなか集められなかった情報に、現在ではよりどりみどりを通り越してフィルタ処理にうんざりしたり、ただただ押し流されそうになったりする。

僕はここ10年くらい新聞を取っていない。もちろんまったく不自由は感じない。新聞を信用してはいないけど、嫌いというわけではない。出先で、あれば読む。

テレビはできる限り見ないようにしている。昔からテレビを見ないと寂しいとか、生きてゆけない、などと感じたことは一度たりともない。その代わりといっては何だが、レンタルビデオは週に5本、月に20本以上借りて見ている。これといった好みのジャンルを限定しているわけでもなく（でもオドロオドロしいのはなるべく借りない、怖いのは好きじゃないから）、新作を中心に手当たり次第に鑑賞している。すでに1000本以上の映画を見たことになる。でも本当に面白かったとか、感動したのは、3本だけだ。これらは大作じゃないし、監督や出演者も有名じゃないし、泣けたとか作品がすごいってわけじゃなくて、単に共鳴したと説明したほうがいいかもしれない。ただ自分の中の名作を探したり、それを自慢したり、何かを満たしたくて見ているのではないから、映画に対するこだわりといったものは何一つ持ち合わせていない。だからそれぞれの映画に合わせて自分のスタンスを優柔不断に変えられるので、よほどの駄作でない限り、どんなものでもそれなりに楽しめてはいる。

雑誌は近所のコンビニでパラパラめくって見るだけで定期的に購入しているものはない。書籍は月に新刊を数冊、あと古本屋に不要なものを売るのと同時に数冊購入する程度だ。ただ僕自身が購入するもの以外に同居人が仕入れてくる雑誌や書籍があるにはある。

僕の場合、さまざまな媒体の中で情報と接する機会がもっとも多いのはインターネットということになる。上記の媒体のすべてを代用できる力を秘めたメディアなわけだけれど、それだけにインターネットを介した情報との接触は最低限にとどめるようにいつも戒めている。

理由は単純だ。僕が消化できる情報量はそんなに多くない。だから僕は巷に溢れる情報を極力遮断しておきたい。でないと僕が考える時間がなくなってしまうのである。

### 情報化社会の光と影

僕らには、有象無象の中から、本当に必要な情報を取捨選択し、処理する能力が求められているわけなんだけど、これが準備をしないまま巻き込まれてしまうと、自分というものを見失いそうになるくらい危険な状況に陥ることがある。まぁそのこと自体は身をコミットしなければどうっ

てことないんだけど、実はそれには強い精神力というか、考える力が必要となる。僕の精神力は弱いから、情報を制限することでなんとかしのいでいるわけだ。

以前、文部省が教育改革の一環として示した情報の中に、小学校の算数で円周率を従来の3.14から簡略化して3として計算させるようにする、という報道を耳にしたことがあると思う。改革案はこれ以外にもいろいろあったのだが、報道の多くはこのことを見出しとしてクローズアップしていた。

新聞やテレビをはじめとするマスコミ、またインターネットのポータルサイトなどは、それなりの情報フィルタとなっているものの、センセーショナリズムで大衆受けを第一に考えていたり、ものの見方の偏りがあったりして、残念ながら全幅の信頼がおけるものではない。まぁこのことを嘆いてもしょうがないし、そもそも伝達情報というのは出所のソースと同じであるかどかもわからない、というか誰かの頭の中でシャッフル(要約されたり、解釈されたり、部分引用されたり)されて再構築されたものは、歪曲とは言わないまでも、本質から変化していることが多いのも事実である。

本来マスコミは真実、真相を伝える、伝えようとしていると多くの人々は考えていると思うが、このこと自体が誤りなのだ。真実や真相というのはあるような、ないようなもので、ちっとも重要ではない。本当に重要なのはそれを見聞きする側が、考えをめぐらせるに足る多角的な情報を提供することにある。つまり最終的な答えを出すのはそれぞれの個々人であって、情報を提供する側ではないのだ。しかし困ったことに現状には、考えようとしない、いや考えられない人々が蔓延してしまっている。その結果マスコミがあたかも真実のように垂れ流す情報を鵜呑みにし、企業の提案するまやかし情報に彩られたくだらない陳腐な商品を買い漁る消費者が一向に減少しないのである。

そういう考えるという教育を受けていないし、日本という国にはそういった習慣があまりないから、いや十分考えてのことだ、と言い訳をする人もいるかもしれない。まぁ宗教で成り立っている国家は小さなころから有無を言わさずそういう教育や習慣の中にあって、各人が考えをめぐらす幅を狭めているからこそ成り立っているわけだし、宗教的な締め付け以外にも、国や自治といったあるまとまりが秩序を持ってまとまっていくために法律やルールという一定の制限を設けている。実は権力の側は、統制しやすいような教育を行い、宗教もそれに利用してきたという歴史があり、今なおその傾向が続いているのだ。本当に考えることができる人々を多く生み出してしまうと、統制し難くなるためだ。

ここで冒頭の円周率の話に戻る。円周率が無理数であることはここの読者なら知らない人はいないだろう。また円に関連した公式も叩き込まれていると思う。円の半径を与えられればその公式を使って円周の長さや面積を求め、答えを出せるだろう。でも円周率とは何かと聞かれて、円周の長さとその直径の比であるとか、円の面積と半径の平方との比であり、それにどんな意味があるのかということはほとんどの人は説明できないはずだ。実は僕らの受けてきた教育の実態というのは、試験問題を解くために公式を覚えて正確に計算するといった、ある種姑息なテクニックの修練でしかなくて、その本質に触れ考えるためのアドバイスは何一つなかったのである。これは算数や数学だけでなく、すべての教科で言えることだ。よく学校で習ったものは社会でちっとも役に立たないと言われる。そりゃそうだ試験問題を解くためのテクニックなのだから、試験のない社会生活で役に立つわけがない。

円周率を3として計算させるということは、円周率が3であると教え込むのではなく、円周率から派生するさまざまな概念に興味を持ち、考えを巡らせるのに邪魔になりかねない小数点以下の計算を省略しているに過ぎない。これ以外の教育改革として言われていることの内容も同様のポリシーによるものだ。

にもかかわらず、マスコミはやれ学力が低下するとか言っている。このとき僕は、マスコミって、なんて馬鹿なんだろうとつくづく思った。そういう小手先のテクニックだけを身に付ける教育を忠実に受けてきた連中だから仕方ないのかもしれないけど、考えを巡らせることに重点を置いた教育というのは、まぁ理解されにくいことも確かだ。しかも、もしこれらの方針が実施されたとしても、現状の先生と呼ばれる教師ではその楽しさを芽生えさせ、大きく育てていくことは到底できないんじゃないかと思う。

僕は年齢順送りという制度や、いわゆる落ちこぼれに進行速度を合わせるようなことはクソだと思っているし、ゆとり教育って言葉も誤解されやすいので好きじゃない。僕が考える社会人としての最低限必要な教育というのは、ただ単に必要な情報を集められ、それを多角的に分析し、自分の考えをまとめられる力を養うことだ。決して緻密な計算能力や暗記の能力ではない。そんなものはいくらでも代用できる。もちろん、最終的な明快な答えを出すことを求めるのでもない。実際の問題はそんなに単純じゃない。だからある程度自分の考えをまとめたうえで、さまざまな考えを持つ人々と議論を重ねることが大切になる。自分の考えを貫き通すこともあれば、他人の意見に感化されること

もあるだろう。ここで重要なのは、議論の結果は勝ち負けではないということだ。議論は参加するすべての人々にとってプラスとなるべきものなのだ。

しかし悲しいかな現状は、議論とは名ばかりの自分の考えを押し通すために相手の考えの揚げ足取りに終始し、非難することが横行している。また結論をあまりにも急いで出そうとしている。

これらの問題の原点は、明快な答えのみを求めてきた教育にある。僕らは明快な答えに慣れてしまい、それを期待してしまっているけど、本来それは自分たちで考え、そして議論することで導き出せるかもしれない、導き出そうとするもので、それとて絶対的なものではないということを理解しておく必要があるのだ。

