# Windows＆Linux デュアルOS構成のノートPC IBM ThinkPad i Series 1620 Turbolinux＆Windows デュアルOSキットモデル

ターボリナックス ジャパンは、日本IBMのノートPC「IBM ThinkPad i Series」に、Windows MeとTurbolinuxの2つのOSをインストールし、各OS用に多数のアプリケーションソフトをバンドルした「IBM ThinkPad i Series 1620 Turbolinux＆WindowsデュアルOSキットモデル」を発売した。

これは、ターボリナックスが、日本IBMとアルゴ21の技術協力・販売促進支援により、共同で企画した特別価格のパッケージモデルで、WindowsとLinuxのデュアルOS構成のノートPCを販売するのは国内初だ。価格は24万4800円で、Windows搭載標準モデルと同じ価格帯で提供される。

Turbolinux Workstation日本語版6.0 LEとWindows Meの2つのOSを、ブート時に選択可能なデュアルブート環境を提供する。オフィスソフトとして、Linux用「Applixware Office for Linux 5.0日本語版」とWindows用「Microsoft Office 2000 Personal」の両方が含まれている。

ThinkPad i Series 1620（2661-2CJ）は、B5サイズ、約1.6kgとコンパクトで軽量ながら、低電圧モバイルCeleron 500MHz、64Mバイトメモリ、20Gバイトハードディスク、12.1インチTFT液晶ディスプレイ（1024×768ピクセル）を搭載したノートPC。本キットには、拡張モジュールである「Ultra Base X2」をバンドルしており、24倍速CD-ROMドライブおよび3.5インチフロッピードライブを同モジュールに内蔵している。

ThinkPad iシリーズ1620

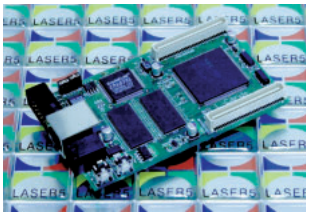
| | |
|---|---|
| 発売日 | 2001年3月1日 |
| 発売 | ターボリナックス ジャパン株式会社 |
| TEL | 03-5766-1660 |
| 価格 | 24万4800円 |
| URL | http://www.turbolinux.co.jp/ |

Hardware　発売日　2001年2月26日

## 名刺サイズの超小型組み込み用Linuxサーバ L-Card＋

URL　http://www.laser5.co.jp/

発売　レーザーファイブ株式会社
TEL　03-5818-6626
価格　3万8600円



レーザーファイブは、64ビットCPUを搭載した名刺サイズ（60mm×91mm）の組み込み用コンピュータ「L-Card＋」を、2月26日より発売した。NEC製最新RISC型CPUでスーパーコンピュータなどにも採用されているMIPSアーキテクチャのVR4181-66（66MHz動作）を搭載した。

16MバイトSDRAM、2MバイトフラッシュROM（簡易モニタ＆ブートプログラム）、RS-232C×1、10BASE-T、コンパクトフラッシュインターフェイス、LED×4個、AC電源5V－3.3Vレギュレータを搭載し、液晶表示信号、タッチパネル信号、ROM拡張、I/O、音声入出力などが拡張コネクタに出力されている。

専用にカスタマイズしたLinux（カーネル2.4-test9）を採用しており、モバイル（情報携帯端末、遠隔無線コントローラ）、システム組み込み（自販機、工作機器、計測機器、データロガー）、情報機器（各種サーバ）などの分野に利用可能となっている。

Hardware　発売日　2001年2月5日

## 持ち運び可能な小型電子メール監視サーバ AddPoint/MailAuditServer

URL　http://www.necsoft.co.jp/

発売　NECソフト株式会社
TEL　03-5569-3399
価格　オープンプライス

NECソフトは、電子メールの情報を監視し、機密情報の漏洩と私用メールの濫用を防止するためのアプライアンスサーバ「AddPoint/MailAuditServer」を、2月5日より発売した。同社の電子メール監視ソフト「GUARDIAN AUDIT V3.0 for Linux」を、アクアリウムコンピュータの小型Linuxサーバ「silver neon」に搭載した。