## IT戦略会議ってまだ存続してたかな？

ちょっと前置きが長くなったが、何でこんなことを言っているのかというと、今日の情報化社会のベクトルの先には、代表民主主義から直接民主主義があるからだ。これって遠い未来の話ではなく、5年後に訪れてもおかしくないことなのだ。選挙で代表者を選んでその人に任せることに多くの人々はうんざりしていると思う。人々に行き渡る情報が少なく、先生にすべてを一任しますだ、という時代は着実に終わりを迎えつつある。現在改革の柱のひとつとなっている多くの行政機関や特殊法人／公益法人も不要だけど、議員諸君もいらないんだよなあ、これが。すでに役所が管理する住民台帳や納税に関するデータは当然のように電子化されていて、統合はされていないかもしれないけど、内部的にはID番号で管理されている。民間のデータベース上でもそうだ。これから情報化社会をさらに進めるというのなら、早く国民背番号制でも納税者番号制でもいいから統一してIDカードを発行すべきで、それを基に公開鍵システムなどの何かしらの証明システムを厳正に運用すればいい。そうなれば現時点でも政策グループや個人が法案を立案し、最終的に電子投票で採決を行うための技術的な問題は少ないし、立法だけでなく、行政や司法も同様の手法でよりよいものにできる。

しかしここで問題となるのが、先に述べた個々人の考える力、議論し、ものごとを見極める能力なのだ。

まぁそういった意味では本当のところいつ実現できるかわからないというか、実現してしまって大丈夫かなと不安になる部分もあるけれど、とりあえず自治体レベルから実際に実施していくことで個人の意識も社会の意識もたぶん着実に変わっていくと信じている。

あっそうそう、僕は政治的な思想からこんなことを言っているわけではない。単に間接的な無駄がなくなればいいと思っているだけなのである。ただ無駄は無駄でも、直接的な無駄、個々人のある種の無駄な消費は美徳だと思っている。それはその人にとっての価値観の問題だから。でもまぁ僕にとって価値のないものがヒットしていることには呆れてしまうけれども。

そんなわけで、まずはマスコミを過信せず、押し寄せてくる情報を遮断し、情報飢餓状態に自分を置いてみることをお勧めする。きっと大切なものが見つかると思う。それと特にテレビを見る場合、効果音の存在に注意したい。映画やドラマ、バラエティでなら常套手段で別にどうってことないんだけど、報道でこれをやられるとたまったもんじゃない。内容によって声のトーンを変えるのはまだ許せても、無意識に作用する効果音だけは付けてほしくない。活字、音声、映像、効果音、この順序は受け取る側がその情報を意識的に処理できるかどうかを表している。活字はイマジネーションが働き、受けて側によってその感想はさまざまだ。しかし効果音となると、ほとんど個人差なく無意識下でベクトルの方向を決定してしまう。シナリオどおりのコメントが求められ、それに沿うご意見番やスペシャリストも見苦しいけど、効果音でサブリミナル効果を狙ったつくりは、ほんと最悪だ。

ちなみに僕のお気に入りのニュース番組はNHKの手話ニュース。この番組が見たくてテレビをつけるわけじゃないんだけど、ザッピングしているとなぜかしらこの番組に落ち着いている。だから何時からやっているのかは知らない。でも主要なニュース数本をシンプルに伝えているところが落ち着く理由だと思う。それに番組後半の連載ものが結構奥ゆかしい。しばらくあいだみつお氏の書詩の紹介が続いていた。最近は金田一春彦氏や戸田奈津子氏などによることばに関するうんちくやウラ話を聞いた。いやぁ日々のニュースの中で最低限知らなきゃいけないものって何なんだろうと、しみじみと考える今日この頃だったりする。まぁ目まぐるしく変化しているコンピュータ関連分野では、そんな悠長なことは言っていられない気もするけれど。

## ADSL導入後の僕の生活の変化

さてADSLを導入してから数カ月が経過した。動的なアドレスの割り当てが一時的に取得できないなどの些細なトラブルに数回見舞われたものの、総じて快調な日々を送れ

ている。

僕の場合、基本的に24時間この通信環境下に身を置いているわけで、また時間にも束縛されていないので、絶対的な利用者が少ない時間帯に主に利用するというスタイルをとっている。このこと自体は以前のダイヤルアップ環境でも同じだったのだけれど、結局自分のインターネットへの接続回線が太くなったとしても接続先のサーバや経路の性能が向上しない限り、変えようがないのである。以前からインターネットバンキングやオンラインショッピングを頻繁に利用していたが、手前の回線の増強はまったく影響せず、接続先のサーバの処理能力にいらいらしている。

しかし、いくつか変化した部分もある。

まず1つ目は、リアルタイム証券取引システムを手に入れたことである。これまで資産内の証券運用部分は、成り行きでめぼしいところを物色し、塩漬けにしていたのだが、放っておけば右肩上がりの経済に連動して資産が増えるという時期ではない。実際株式市場は指数を見ればわかるように低迷している。しかし個別銘柄を見てみるとまったく動いていないというわけではないのだ。全体の売買代金が小さく、巨額の資金を動かす投資家は必要以上に株価変動に影響してしまうので、身動きが取りづらい状況なんだけど、僕のような少ない資金のユーザーには、全体に手ごろで悪くない環境なのである。

というわけで、リアルタイム証券取引システムを導入した。やり取りするデータ量としては多くなく、ADSLの常時接続環境の部分を活用したものといえる。専用のソフトウェアを使って、証券会社と接続すると、各銘柄の売買が成立した株価だけでなく、売り買いそれぞれの株価にスタックされている株数などをリアルタイム（十数秒程度のディレイはあるとおもうけど）に表示してくれる。これは売り手と買い手の思惑が非常によくわかる。売り浴びせたいのか、買い上げたいのか、はたまた買い支えようとしているのかなど、AラインとBラインとの間で繰り広げられるプレイヤーの動向を目の当たりにできるのだ。でもそう簡単に予想できないのも株価で、ある傾向というものも巨額の資金を有するプレイヤーの登場によって目まぐるしく変化していくのである。僕のような小さな資金のプレイヤーは、信用取引を最大限に活用したとしても、数千万円程度の力しかない。東証一部の取引量が小さくなっている状況とはいえ、日々7000億円以上が動いている。だからまぁ流れに乗れるように努力するしかないのだけれど。もちろんほどほどにね。

次に変化したことと言えば、とりあえずダウンロードしておこうと思わなくなったことだ。いつでもそんなに待つことなく必要なデータを取得できるから、ダウンロードしたドキュメントやアーカイブファイルのライブラリを管理する手間が省けた。まぁお気に入りのリンクは増えてしまったけど、ファイルサイズが50Mバイト以下ならインターネットが僕のファイルシステムの一部という思いである。

あとはそう音楽や映像のストリーミングを気兼ねなく見られるようになったんだけど、日本ってこの分野が非常に立ち遅れているというのを痛感する。ろくなコンテンツがありゃしないし、通信回線もサーバの能力も不足しているところが多くてまともに見られたものじゃない。ａｖｅｘｎｅｔＴＶには早々に接続してみたものの、現状におけるストリーミングの品質は申し分ないんだけど、結局すべてavexサウンドなわけで僕はもう若くないのかなぁ2ターン目で飽きてしまった。で、行き着いた先は米国のインターネットラジオ局。それこそ24時間垂れ流すことを考えるとバラエティに富んだものでないと僕にはつらい。まぁ好みの問題。それにしても僕の環境、いやたぶん多くの人がそう感じていると思うけど、日本国内のサイトより米国のサイトのほうが、接続品質が総じていいんだよなあ。うーむ、何かをやろうと思ってから決裁承認され実行に移るまでのスピードの差なんだろうね、きっと。

そういえばつい最近、民放3社がブロードバンド向け映像コンテンツ配信事業に共同進出するというニュースが流れていた。正確にはその事業を立ち上げるための準備会の設立である。まぁ早い話が、本腰を入れるとなると莫大な資本投下が必要なわけだけど、ブロードバンド向けサービスが金になるかどうかまだわからないし、さらにCSとかデジタルBSとかに手を出しちゃったもののその視聴者はどん詰まりで金もない。でも金になるとわかったときに遅れをとりたくない、だからツバをつけておこう、みんなで共同研究というわけだ。まぁ実際問題として、在京キー局レベルが映像コンテンツ配信を行うとなると、それなりの品質で、10万、100万を超えるユーザーの同時アクセスをさばくことが求められるし、既存の技術で著作権問題なども含めて大丈夫なのかどうかも検討する必要があることも事実なんだけど。それにしてもデジタルBSがさっぱりだからなぁ。

## せっかくADSLを導入したのに

ADSLを中心に快適な日々をすごしていたのだけれど、近々に引越しをすることになってしまった。あーあ、すべ

て最初からやり直し、リセッション、いやリセットである。

さまざまな機関に登録している住所の変更って何でこんなに面倒なんだろう。だいたいアクティブに利用しているもの以外覚えていない。大丈夫かな。でも少し便利だなと思えたのは、いくつかの銀行などはインターネット経由で申請ができるようになっていたことだ。最終的な確認のために送付された用紙にサインと押印が必要な場合が多いけど、それでもかなり楽だ。

さて一番の問題はADSLである。実は引越しといっても200ｍくらいの移動で、同じ町内だったりする。もちろんその建物はメタル回線。そしてNTTの収容局には、より近づくことになる。だから変更届を申請すれば、これ自体は特に問題は何もないのだけれど、別の系からの問題が浮上してきた。

その建物の管理会社が昨今のインターネットブームにあやかって、いくつかの提案をしているというのだ。ひとつはケーブルテレビの導入。このケーブルテレビ会社はインターネット接続と電話のサービスも提供している。もうひとつはHomePNAの導入。既存の建物内の電話配線を利用したインターネット接続である。実は両者のどちらかを選択しようとしているのではなく、両方導入しようとしているらしい。本来ならケーブルテレビの導入ですべてがまかなえるのだが、HomePNAは管理会社の関連会社によるものだから、どうしても導入させたいという思惑があるようだ。性能と料金は、ケーブルのほうが、下り2Mバイト／秒、上り128Kバイト／秒で、単体の料金は月6500円（初期工事費1万5000円）、HomePNAは、各室内は1.0か1.1の規格で1Mバイト／秒、上流への接続は建物全体で最大でも2Mバイト／秒、下手をすると1Mとか0.5Mバイト／秒の回線しか提供しないらしい（加入戸数が70でようやく0.5Mとか言っていた）。しかもその程度の回線速度しかないにもかかわらず、無線の設備を追加して周辺にサービスを提供するってなことも言っていた。価格は月2980から3990円程度で、初期費用は2万5000円程度。

少なくとも僕にとっては、ケーブルのほうはまだしも、このHomePNAは昨今の通信環境を考えるといただけない。せめて10M、あわよくば100Mバイト／秒にしてくれないと。本当は理事会なんぞに行きたきゃないんだけど、さすがに管理会社のいいようにされてしまうのは嫌なので、出席することにしている。

というわけで今現在、我が家の通信環境がどうなるか、不透明な状況にある。HomePNAとADSLは規格的に同居可能らしい（想定する通信の距離が違うため使用する周波数帯域も異なる。HomePNAは5.5MHzから9.5MHzと高い、その代わり数100mと通信可能距離が狭い）が、MDFにどういう変更を加えるかわからないので、本当に不透明なのだ。はてさて何がいいんだろうねぇ。しばらくアナログモデムに逆戻りかも…

そういえば、最近インターネットマンションとか銘打っているものもあるけど、実態はしょぼいんだろうなぁ。まあアナログモデムを使っていた人から見れば、大容量回線なのかもしれないけれども。

さて、ちょっと余談が過ぎたかな。でもこのところネットワークの下位層の話に終始していたので、少し箸休め。ついでにここ最近気になっていたことをいくつか書き留めておこう。

## ramfs（メモリ内ファイルシステム）

チューニングというか、パフォーマンスアップのためというか、ボトルネックを少しでも軽減して、システムを快適に動作させるための手法として、以前RAMディスクを/tmpにマウントして使う話をしたけど、このところramfsが安定しているようなので、こちらを紹介しておこうと思う。

ramfsはその名のとおりメインメモリ内に構築されるファイルシステムで、その使用する領域は、ファイルキャッシュのメモリ空間の一部として動的に切り出されるようになっている。このramfsの一番のポイントは、RAMディスクと違い領域サイズが固定ではなく、そのファイルシステムにファイルデータが書き込まれれば自動的に拡大され、削除すればこれまた自動的に解放されることにある。つまりメモリの利用効率が非常に優れている。

デフォルトでは搭載する物理メモリサイズの半分まで拡張できるようになっている。たとえば、256Mバイトのメモリを搭載しているマシンだと、128Mバイトまで利用できる。

ただ、このramfsはカーネル空間で動作し、消費するメモリ領域もその空間から割り当てることになるので、同様の仕組みをユーザープロセス空間に実装するのと比較して、性能が良い（低下しない）半面、その空間サイズの影響を受けてしまう。一般的な32ビットx86CPUの場合、通常各プロセスのアドレス空間が4Gバイトってことはよく知られていると思う。

Linuxシステムでは、この空間をユーザー3Gバイト／カーネル1Gバイトとしてレイアウトしている。つまり

0xFFFFFFFF中の0xC0000000以降がカーネル空間で、ファイルキャッシュやバッファを始め、その他諸々のカーネルドライバが使用するメモリはその空間にマッピングされていなければならない。ramfsが使用するメモリはファイルキャッシュの一部としてカウントされるため、それ相応の制限が生じるというわけだ。

ちなみにユーザー空間だともっと多くのメモリ領域を利用できるかというと、そう単純なものでもない。先ほどユーザー空間は3Gバイトと言ったが、この空間は完全な自由レイアウトではなく、どのような目的で利用するかがアドレスごとに決まっている。たとえば、実行プログラムは0x08048000からの割り当て、それに伴う共有ライブラリが0x40000000からと決められていて、ある程度自由に使えるヒープメモリは0x80000000以降、スタックメモリは0xBFFFFFFFから下に向かって拡大されるようになっている。結局本当に自由な空間は3Gバイトのうち1Gバイト以下でしかない。

そのためメモリを多く搭載しても、プロセス空間にデータファイルをマッピングして処理したい場合、たとえばデータベースサーバなどでは問題になっている。このようなことは、Linuxに限ったことではなく、32ビットアドレッシングのすべてのOSに共通した問題だ。Linuxでは、この問題を解決するためにアドレス空間のレイアウト変更やCPUのPAE機能を使用するHigh Memoryサポートと呼ばれる機能が実装されている。まぁ一般ユーザーには縁遠い話で、必要なら64ビットCPU、64ビットサポートLinuxという選択肢がすでに実用化段階にあるから気にしなくてもいいのだけれども。

ちょっと話がそれてしまったが、そもそもramfsはファイルシステム上に一時／中間ファイルを作成するように実装されているプログラムの不要な待ち時間を軽減するのが主目的であり、電源が落ちたり、システムがフリーズしたりすれば、跡形もなく消滅してしまう運命なので、細心の注意を払って利用しないといけない。

その昔、DOSが主流だったころ、フロッピーディスクやハードディスクの性能や容量の問題もあって、FEPの辞書ファイルをシステムの起動時に作成したRAMディスクにコピーしたり、あらかじめ圧縮しておいたプログラムやデータをRAMディスクに展開して実行したりしていた人は少なくないと思う。しかしながら、現在のようにファイルキャッシュ、バッファが優秀な状況においては、そんな用途も無用で無駄だ。まぁフロッピーディスク1枚に収まるLinuxシステム、とかなら役に立つと思うけど。

つまるところ一般的にramfsが有効なのは、/tmpと特定のアプリケーションサーバプログラムに依存したキャッシュディレクトリであろう。/var/tmpや/usr/tmpを含めてもいいけど、いくつかのviクローンはそこにリカバリーファイルを作成するので確認したほうがいいだろう。なお今僕が使っているvim6.0ak ALPHA版では、編集ファイルのカレントディレクトリにスワップファイルを作成するのがデフォルトの動作となっている。実はここ最近の考え方としては、tmpなどのテンポラリスペースはセキュリティ的に問題があり、消し忘れなどによる不注意も起こりやすいので、ほどほどにしか利用されていない。だから/tmpをメモリファイルシステムにしたからといって、必ずしも効果があるというものでもないのである。

あと物理メモリが少ない環境、フリーのページメモリに余裕がなく、メモリのやりくりが頻発するような環境では利用しないほうがいい。あきらかに性能が低下する。アプリケーションやシステムがページメモリを必要とした時、通常のファイルキャッシュではパージすることでメモリのやり繰りができるけど、ramfsで消費している領域は、勝手にパージするわけにもいかないし、この領域は当然ながらページアウトの対象でもないからだ。でもまぁメモリが32Mバイトの環境であっても、変な使い方をしなければ問題ないと思われるけれども。

では実際に設定することにしよう。

まずはカーネル構成の設定。ファイルシステムの項目で“Simple RAM-based file system support”を有効にする。モジュール構成にすることも可能だ。モジュール構成にしたとしてもモジュール設定ファイルなどの修正は必要ない。またドライバモジュールのロードは、マウントコマンドの実行時に自動的に行われるので気楽に好きなほうを選択しても問題はない。

```
[File systems  --->]
  <*> Simple RAM-based file system support
```

次にマウントの設定。まぁ/etc/fstabにそのエントリを追加するわけだけど、テストもしないでfstabに書き加えるのは危ないので、実際には最後に行ってほしい。