SMTPに準拠したメールサーバと同一のネットワークセグメントにプラグインするだけで、パケットモニタ方式によって、メールサーバが送受信するメールを監視する。メール内容の分析、機密情報など特定情報の流れの監視、発信記録の保存機能などを備えている。標準装備しているWebベースの専用運営管理ツール「AddPoint Manager」を利用することで、Linuxを意識せずに障害復旧処理などのシステム管理・運用の操作が行える。

Hardware　発売日　2001年2月1日

## IDE RAIDハードディスク内蔵1Uラックマウントサーバ POWERSTEP 1U Rack Server

URL　http://www.amulet.co.jp/

発売　アミュレット株式会社
TEL　03-5295-8418
価格　16万3000円

アミュレットは、Linuxに対応した1Uサイズのラックマウントサーバ「POWERSTEP 1U Rack Server」を2月1日より発売した。

OSを問わずに対応可能なIDE RAIDコントローラを搭載し、標準でRAID 1（ミラーリング）とハードディスクのホットスワップ（自動リビルド）をサポートしている。

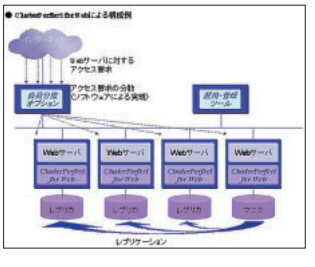
CPUにCeleron 700MHz、メモリ128Mバイト（最大512Mバイト）、20GバイトIDEハードディスクを2台搭載し、ミラーリングで使用する。3.5インチフロッピードライブと24倍速CD-ROMはスリムタイプを採用している。マザーボードは、S2420（Intel 810Eチップセット）で、オンボードに、10/100BASE-Tネットワークコントローラを搭載している。FSB 66MHz／100MHz／133MHzに対応し、PentiumIIIへのアップグレードも可能。

Software　発売日　2001年2月1日

## Webサーバの性能と可用性を高めるクラスタソフト DNCWARE ClusterPerfect for Web

URL　http://www3.toshiba.co.jp/cn3/cluster/index_j.htm

発売　株式会社東芝
TEL　03-3457-2725
価格　64万円（2ノード）～



東芝は、ネットワーク負荷分散クラスタソフトウェア「DNCWARE ClusterPerfect for Web」を2月1日より発売した。Webサーバに対するアクセス要求を複数のWebサーバに分散させる負荷分散機能と、マスタのWebサーバ（マスタサイト）から他のサーバ（レプリカサイト）にコンテンツを自動配布するコンテンツの同期の機能を備えている。

最大256台のサーバでのクラスタ構成に対応し、負荷分散には、サーバの処理能力に応じた重み付けが可能なラウンドロビン方式を採用。同一クライアントからのアクセスを常に同じサーバに割り当てることもできる。オンライン中にWebサーバの追加／削除などの構成変更が行える。

JavaベースのGUIによる運用管理ツールで、クラスタを構成するWebサーバを集中管理できる。Linux、Solaris、Windows NT／2000のマルチプラットフォームに対応し、異なるOSが混在したWebサーバクラスタを構築可能。

Software　　発売日　2001年2月26日

## eビジネス向けLinux版エンタープライズRDBMSの最新版 Oracle8i Enterprise Edition R8.1.7 for Linux

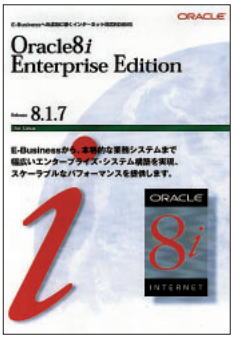
発売　日本オラクル株式会社
TEL　03-3511-5331
価格　120万円（10指名ユーザー）～

URL　http://www.oracle.co.jp/

日本オラクルは、Linux版RDBMSの最新リリース「Oracle8i Enterprise Edition Release 8.1.7 for Linux」を2月26日より発売した。

Java、XMLの開発・実行環境を拡張し、インターネットアプリケーションのサポートのため「Oracle HTTP Server Powered by Apache」を組み込んだ。ネットワークセキュリティを強化するため、Triple DES（Data Encryption Standard：データ暗号化標準）とRC4（256ビット）に対応する長い暗号化キーをサポートする有償オプション「Oracle Advanced Security」が用意されている。