**●/etc/fstabに追加**

```
none  /tmp  ramfs  defaults  0 0
```

ポイントは、接続元とファイルシステムタイプ。接続元はスペシャルファイルじゃないのでnone、ファイルシステ

ムタイプはramfsとする。オプションなどは適当に。

とりあえずテストとしては、マウントコマンドで実際に接続して、ファイルを書き込み、そして削除してみるとよいだろう。マウントコマンドでは次のように実行する。もちろんrootの権限で。

```
# mount -t ramfs none /tmp
```

●接続状態は次のようになる（最後の行）

```
# mount
/dev/ide/host0/bus0/target0/lun0/part1 on / type
ext2 (rw)
none on /proc type proc (rw)
devfs on /dev type devfs (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
none on /tmp type ramfs (rw)
```

●使用領域の状態（AlanCoxパッチカーネルを使用）

```
# df
Filesystem  1k-blocks Used Available Use% Mounted on
/dev/ide/host0/bus0/target0/lun0/part1
       8096152  4815032  2869860  63% /
none    127504        0   127504   0% /tmp
```

なお単にマウントしただけだと/tmpのパーミッションは0755と、上位のディレクトリやコマンドの実行環境設定などの影響を受けて、既存のものとは異なっている。しかし/tmpのパーミッションは1777が一般的だ。これを修正する必要がある。

説明する必要はないかもしれないが、パーミッション0755は所有者以外書き込めない。/tmpの所有者はroot。でも/tmpには通常誰でも書き込め、それぞれの所有のファイルを消せないと困る。というわけで/tmpディレクトリのパーミッションは1777となる。

最初の1はSticky bitといって、ファイルだとプログラムのテキスト部分のスワップを許可するという古めかしい指定だったのだが今となっては意味がない。しかし現在では、ディレクトリにこのビットを立てると、ファイルの所有者のみに削除の権利を与えるという指定になる。777という指定だけの場合、所有者に関係なく誰でも自由にファイルを削除できてしまう。それを防止するための指定である。特に/tmpなどのようなパブリックスペースの場合、この指定を忘れないようにしたい。

●パーミッションの確認

```
# stat tmp
  File: "tmp"
  Size: 0          Blocks: 0          Directory
Access: (0755/drwxr-xr-x)          Uid: (    0/
root)  Gid: (    0/    root)
```

●パーミッションの設定

```
chmod 1777 /tmp
```

/etc/fstabにマウントの指定を行った場合は、rc.sysinitかrc.localあたり（ファイルの最後でいい、マルチユーザー環境が実行される前であれば大丈夫）に“chmod 1777 /tmp”を追加しておくこと。ただし、初期化ファイル内で実行されるプログラムに、root以外のユーザーで処理されるものがあり、それが/tmpに何かしら書き込みを行う必要がある場合は、きちんとrc.sysinitのローカルファイルシステムのマウントの直後に追加する必要がある。僕のRedHat7.1ベースの/etc/rc.d/rc.sysinitファイルの場合、だいたい526行目前後がそれに該当する。今のところマウントコマンドのオプション設定では、通常のディレクトリの場合、パーミッションを指定できないようだ。

■彷徨えるファイルシステム

以前、僕はExt2の次はReiserfsだと思い、使用しているという話をしたけど、それは一時撤回します。このReiserfs、NFSとの競合問題がなかなか解決せず、デッドロックを繰り返すばかりなので、使わないことにしたのだ。で、後継をIBMのJFSとSGIのXFSの2つに見据えたんだけど、どっちもまだまだって感じで、さらにコード以外の部分の問題もあったりして、煮え切らない。

まぁJFSはベータ3で、XFSはリリース1.0だからXFSの開発ツリーlinux-2.4-xfsを暇しているマシンでテスト運用している。悪くはないんだけど、しばらくはいろんなプロジェクトの動向を見てからじゃないと、30Gバイト以上のファイルの移転はしたくないなぁという感じ。

■GCC 3.0

やっとGCC 3.0、リリースされましたね。まぁ当面は注意して複数のコンパイラバージョンを使い分けたほうがいいと思う。僕の環境では、5月の初旬あたりからLinuxカーネルはGCC 3.0でコンパイルできるようになり、特に問題もなく稼動しているけど、glibcではコンパイラの内部エラーが発生するなど、まだ怪しい。特にC++のコードは注意が必要だ。でもまぁ使ってみるに限るでしょ。

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16回**
**【常時接続】（じょうじせつぞく）**
**【ブロードバンド】（ぶろーどばんど）**
**【FTTH】（えふ・てぃー・てぃー・えっち）**
**【フレッツ・ISDN】（ふれっつ・いすどん）**
**【ADSL】（えいでぃーえすえる）**

## 常時接続

**【じょうじせつぞく】**

～がべこれ劇場「昼前の情事・接続のしたたり」～

世田谷の瀟洒なマンション。軽く“行ってきます”のキスを交わし、夫がいつものように会社へ出かけたあと。1人で部屋に残された美津子は、いつものように密やかで淫靡な儀式の準備を始めた。儀式。それは、良き妻を演じる日常に疲れた美津子が、唯一のなぐさみとして覚えた背徳だった。

絶句

美津子は高まる胸の鼓動に恥じらいを感じていた。しかし、美津子にはもうがまんができなかった。以前は、夜も11時を過ぎてから、愛する夫と二人でつなげあっていたこともあった。それは若い二人にふさわしい、ささやかな行為であった。最近はすっかりごぶさただ。そう。まだ陽も高いうちから営む“情事”の味を覚えた美津子には、夫との行為は満足できるものではなくなってしまったのだ。

ふるえる指で、美津子はパソコンのスイッチを入れた。「へへへ。奥さん。今日もつなげるのかい」美津子には、パソコンのハードディスクの回転音が、あたかもそう言っているかのように聞こえた。「奥さんも好きだねえ。オレもたくさんの女と情事してきたけど、あんたほど好きな女は初めてだよ。ひひひ」美津子の罪悪感を映すかのように、今日のパソコンは饒舌だった。彼女は、その口をふさごうとしなやかな指でパソコンのキーボードをさすり、息を荒らげながらつぶやく。「……あなたはだまって私を愉しませてくれればいいのよ」いつになく積極的な美津子に、パソコンはしばし驚きながらも、やがて女の心の底のどす黒い欲望を察したか、ニヤリと笑いながらBIOSのイニシャライズを終えた。「さあ、奥さん。お望みどおり愉しませてやるよ！」

美津子は、これから始まる情事を想像してうっとりと目を閉じた。そのとき、パソコンにつながったADSLモデムのリンクランプが点灯し、ブロードバンドの荒々しいパケットの奔流が美津子の体内に流れ込むのであった。「ああ！すごい！」「はあ、はあ。どうだい奥さん。いいかい？」

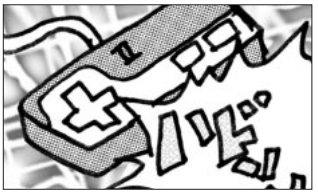
夫との夜には想像もしなかったような巨大なパケットが、美津子のポートを通り過ぎていく。TCPセッションが外から内へ、内から外へ……。快楽の波に埋もれる美津子に、パソコンがささやいた。「奥さん。今日はちょっと違った趣向で行こうか」えっ？　なに？　と聞き返す間もなく、パソコンの毒牙が美津子を襲った。「そ、そこは違う！」

美津子が初めて味わうポートスキャンであった。背徳に黒光りするパケットが、美津子のふだんとは違うポートを襲ったのだ。それも見境なく。「はあはあ。奥さん。なかなかいいぜ」

当初はあった苦痛も、やがて快感に変わりつつあった。美津子は新たに開かれた世界に身を任せながら、常時の世界にのめりこんでゆく自分を感じていた。そのとき、美津子の脳裏から、純朴な夫の笑顔はすでに消えていたのであった。（つづく）

予告：さらなる刺激を求める美津子は、より激しい情事の世界へと旅立つ。次回「美津子、光の世界へ」（なお、残念ながらLinux magazineにLinux誌初の青少年健全育成条例が適用された場合は、次号が発行できない場合がございますので、あらかじめご了承ください）。

## ブロードバンド
**【ぶろーどばんど】**



2コンに向かって叫ぼう

このあいだ、雑誌のブロードバンド特集を読んでいたら「ブローダーバンド環境はすこぶる快適」などと書いた記事があった。穴を掘って敵を埋めたりとか、ヘリコプターで敵地を攻撃したりするのには、確かに快適そうな環境だと思った。ハドソーン！　と叫ぶと速くなるらしい。

## FTTH
**【えふ・てぃー・てぃー・えっち】**

Fiber To The Homeの略。戦中戦後は大根めし、さつまいもを主食にしていたことから国民の繊維質（fiber）摂取量が多く「ファイバー大国」と呼ばれた日本も、食習慣の西欧化により、日増しに繊維を摂る機会が少なくなってきているといわれる。2000年度厚生省白書によれば、繊維摂取習慣が薄れることによって、大腸ガンなど消化器官系疾患の罹患率が上昇、これが医療費の増大をまねき、国家財政逼迫の原因となっている。90年代の“失われた10年”は、便秘に悩む日本国民がトイレで無駄に苦しんだ10年でもあった（日本経済的新聞社刊『日本が臭い』より）。

これを嘆き、かつての名宰相・森嘉朗が実現に腐心したのが、いわゆる「Fiber To The Home」（繊維質を各家庭に）である。森内閣では、FTTH実現のため「eJAPAN構想」（“いい排泄を日本に”構想の略）を提言。2005年度までに1人あたり1メガトンの繊維食品摂取を義務づけるほか、すべての家庭へのウォシュレット設置を目指すという野心的な目標を掲げた。この提言には、首相の私的諮問機関に参加していたソニー・出井会長の影響が少なからずあったとされている（当時、ソニーはアグリ部門、トイレタリー市場への進出が噂されていた）。

eJAPAN構想は、全国の下水道普及率向上に貢献することが期待され、年間数百億円の経済効果によりゼネコン各社だのみの景気回復にも資すると見込まれていた。しかし、パーマと毛染めに失敗した永田町の変人・小泉純一郎の登場により、森首相は退陣。日本のFTTH実現はさらに遅れることになる。

## フレッツ・ISDN
**【ふれっつ・いすどん】**

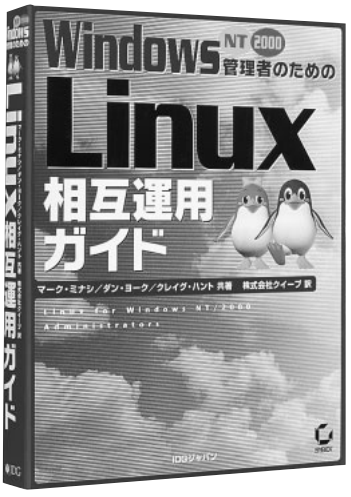
チクビームにあらず

常時戦隊ブロバンジャーに一度は破れた課金獣イスドン。だが、ミカカ界の神官・ミヤーヅの法力により新たに幹部フレッツ獣として蘇った。それがフレッツ・イスドンである。そのパワーは強大であり、ミカカ界幹部・慎吾ママの“おは攻撃”とともにブロバンジャーを苦しめた。なかでも東京方面担当のブロバンジャーレッド（実名はあえて秘す）は、イスドンの巧妙な攻撃により、財務面での弱点を露呈してしまったのであった（常時戦隊ブロバンジャー第24話「暁に死す？！　銅線の戦士レッド」）。だが、慎吾ママの必殺技を「おはスタ」からのパクリと見抜いたブロバンジャーは、渾身のDoSアタックを放ちイスドンの野望をみごと打ち砕いた（第27話「くらえ！　勇気のDoSアタック」）。とはいえ、あれが最後のイスドンとは思えない。人類が同じ過ちを繰り返す限り、第2、第3のイスドンが……。

## ADSL
**【えいでぃーえすえる】**

Asymmetric Digital Subscriber Lineの略。日本語に訳すと“非対称デジタル加入者線”を意味し、金持ちのところには高速な回線を、貧乏人のところには低速の回線を非対称に提供する差別的なデジタル回線サービスのことだと思われる。くそっ。それでうちのADSLは790kbpsしか出ないのか。非対称なADSLに対し、対称なSymmetric Digital Subscriber Line（SDSL）という回線サービスも存在するが、こちらは金持ちにしか利用できないような高額の利用料金が課されるのは言うまでもない。結局、貧乏人は悲しい思いをするのである。担当編集は飯もおごってくれないし。「ところで今月のこのコーナー、ぜんぜんLinuxの話が出てこなかったんですけど。いちおう、これLinux magazineなんで困るんですが」「えっ。そうですか？（なんだ、いたのか）　57行目に2回も“Linux”って出てきてますよ？」「……そういう問題じゃないだろ。おい」

# Books




## Windows管理者のためのLinux相互運用ガイド

**マーク・ミナシ、ダン・ヨーク、クレイグ・ハント共著／株式会社クイープ 訳**
**IDGジャパン**
**B5変形判／568ページ　本体価格 4286円**

Windowsユーザー向けのLinux解説書は、これまでにも何冊か出版されているが、多くはネットワークでの相互運用に関するものであった。本書は、それとは少し趣きが違って、Linuxの基本操作から、サーバ構築の基礎知識に大部分の章をさいている。Windows（NT／2000）とLinuxの相互運用について具体的に解説しているのは、全10章のうち第9章だけである。では「看板に偽りあり」なのかというと、そうではなくて、随所でWindowsを引き合いに出してLinuxを理解するための比較対象としている。Window NT／2000に習熟している管理者のためのLinux解説書といった構成になっているのだ。

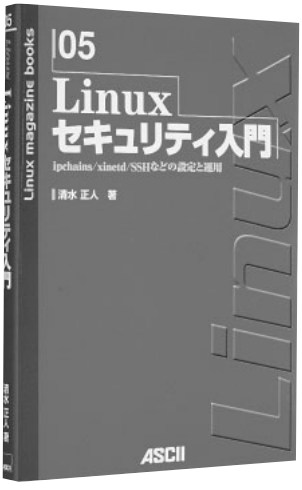
前述のようにLinuxの基礎からサーバ構築までひと通り解説されており、インストール、XFree86の設定、基本的なコマンドの使用法なども、かなり詳しく取り上げられている。管理者だけでなく、Windowsユーザーのための「Linux入門書」としても使えそうな1冊である。

## Red Hat Linux 7.1 オフィシャルマニュアル

**Red Hat, Inc. 著／レッドハット株式会社 訳**
**インプレス**
**B5変形判／808ページ／CD-ROM4枚付き　本体価格 6200円**

形態は書籍となっているが、本書はRed Hat Linux 7.1の市販パッケージのひとつとも捉えられる。インストールCDが2枚に、ソースCDとデキュメントCDが各1枚が付いていて、さらにユーザー登録から15日間有効なインストールサポートが受けられるのだ。「主役」であるオフィシャルマニュアルは、4つのオフィシャルガイド（インストールガイド、入門ガイド、リファレンスガイド、カスタマイズガイド）をまとめて書籍化したもの。アプリケーションやフォントの代わりに印刷されたマニュアルが付いてくるというわけである。

実をいうとオフィシャルガイドはHTML形式でドキュメントCDにも収録されている（つまり通常のパッケージにも含まれている）のだが、やはり本の形でまとまっているほうが作業中に参照する際など、なにかと重宝するはず。これからRed Hat Linux 7.1の購入を考えている方は、本書も候補のひとつとして検討してみてはいかがだろうか。


## Linuxセキュリティ入門

**清水正人 著**
**アスキー**
**A5判／204ページ　本体価格 1800円**

Linux magazine Booksの第5弾はセキュリティ本。セキュリティについて体系的に捉えられるよう、幅広いトピックを網羅する構成になっている。ひと通り内容を紹介しておくと、パケットフィルタリングの設定方法（ipchainsおよびiptables）、パスワードなどLinuxサーバ側での基本的なセキュリティ設定、inetd／xinetdによるアクセス制限、SSHを使った安全な通信、各種セキュリティツールの導入と基本設定となっている。ツールとしては、Nessus、swatch、Tripwireを取り上げている。

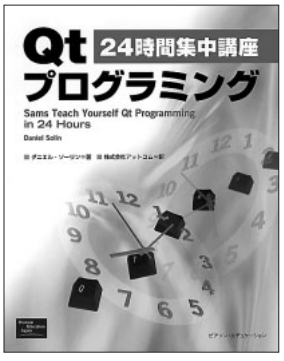
本誌でも5月号の特集を始め、カーネル2.4の新しいセキュリティ機能の紹介など、折に触れてセキュリティに関連する話題を取り上げてきたが、重要なテーマだけにじっくりと腰を据えて取り組みたいという方も多いことと思う。まずは、本書でLinuxで実現できるセキュリィ対策の基礎を学んでみてはいかがだろうか。