また、同時に統合アプリケーションサーバ「Oracle9i Application Server Release 1.0.2 for Linux」も発売した。価格は7万円（10指名ユーザー）から。アクセスの頻繁なWebページを中間層でキャッシュする機能とロードバランシング機能を持った「Oracle Web Cache」を新たに搭載した。Webアプリケーションの繰り返し実行の負荷を低減し、大量なリクエストを効率的に処理することができる。




Software　　発売日　2001年2月23日

## メールサーバ用アンチウイルスソフト アンチウイルスfor Linux

発売　富士マグネディスク株式会社
TEL　0424-81-8222
価格　13万2000円（1～50ユーザー）～

URL　http://www.fujifilm.co.jp/fmd/linux/avl.html

富士マグネディスクは、Red Hat系Linuxに対応したメールサーバ用アンチウイルスソフト「アンチウイルスfor Linux」を、2月23日より発売した。すでに稼働しているLinuxメールサーバにインストールすることで、ウイルスチェック機能を持たすことができる。また、ウイルスチェック専用サーバとして、メールサーバと別に用意することも可能。

ウイルス検出ソフトは、世界的に実績を誇るエフセキュアの「F-SECURE Anti-Virus Linux」を使用。ウイルス定義ファイルは、毎日自動で更新状況をチェックして、常に最新の定義ファイルにアップデートされる。

無償サポートとして、導入に関する1年間3件までのメール／電話によるサポートが含まれ、有償導入コンサルティングとしてメールサーバの構築サポート（10万円より）も用意されている。


Software　　発売日　2001年2月26日

## 顧客とのやり取りを報告書として共有するツール サイボウズ コンタクト4

発売　サイボウズ株式会社
TEL
価格　5万9800円（10ユーザー）～

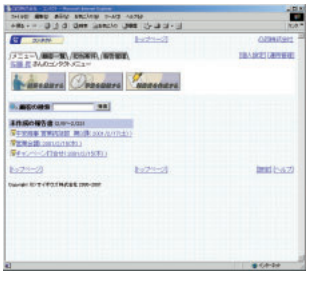
URL　http://cybozu.co.jp/

サイボウズは、顧客とのコンタクトの履歴やさまざまな関係情報を、整理・共有できるツール「サイボウズ コンタクト4」を2月26日に発売した。Webブラウザからアクセスするグループウェアで、サイボウズ コンタクト4単独でも利用可能だが、同社のサイボウズ Office4と連携することで、スケジュールの内容を引き継いで、そのまま報告書を作成できる。

「誰が誰と何を話したのか」という情報を報告書の形で記録し、顧客・社員・プロジェクトを軸に検索することが可能である。

報告書に対して自由にコメントを書き込むことができ、複数の部署で利用することで横断的な情報共有を実現する。

サイボウズOfficeシリーズで実績のある高速オブジェクト指向データベースを搭載し、目的の情報を素早く検索できる。顧客データベースや報告書の項目を自由にカスタマイズ可能となっている。



Software　　発売日　2001年2月1日

## ルータ・スイッチなどネットワーク管理ソフト RouterCare Ver 2.0

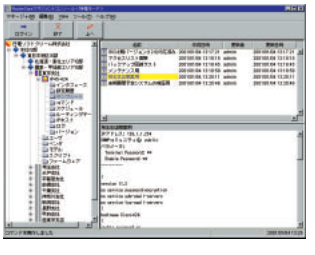
発売　NECソフト株式会社
TEL　03-5569-3399
価格　62万円（20ルータライセンス）～

URL　http://www.necsoft.co.jp/

NECソフトは、企業ネットワーク内LANに接続される複数のルータ、スイッチの設定情報を統合管理するネットワーク管理ツール「RouterCare Ver 2.0」を2月1日より発売した。価格は、基本機能に20ルータライセンスが付いた基本パックが62万円。

RouterCareは、マルチベンダーのルータ、スイッチなどに対応する管理ツールとしては唯一の製品。ルータ、スイッチの設定情報を一括管理し、ルータの遠隔操作もできるので、ルータ、スイッチの運用管理作業を効率的に行える。