## リナックスの革命 ― ハッカー倫理とネット社会の精神

**ペッカ・ヒマネン 著／安原和見、山形浩生 訳**
**河出書房新社**
**四六判／256ページ** **本体価格 1600円**

コンピュータ・ハッカー達の仕事ぶりは、芸術家や音楽家などの創作活動に似た面があると思う。表現手段こそ違うが、その原動力となっているのは一種の「情熱」である。また彼らは、基本的に自らの仕事の成果に対して経済的な見返りを求めてはいないし、時間的な制約に縛られることもない。そこに見られる価値観は、近代以降の資本主義社会で育まれてきた勤労の精神とか職業倫理とはまったく別のものだ。本書では、こうしたハッカー達の精神をより大きな視点で捉え、歴史的な意味や社会への影響といった面も考察している。サブタイトル（原書ではメインタイトル）は、資本主義社会における労働の価値と宗教的倫理観の相関について考察したMax Weberの有名な論文『プロテスタンティズムの倫理と資本主義の精神』のもじり。なお、著者はフィンランド生まれでヘルシンキ大学の教授である。その縁もあってか、Linus Torvaldsが本書のプロローグを執筆している。

## 24時間集中講座 Qtプログラミング



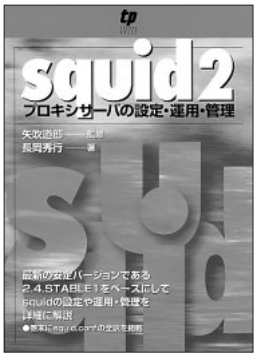

ダニエル・ソーリン 著
株式会社アットコム 訳

ピアソン・エデュケーション

**B5変形判／448ページ**

本体価格 3800円

## squid 2 プロキシサーバの設定・運用・管理




矢吹道朗 監修
長岡秀行 著

テクノプレス

**A5判／272ページ**

本体価格 2600円

## OpenSSH セキュリティ管理ガイド


新山祐介／春山征吾 著

秀和システム

**A5判／376ページ**

本体価格 2500円

## Rubyを256倍+使うための本 赤玉制覇編・場外乱闘編


Rubyを256倍使う会 著

アスキー

**B5判／272・264ページ**

本体価格 1800円

# 読者の声

俺にもいわせろ！

モルモットを飼い始めて、そろそろ2年になります。モルモット君が家に来たときは手のひらに収まるぐらいの大きさで、ものすごいスピードで部屋の中を駆け回っていました。あれから2年、すくすく育った彼は、担当の友人たちに「これって、ウサギ？」と言われてしまうほどの大きさになり、動きもすっかり鈍くなってしまって、本当にウサギっぽい……。実は、ウサギの突然変異なんじゃないか？　と最近疑っている担当です。

### ADSL常時接続への道

**Linuxを始めて約1年。主に、業務で使用しているのですが、やはり自宅にも常時接続サーバを立てたくなりました。そこでADSLサービスが利用できるかどうか調べてみたら同一市内でも私が住んでいる市の西部ではADSLがまだ使えませんでした。CATVはあるのですが、プライベートアドレスをDHCPで割り振る方式なので外部に公開できません。今さらフレッツ・ISDNもどうかと思うし……。**

**中途半端に田舎に住んでいる人は、私と同じような悩みを持っているかもしれませんね。結局ADSLがカバーされるまで待つことにしました。夏くらいにはカバーされるそうです。**

**（松嶋喬史さん）**

**毎号楽しみにしてます ついにADSLの導入にこぎつけました。来週（6月の上旬）にサービス開始です。**

**Linux機（といってもハードディスクの取り替えですぐ変身！）への接続はもう少しこなれてから。いまは、一応「速くなったこと」と「いつでも使える」点を嫁さんにアピールすることに専念したいです。→　初期費用がオイラの財布か家計かの分かれ目なもんで（必死）。**

**Linuxの接続はバックナンバーを見てがんばります。上手く設定できることを祈っています！（結局はLinux magazineと神頼みです）これからもよろしくお願いいたします！**

**（金田栄治さん）**

こちらこそ、よろしくお願いいたします。

Linux magazineの読者の方の中にもADSLで常時接続されている方が、ずいぶん増えてきたようです。金田さん、奥様へのアピールとインターネット接続は上手くいきましたでしょうか？

Linuxはお金がかからないとはいうものの、趣味には何かとお金と（もちろん、Linux magazine購入費も！？）時間がかかってしまいますよね。パートナーの理解が必須ですね。皆さんはパートナーの理解を得るためにどんな工夫をされているのでしょうか？

夏が待ち遠しいですね。＞　松嶋さん

### RPMパッケージの達人

**7月号から始まったRPMパッケージ開発講座は、よかったです。RPMパッケージを使用する頻度が高い割に、実際のところほとんど理解していませんでしたので、これを機に理解して有効に使いこなせるようになりたいです。**

**（伊藤真和さん）**

新連載「RPMパッケージ開発講座」はご好評いただいております。担当もRPMパッケージの必要最低限な部分しか覚えていないので、「RPMパッケージ開発講座」はとても助かります。でも、担当は欲深くないのでRPMパッケージは使いこなせるようにならなくてもいいです。「RPMパッケージ開発講座」が辞書となって担当を支えてくれるなら……。え？　甘い？

### 初心者の苦悩

**Power Linux Userになるにはどのような方法がよいのでしょうか？　現在市販されている本等では初級クラスの本と、上級クラスの本に集中しているような気がしています。（私の本の探し方が悪いのかも ^^;　）結局Linuxを PCにインストールしてもそれから何をしたらよいのやら、どの方向に向かっていっていいのやら、わからなくなり、ついにはLinux PCの電源すら入れなくなってしまうのではないのかという不安が募ります。**

**できれば、このようなLinux初心者への愛の手をLinux magazineを通して**

**さしのべてください。**

**（道を忘れたLinux初心者さん）**

「Linuxを使えるようになりたい」から「こんなことをLinuxでやってみたい」というように、具体的な目標を作れば、道が見えるのではないのでしょうか。具体的な目標があってこそやる気が持続するものです。

なぁんて書いちゃってる担当も、愛の手が欲しいぃ〜と助けを求めて止まない一人です。だって、何か始める度に、第一歩からぬかるみにはまってしまうんですもん。やっとの思いで抜け出せたかと思うと、金網のはずれた排水溝にタイミングよく落ちて、ようやく抜け出せたときには、大幅に進路がずれていることに気づく……。始めからくたくたです。でも最初のこの時期を抜ければ、あとはしばらくスイスイいけますよ。きっとね。いや、そう信じましょう。

Linux magazineでは、初心者の方にもわかりやすい誌面づくりを心がけています。これからもぜひ、参考にしてくださいね。

## SMBとNFSの間柄

**Linux同士でのディスクの共有はどうやったらいいのでしょう？　どうやってもできないのです……。WindowsとはSambaで共有ができたのですが、どうしてもLinuxからはサーバが見えません。LinuxとWindowsばかりではなくて、Linux同士での活用の“基本的”な（たぶんわかり切っている、みんな知っている）ことも紹介してくれると助かるのですが……。簡単すぎますか？　でもわからない人間もここにはいます。**

**（坪井邦仁さん）**

いえいえ、疑問に思われる気持ちわかります。WindowsとLinuxのファイル共有のことについては、いろいろ記事があるのに、Linux同士ということになると、記事はほとんど見当たらないですものね。

Linux（Linuxに限らずUNIX系のOSはすべて）は、ネットワーク上でNFSというプロトコルを使っています。一方Windowsは、SMBというプロトコルを使っています。両者が会話をするとき、LinuxでSMBプロトコルを使えるようにするためのソフトがSambaです。反対に、NFSプロトコルをWindowsで使えるようにようにするためのソフトもあります。しかし、現在はLinuxがWindowsに合わせる形でSambaを使う方が一般的です（研究所などUNIX系のOSが多く存在する環境では、WindowsがNFSに合わせることも多いようです）。

ほかには、Windowsでも、Linuxでも気軽に使えるFTPを使うというのもよいでしょう。いろいろと試してみてはいかがでしょうか？

## ピラニア編集部？

**Linux magazine for beginners購入しました。FMVのDESKPOWERに、入れました。でもCOM1を認識してくれませんでした。OSSの設定も失敗しました。初心者を脱皮できないまま、Linuxを使いこなすという夢が潰えてしまった（とほほほ）。こうなったらプレインストール機を入手しよう！**

**ということで、編集部に転がってる（うっ、失言。カットして）動作確認用の古い機種をゆずってもらえないでしょうか？**

**（千葉康広さん）**

「Linux magazine for beginners」をご購入いただきありがとうございます。その後、設定は上手くいきましたでしょうか？　編集部に転がっているマシンですか……。ごめんなさい！　編集部に転がっているマシンにまともなものはないんです。なぜなら、古いマシンであろうと、新しいマシンであろうと、役目が終わったマシンは、お腹をすかせたピラニアがいる川に、放り投げた肉の塊のように、あっという間に各編集部員のマシンの中に、組み込まれてしまうからなのです。Linux magazine編集部の特性というものなのでしょうか？　どうか、ご了承くださいませ。

## 検索システム

**「隠喩としてのコンピュータ」にあるように、これからは大量に溢れるデータの中からいかに的確なデータを探すか、ということがすごく必要になってくると強く思います。Linuxの設定とかで、うまくいかないときWebで検索するけど、必要なデータを見つけるのはすごく難しいし、もどかしいですから。**

**就職活動をしていて、このような検索システムの開発がしたい、とか言っていたのですが、どうなることやら、って感じです。どこの部署に飛ばされるかわかりません。Linus氏みたいに好きなことを会社でもさせてもらってるってすごく羨ましいです。**

**（星野美保さん）**

担当が就職活動をしていたときに、「社会人になるにあたって、あなたに足りないと思うところは何ですか？」と面接官の方に質問されたので、迷わず「わりきることです」と答えたところ、面接官の方々に大笑いされてしまいました。担当は、大まじめに答えたつもりなんだけどな……。

星野さん、希望の部署に入れるといいですね。違う部署に決まってしまったら、そこは「わりきり」で乗り切ってください。ガンバレ！

## 付録CD-ROMに収録した

# Turbolinux Workstation 7(Monza) Betaのインストール

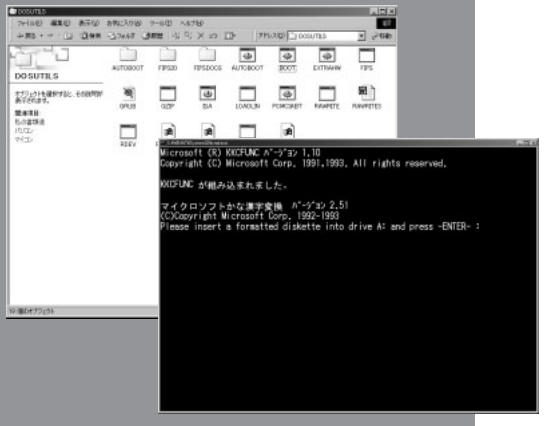
**本誌付録CD-ROMのTurbolinux Workstation 7(Monza) Betaは正式版リリース前のベータ版です。非商用ソフトだけが含まれており、ターボリナックスジャパン株式会社からサポートを受けることはできません。Turbolinux Workstation 7(Monza) Betaのフィードバックは、(http://the.turbolinux.co.jp/bugzilla/)で受け付けております。不具合など発見されましたらぜひお知らせください。**

**また、CD-ROMのメディアに不良があった場合は、お手数ですが住所、氏名を明記のうえ(linux-cd@ml.ascii.co.jp)宛にご連絡くださるようお願いいたします。**

### 1 インストールの前準備

これからTurbolinux Workstation 7(Monza) Betaのインストールを始めます。多くのネットワークカード、ビデオカード、モニタは自動認識されますが、自動認識されない場合に備えて、あらかじめハードウェアのマニュアルなどを用意しておくとインストール中にあわてなくてすみます。

また、レスキュー用のフロッピーを作成するために、空のフロッピーを1枚用意しておいてください。CD-ROMブートができないマシンを使う場合は、インストーラ起動用のフロッピーも用意します。




### 2 ブートフロッピーの作成

CD-ROMブートできないマシンにTurbolinuxをインストールする場合は、以下の手順でブートフロッピーを作成します。

まず、Windowsマシンに付録CD-ROMのDisc 1と空のフロッピーディスクをセットします。次に、エクスプローラを使ってCD-ROMの［DOSUTILS］を開き、［BOOT］をダブルクリックします。するとDOS窓が立ち上がるので、［Enter］をタイプしてインストーラ起動用のブートディスクを作成します。


### 3 インストーラの起動と使用言語の選択

CD-ROMやブートフロッピーをセットしてマシンを起動すると、Turbolinuxのインストーラが起動します。［boot:］プロンプトが表示されたら、［Enter］を押します。しばらくすると、使用する言語の選択場面に切り換わるので、［↓］キーで［Japanese］を選択し、［Tab］キーと［Enter］キーを使って［OK］を押します。しばらくすると、Xを使ったグラフィカルなインストール画面が表示されます。

なお、［boot:］の箇所で［text］と入力して［Enter］を押すと、テキストベースのインストーラが起動します。また、インストーラがグラフィカルな画面の表示に失敗した場合は、自動的にテキストベースのインストール画面が起動します。

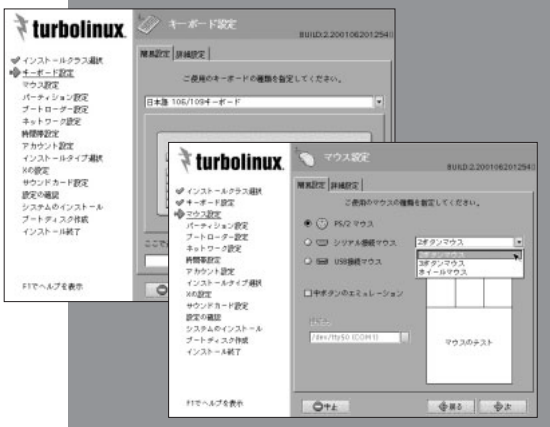
### インストールクラスの選択

本誌に収録したインストーラでは、この場面で［標準インストール］のみを選べます。［Turboインストール］と［アップグレード］を選択できないのは、CD-ROMの不具合ではありません。このまま［次］を押して、次の場面に進みます。


### キーボードとマウスの設定

使用するキーボードタイプをプルダウンメニューで選択します。たいていのキーボードはプルダウンメニューに含まれていますが、Sunのキーボードなど特殊なものを使う場合は、［詳細設定］タブで画面を切り替えて、キーボードタイプを選択します。キーボードを選択したら、［ここで選択した設定をテストしてください］の欄でキーボード入力をテストします。［Shift］キーを押しながら［1］、［2］、［3］を押して、キーボードの刻印どおりの記号が表示されたら、キーボードの設定は成功しているでしょう。

次にマウスを設定します。使用するマウスタイプをプルダウンメニューから選択しましょう。




### パーティション作成ツールの選択

Turbolinuxのインストーラには、パーティションを作成するためのツールが2つ用意されています。［自動パーティション設定］を選択すると、既存パーティションの削除後にTurbolinux用のパーティションが自動で作成されます。

［TFDisk］ではグラフィカルな画面で手動でパーティションを作成します。

ここでは［TFDisk］を選択してインストールを進めます。

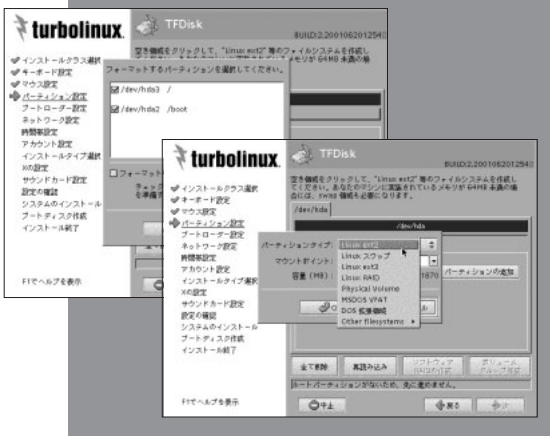

### パーティションの作成

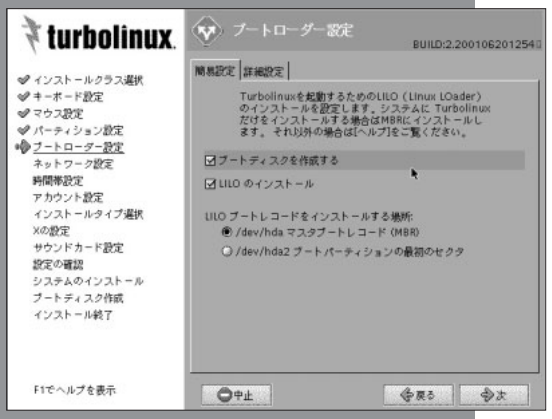
Turbolinuxのインストーラでは、Linuxシステム用パーティション［/］とSwapパーティションのほかに、Linuxカーネルを格納する［/boot］パーティションの作成が推奨されています。以下は、それにしたがって3つのパーティションを作成した例です。