CISCO、YAMAHA、Extreme Networks、Lucent Technologies、Foundry Networks、セイコーインスツルメントなどのルータ機器に対応する。Ver 2.0では、Linux、Solarisへの対応と、OpenView、Netvisorなど主要なネットワーク監視ツールとの連携機能などが追加された。




## シャープ、Linux搭載のPOSターミナルを発売

2001年2月23日

シャープは、チェーン展開やテナント展開を行っている小型店舗向けに、Linuxを搭載したPOSターミナル「RAZ-A320」を3月5日に発売すると発表した。価格は、67万2000円。

OSにオープンソースのLinuxを採用したことでシステムの開発の自由度が高く、Web環境でのシステム構築が行える。アプリケーション開発用の言語にJavaを採用し、本部サーバ上に置かれたJavaに対応した仕入れ発注や勤怠管理などのWebアプリケーションを、ホームページを閲覧する感覚で利用できるという。

本体には、タッチパネル付きの12.1型DSTNカラー液晶ディスプレイ（SVGA）、64Mバイトのメモリ（最大192Mバイト）、10Gバイトのハードディスクを内蔵する。インターフェイスは、RS-232C×2、USB×2、PCカードスロット（TypeII×2またはTypeIII×1）、磁気カードリーダなどを標準で装備し、オプションとして、無線LANカード、PHSカードなども用意する。本体サイズは、325（W）×410（D）×377（H）mmで、重量は約9kg。

シャープのニュースリリース
（http://www.sharp.co.jp/corporate/news/010223.html）

## TurbolinuxによるLinuxcareの合併、最終合意へ

2001年2月22日

米TurbolinuxによるLinuxcareとの合併について、2月21日に両社が最終合意に達したことが発表された。合併後はTurbolinuxとして運営されることとなるが、詳細については発表されていない。

TurbolinuxのCEOであるPaul Thomas氏はそのままCEOを継続、LinuxcareのCEO、Arthur F. Tyde IIIは、CTO（Chief Technology Officer）となる。

米Turbolinux（http://www.turbolinux.com/）
米Linuxcare（http://www.linuxcare.com/）

## アックス製組み込みLinux機用GUI環境「式神」事業で、レッドハットとテンアートニが提携

2001年2月22日

レッドハットとテンアートニは、アックスが現在開発中の携帯Linux機器向けGUI環境「式神（しきがみ）」について、開発から配布まで広範囲にわたる事業を共同で推進していくことを発表した。また、3社は組み込みLinux事業においてセミナー、展示会などのプロモーション活動、フィードバックの収集、開発サポート、個別適用などを共同で行う。

式神は、携帯機器に合わせて設計させたGUI環境で、操作が容易に行えるよう単純化された「デスクトップ」インターフェイスと、98%の認識率を誇る手書き文字認識システム「布目（ぬのめ）」が搭載されているのが特徴。

式神のAPIはすべてGNOME APIに準拠し、GPLのもとで公開されている。テンアートニは、Embedded System推進室を事業部に昇格させるなど組み込み系機器に対して積極姿勢を示し、「式神」に対しても開発コミュニティに自社エンジニアの参加を実施するという。

アックス（http://www.axe-inc.co.jp/）
テンアートニ（http://www.10art-ni.co.jp/）
レッドハット（http://www.redhat.com/）
式神の専用サイト（http://www.sikigami.com/）

## Linus Torvaldsの自伝、日米同時発売!

2001年2月19日

Linus Torvalds氏の自伝「JUST FOR FUN」が、米国ではHarperCollins、日本では小学館プロダクションの手により、5月上旬同時発売される。邦題は未定。その他の本の仕様については以下の通り。

タイトル……「JUST FOR FUN -THE STORY OF AN ACCIDENTAL REVOLUTIONARY-」（邦題未定）楽しくなくっちゃ―偶発的革命の物語―
著者……リーナス・トーバルズ／デビッド・ダイヤモンド
翻訳者……風見潤
発売日……2001年5月上旬、日米同時発売
体裁（予定）……260ページ前後