まず、ハードディスクの空き領域をマウスでクリックして［パーティションの追加］を押します。すると、ダイアログ画面が別に開かれるので、

| パーティションタイプ | 容量 | マウントポイント |
|---|---|---|
| Swap | マシンが搭載するメモリの1～2倍 | なし |
| Linux ext2 | 64Mバイト | /boot |
| Linux ext2 | 3Gバイト以上 | / |

の3つのパーティションを作成します。［/］パーティションには、新しく選択項目に追加されたext3ファイルシステムを選択してみるのもおもしろいでしょう（ReiserFSとJFS FSは選択できません）。このあと作成したパーティションをフォーマットします。






## ブートローダLILOの設定

ブートローダLILOのインストール先を選択します。TurbolinuxをWindows 9x／Meと共存させる場合は、［LILOブートレコードをインストールする場所］に［/dev/hda マスターブートレコード（MBR）］を選択します。Windows NT／2000の場合は、［LILOのインストール］からチェックをはずします。

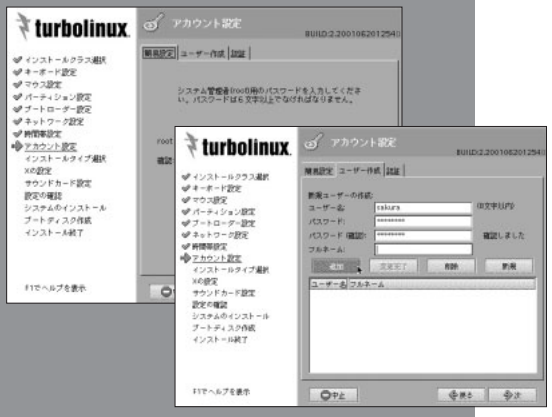
いずれの場合も、必ず［ブートディスクを作成する］をチェックしてください。特にWindows NT／2000と共存させる場合は、あとで作成するブートディスクがないと、Linuxを起動できません。


## ネットワークとタイムゾーンの設定

TurbolinuxをDHCPサーバからネットワーク情報を取得する環境で使う場合は、［DHCPを使用して設定する］を、DHCPサーバのない環境で使う場合は［DHCPを使用して設定する］のチェックをはずして、IPアドレスなどのネットワーク情報を入力します。なお、この場面ではダイヤルアップ接続の設定はできません。

次にタイムゾーンを設定します。デフォルトでは日本時間がセットされているので、たいていのユーザーはタイムゾーンを設定する必要はありません。［システムクロックでUTCを使用］は世界標準時に合わせるためオプションです。Windowsなどと共存させる場合は、このオプションをはずしておくのがよいでしょう。

Turbolinuxのインストーラでは新たにNTPを設定できるようになりました。NTPを使ってマシンの時刻を設定する場合は、［タイムサーバー］タブをクリックして使用するNTPサーバを入力します。

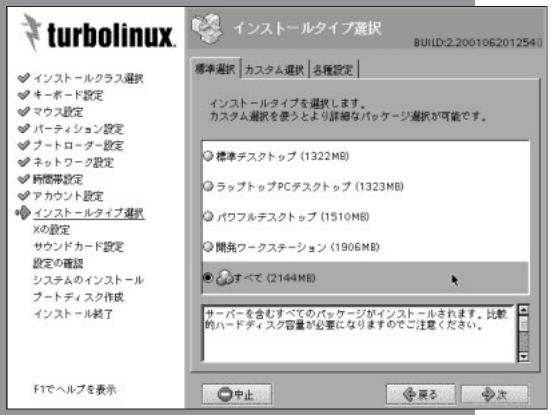



## ユーザーアカウントの作成

ユーザーアカウントの作成

まずはLinuxの管理者用ユーザー（ログイン名はroot）のパスワードを設定します。［rootパスワード］と［確認］に同じパスワードを入力します。［確認］欄の右側に「確認しました」と表示されれば設定は成功しています。

次に、メールの読み書きなどを行う一般ユーザーを作成します。まずは［ユーザー作成］タブをクリックして画面を切り替えます。［ユーザー名］に一般ユーザーのログイン名を入力して、［パスワード］と［パスワード（確認）］に一般ユーザー用のパスワードを入力します。［パスワード（確認）］の右側に「確認しました」と表示されたら［追加］を押し、さらに［次］を押して次に進みます。




## インストールタイプの選択

インストールするパッケージグループを選択します。各グループを選択すると、そのグループの説明が下の欄に表示されます。ここでは、［すべて］を選択してインストールします。［すべて］を選択した場合は、ハードディスクを約2.5Gバイト消費します。

## モニタとXの設定

使用するモニタが自動認識されなかった場合は、リストの中からモニタを選択します。使用するモニタがリストにない場合は、［製造元］から［Generic］を選択して、モニタのマニュアルを参照しつつ［モデル名］から無理のない解像度のものを選択します。液晶モニタの場合は［Laptop Display Panel］から選択します。

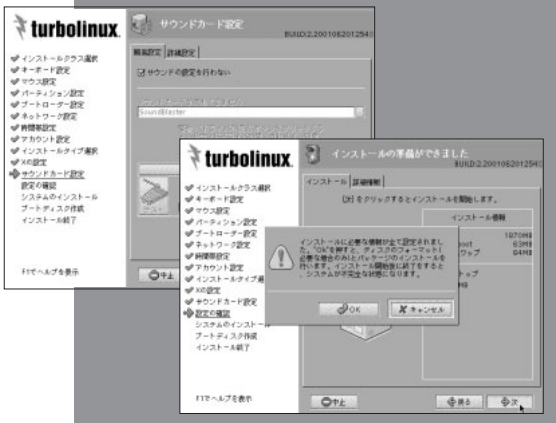
次にXを設定します。ほとんどのビデオカードは自動認識されるでしょう。ひとまずインストーラが自動設定した［デスクトップカラー］と［デスクトップサイズ］のままで、［この設定をテストする］を押してXの表示をテストします。表示に成功したら、Turbolinuxのデスクトップ画面が表示されるので、表示される質問に［はい］と答えてテストを終了します。テストに失敗した場合は、［Xの設定を行わない］をチェックして、インストールを終えてからturboxcfgを使って、じっくりとXを設定してください。

なお、［グラフィカルログインの使用］は、Xの表示テストに成功した場合のみにチェックしてください。［グラフィカルログインの使用］をチェックすると、Linuxの起動後にグラフィカルなログイン画面が表示されます。

## サウンドカードの設定とパッケージインストール前の確認

サウンドカードが自動認識されたら、［Load driver now］を押して、次に進みます。自動認識されない場合は、そのまま次に進みます。

［詳細情報］タブをクリックすると、ここまでの設定情報を確認できます。設定情報を変更する場合は、［戻る］で該当画面まで戻って再設定します。［インストールの準備ができました］で［次］を押すとパッケージインストールの確認を求めるダイアログが表示されます。［OK］を押してインストールを始めましょう。

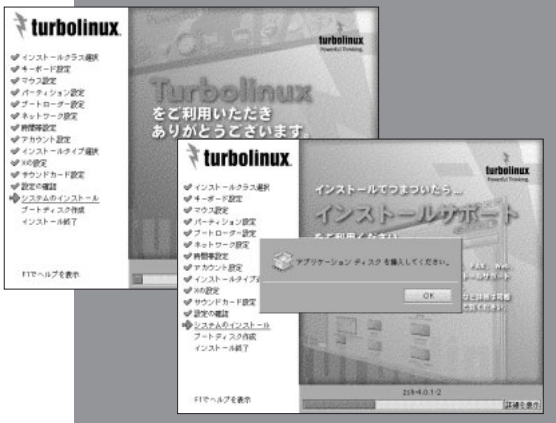


## パッケージのインストール

［インストールタイプの選択］で［すべて］を選択した場合は、パッケージのインストールが終わるまでに、128Mバイトのメモリを搭載したAMD K6-II/350MHzマシンで約60分かかります。画面の下段にある［詳細を表示］を押すと、インストール中のパッケージ名などより細かな情報が表示されます。

パッケージインストールの途中で［アプリケーションディスクを挿入してください］というメッセージが表示されたら、付録CD-ROMのDisc 2をセットしてインストールを続けて下さい。

パッケージのインストールが終了すると、追加パッケージのインストールに進みますが、本誌付録のCD-ROMには追加パッケージを収録していません。［追加パッケージのインストールをスキップする。］をチェックしたまま次に進みます。



## ブートディスクの作成

Linuxをハードディスクから起動できなくなった場合などに備えて、レスキュー用のフロッピーディスクを作成します。空のフロッピーディスクをセットして、［次］を押してください。

フロッピーディスクの作成が終わるとインストールは終わりです。［完了］を押し、CD-ROMとフロッピーディスクを抜いてください。お疲れさまでした。