小学館プロダクション
（http://www.shopro.co.jp/）

## 「Mozilla 0.8」リリース

2001年2月17日

Mozilla.orgは2月15日、Mozillaの最新版「Mozilla 0.8」をリリースした。新たに搭載、改良された機能は以下の通り。

- Sidebarに、新しく“History”を実装し、履歴情報が一覧表示できるようになった
- 従来は新しいウィンドウを開くと元のウィンドウと完全に重なってしまったが、それがカスケード表示されるようになった
- WindowsやMacintoshにおいて、ダウンロードしたファイルをダウンロード終了と同時に起動する機能や、ダウンロード済みを表示するダイアログから直接ファイルマネージャなどを開く機能を実装した
- Linux上で、ポップアップメニューやコンテキストメニューが、ドラッグすることで移動したり順序を入れ替えたりできるようになった

Mozilla Mailに追加された機能は、

- メールを書く際に、従来の検索機能だけでなく置換機能が利用できるようになった
- メッセージの新着を音で知らせることができるようになった

このほかにリリースノートに記載されている新しい機能は、以下の通り。

- アニメーションGIF画像を非動作、もしくは一度だけ動作するように設定することが可能

・Mac OSの“Appearance Manager”のサポートが改善された
・Linux、Windows NT、Windows 2000のシステムカラーのサポートが改善された

2月15日現在公開されているのは、以下の3プラットフォームだ。

・Linux版（x86）
・Windows版（95／NT4.0以降）
・Mac OS版（Mac OS 8.6以降）

Linuxにインストールするには、glibc 2.1かそれ以降が必要になる。Red Hat Linux 6.0でJava 2をインストールしている場合には起動できないとのことだ。Red Hat Linux 6.1かそれ以降では、この問題は改善されている。また、Red Hat Linux 7では Standard C++ libraries for Red Hat 6.x compatibilityをインストールする必要があるとのこと。

Mozilla Organization
(http://www.mozilla.org/)
Mozilla.gr.jp（http://www.mozilla.gr.jp/）

## 「Samba 2.0.7-ja-2.2」リリース
2001年2月15日

日本Sambaユーザ会は、「Samba 2.0.7-ja-2.2」をリリースした。Sambaとは、UNIX系OSを使用しているマシンを、Windowsネットワークのファイルサーバやプリントサーバにするオープンソースソフトウェア。

今回の「Samba 2.0.7-ja-2.2」では以下の改良がなされた。

・文字コード変換ルーチンの最適化とWindows NT完全互換の達成
現時点で判明しているWindows 2000に関する不具合がすべて改修され、ネットワーク上にWindows 2000が存在する場合は、2.0.7以降を使うよう日本Sambaユーザ会では勧めている
・SWATの国際化機能実装における移植性の向上
SWATは、WebブラウザからSambaの設定を行うことができるツール。SWATがブラウザに出力する言語を、自動的に切り替える機能を向上させ、プラットフォームに依存した複雑な設定を軽減した
・一部の内部処理の高速化／最適化
・“coding system=EUC”で、外字や機種依存文字を含めた日本語化に完全対応
・“eucJP-open”および“UTF-8”への対応
・オリジナルのSambaにあった潜在的セキュリティホールの対応
・その他、これまで判明した細かいバグの修正
・翻訳中の米O'Reilly社の「Using Samba」の添付

日本Sambaユーザ会では、今回のリリースにあたって、Windowsで表示可能なすべての文字について実際に動作テストを行い、問題のないことを確認している。

日本Sambaユーザ会
(http://www.samba.gr.jp/)

## びぎねっとが初心者向けの技術情報提供を開始
2001年2月14日

びぎねっとが、3月1日より自社サイトの“びぎねっと”で、初心者への技術情報の提供を始める。びぎねっとは、ProjectBLUE（Business Linux Users Encouragement）関東セミナー世話人、日本Sambaユーザ会常任幹事渉外、IDG LinuxWorld Conference & Expoアドバイザリーボードチェアマンなどをつとめる宮原徹氏が2001年1月に設立した企業。社名は「Begin」と「Network」を組み合わせた造語だという。

具体的な業務内容は、Linuxやオープンソースソフトウェアを活用したネットワーク構築技術に関する情報の提供、各種ネットワーク製品のデータベースの提供、参加者が相互に支援し合うためのメーリングリストの運用など。

このうちメーリングリストでは、可能な限り敷居を低くし、初心者でも質問しやすいようにする。また、各種ネットワーク製品のデータベースでは、商用製品だけではなくオープンソースソフトウェアに関する情報も一元化して、一覧可能にするとのことだ。

“びぎねっと”ではこれらのサービスを原則無料とし、初年度にメーリングリストの登録者を1万人獲得することを目指すとしている。

“びぎねっと”(http://begi.net/)

## キヤノンがLinux向けBJプリンタドライバを開発。インターネットで無償ダウンロードサービスを開始
2001年2月10日

キヤノンは、Linux対応のカラーインクジェットプリンタ「BJシリーズ」用プリンタドライバ「Canon Bubble Jet Print Filter for Linux」を開発し、同ドライバソフトの無償ダウンロードサービスを「キヤノンBJプリンタオフィシャルホームページ」において開始した。

「Canon Bubble Jet Print Filter for Linux」が現在対応しているディストリビューションは以下の3つ。

・Red Hat Linux 6.2J、7J
・Turbolinux 6.0
・Vine Linux 2.1CR

今回のドライバソフトに対応する機種は、BJ F870、BJ F860、BJ F850の3モデルで、順次対応機種を増やしていく予定だという。また、現在はダウンロードのみだが、2001年2月中旬からはディストリビューションへの同梱も検討しているという。

「Canon Bubble Jet Print Filter for Linux」の詳細およびダウンロードサイト
(http://www.canon-sales.co.jp/drv-upd/linux/bjlinux100.html)

# 日刊アスキー Linux Business Report

http://www.linux24.com/

ここ1年の間にLinux関連ビジネスも勃興期から安定した成長期に入ってきたようだ。従来は主に技術系の人々によって展開されてきたLinux関連ビジネスが、マーケティング系の人々の参入によって、急速に業界としての形を整えてきた。そこで大きく展開されたのが、ディストリビューションベンダー各社による広範囲なアライアンスだ。ここで、Linux業界の現状を確認するため、レッドハット、ターボリナックス ジャパン、レーザーファイブ各社のパートナーシップやベンダー資格制度などを整理してみたい。

## ターボリナックス ジャパン

ここ1年の間、Linux関連ではきわめて多くの提携が行われてきた。特にターボリナックス ジャパンは、1カ月に1～2社のペースで提携先企業を増やしている。KylixやiOffice、サイボウズOfficeなど、注目されるアプリケーションのベンダーをキッチリとおさえている。

同社のパートナー戦略として、1年前に発表されたのが「TurboLinks」である。このプログラムはパートナー各社に対し、技術サポートはもちろんのこと、共同マーケティングや販促活動を行っていくというもので、製品に同梱されるCD-ROMにも業務内容が収録される。パートナーにとっては悪い話ではないだろう。TurboLinksへの参加イコール提携ということではないが、同社との協業を考える企業は増え、結果として提携先も増えたはずだ。現在、SIパートナーだけで100社以上を数えている。

## レーザーファイブ

レーザーファイブのアライアンスの特徴は、技術的サービスを中心にしていることだ。そのパートナープログラムが、昨年のLinuxWorld Conference&Demo/Tokyo 2000で発表された、レーザーファイブの「LASER5 P.a.S.S」である。これは教育と技術支援に重きを置いたプログラムであり、ハードウェアからアプリケーションに至るまで、ワンストップの技術支援を行える点と、LPI（Linux Professional Institute）などの中立的資格の採用など、自社製品使用を必ずしも前提としない点を特徴としている。

具体的な提携先を見てみると、ASP、スーパーコンピューティング（学術計算など）、アプライアンス機器など、堅実な分野をキッチリと押さえているといった印象だ。また、先ほどのLASER5 P.a.S.Sの一端を担うドットネットとは資本提携も行っている。

## レッドハット

レッドハットの提携先は、ノーザンライツコンピュータやぷらっとホームなど、技術者の間で有名な企業との提携が目に付く。さらに、今後大きく伸びる分野と見られている組み込み系企業との提携も特徴だ。50社にのぼるパートナー企業の顔ぶれも、国内の有名コンピュータ関連企業がほとんど加わっていると思われるほど、豪華なものだ。

同社の場合はまた、米国の本社同士が世界的戦略における包括的な提携を結んでいる場合も多い。

## 各社の技術認定資格制度

昨年はLinuxの資格制度がいっせいに花開いた年でもあった。Oracleマスターやコンパックコンピュータのベンダー資格ASEなど従来からのベンダー資格がLinuxに対応し始めたのみならず、ターボリナックスの「Turbo-CE」、レッドハットの「RHCE」、ベンダーからは中立的な立場をとる「LPI（Linux Professional Institute）」が開始された。RHCEはすでに米国で教本が出版されているなど、Linux関連資格の老舗であり、国内でも、RHCE取得が就業条件になっている企業もある。

Turbo-CEも、SIパートナー100社以上という勢いを見る限り、一定の勢力を保っている。Turbo-CEの特徴としては、資格そのものの敷居はあまり高くせず、取得後にサポートをして本人のスキルアップを目指すというものだという。

ここで気になるのがLPIとベンダー資格の関係だ。囲い込みをよしとしないLinux業界の気風に従ってか、Turbo-CEはとりあえずLPIに準拠する方向、レーザーファイブも技術者認定はLPIに拠るとしている。

| ターボリナックスジャパン | | |
|---|---|---|
| 形態 | パートナー企業 | 内容 |
| 提携 | オラクル | Oracle 8iに最適化したTurbolinuxを開発 |
| 提携 | ドゥイット | FlexMessenger 2.01の販売とマーケティング |
| 提携 | ボーランド | Kylix日本語版に関して戦略的提携 |
| 提携 | ネオジャパン | OracleとiOfficeによるソリューション開発と販売チャネル育成 |
| 提携 | サイボウズ | サイボウズ Office 4の販売／マーケティング |
| 提携 | ハミングバード・ジャパン | 日本語全文検索システム「Fulcrum SearchServer for TurboLinux」 |
| 提携 | サイバーキャッシュ | EC分野において提携。成果として「BuySmart Web」とその開発モジュールを同社のサーバ製品に対応させ、サポートを行なう |
| 提携 | シノックス | メッセージングアプリケーションについて、マーケティング／サポートを行なう |
| 提携 | ニューホライズンジャパン | Linuxエンジニア教育関連 |
| 提携 | アドックインターナショナル | ネットワーク監視システムの販売／サポート関連 |
| 提携 | トータルネットワークサービス | オールインワンサーバで提携 |
| 提携 | 東京リーガルマインド | LECのLinux講座にTurbo-CE取得コースを設置 |
| 提携 | スリーアールソフト | メッセージングサーバ「@MESSAGE-MailStudio」やコンサルティングサービス |
| 提携 | コンパック | 「Linuxオールインサーバー」関連 |
| 提携 | オラクル | 「Linuxオールインサーバー」関連 |
| 提携 | オムロンアルファテック | 「Linuxオールインサーバー」関連 |
| 提携 | 富士通 | IAサーバ「PRIMERGY」へのTurboLinuxインストール代行。サポート、教育、開発などに関して包括的に提携 |
| 提携 | フォーラムシステムズ | iモードなどへのリアルタイム広告配信システムや無料アドレス帳サービスをTurboLinux Server上で開発するために技術提携 |
| 提携 | グローバル ナレッジ ネットワーク インク | 教育分野関連 |
| 提携 | ソフテック | クラスタリングソフトウェア「enFuzion 6.0」の販売／コンサルティング |
| 提携 | システム・テクノロジー・アイ | iStudy for Turbo-CEの販売／推奨 |
| 提携 | Connectix | Mac OS上で動作するVirtual PCにTurboLinuxをプリインストール |
| 提携 | VMware | VMware Expressのバンドル |
| 提携 | ヴァル研究所 | 駅すぱあと・イントラネット版について協力、体験版をTurbolinuxに同梱 |
| 提携 | 日立エンジニアリング | 通信、製造業分野におけるクラスタシステムのコンサルティング |
| 提携 | Ingram Micro | 販売流通に関して業務提携 |
| 提携 | シルバン・プロメトリック | Turbo-CEの試験運用 |
| 提携 | オービックビジネスコンサルタント | 奉行シリーズの販売および、Linuxビジネスソリューション支援教育コース |
| 提携 | 華迪計算机有限公司 | Space Linuxの共同開発 |
| 提携 | 大塚商会 | 販売／サポート／営業 |
| 提携 | パーソンアイ・ドット・コム・リミテッド | Personi Office 1.0のTurbolinux Server対応版開発で技術提携 |
| 提携 | NEC | Expressサーバ導入時におけるサポートなどの技術支援 |
| 協業 | 蝶理情報システム | 「FleXML for Turbolinux」の共同開発／マーケティング |
| 協業 | コンピュータ・アソシエイツ | 販売／マーケティング |
| 協業 | 大塚商会と日本アイ・ビー・エム | SMILE α for TruboLinx |
| 協業 | RSA Security | Turbolinux Server日本語版 6.1にRSA BSAFE SSL-Cをバンドル |
| 技術協力 | 富士通 | iアプリ用ツール |
| 技術協力 | NECソリューションズ | 技術研究組合 新情報処理開発機構(RWCP)による大規模クラスタシステムにおける技術協力 |
| **レーザーファイブ** | | |
| 提携 | ドットネット | 教育分野で資本提携 |
| 提携 | ネットエージェント | ネットワークセキュリティ監査サービス |
| 提携 | キヤノン・スーパーコンピューティングS.I | RAIKOU業務で提携 |
| 協業 | 華東師範大学 | 技術協力 |
| 提携 | 理乃科高速計算机（上海）有限公司 | 組み込み、スーパーコンピューティング |
| 提携 | サードウェア | LinuxSI事業者。資本提携 |
| 協業 | ナカガワメタル | セキュリティ機器『パケットブラックホール』の機器製造 |
| 協業 | ネットエージェント | 『パケットブラックホール』の開発 |
| 協業 | でんさテクノ東京 | L-Servoにおける協業 |
| 協業 | アクシスソフトウェア | L-Servoにおける協業 |
| **レッドハット** | | |
| 提携 | アクシスソフトウェア | Linuxシステム構築サービスの提供 |
| 提携 | ぷらっとホーム | 「Open “ S ” Alliance」（アプライアンスサーバ）分野 |
| 提携 | デジタルデザイン | silver neon上でのRed Hat Linuxを採用 |
| 提携 | ノーザンライツコンピュータ | ノーザンライツコンピュータの標準OSとしてRed Hat Linuxを採用 |
| 提携 | トレンドマイクロ | 技術提携 |
| 提携 | HP | 世界規模の提携（本社） |
| 提携 | Dell Computer | 世界規模の提携 |
| 協業 | コンピュータ・アソシエイツ | 包括的協業強化 |
| 協業 | ダイエー情報システム | 包括的な戦略的パートナーシップ（事例としてローソン） |
| 提携 | テンアートニ | iDC／xSP向けソリューション、サポート、組み込みLinux |

各ディストリビューターのアライアンス状況


# LinuxWorld Conference & Expo/NY 2001

**1月30日から2月2日にかけて、21世紀最初のLinuxWorldが米国ニューヨークで開催された。ビジネス／コミュニティ双方のメンバーが集った会場のようすをお伝えしよう。**

「冬のニューヨークは寒い」はずなのだが、LinuxWorldの会期中、1月30日から2月2日はおだやかと言ってもいいような天気だった。翌日の2月3日からは急に冷え込んだことから、おそらくLinux関係者に「晴れ男／晴れ女」が大勢いたということだろう。

会場となったJavitsコンベンションセンターは、マンハッタン島の西端に位置しており、Linuxのみの展示会を開催するには大きすぎるのではと感じてしまうような大規模な施設だ。だがそんな危惧は杞憂であったようで、約300の企業が出展した会場には、約2万5000人が来場し、大いなるにぎわいを見せた。

## オープンソース ドリームチーム

LinuxWorldは、“ Linux ”と銘打たれてはいるが、実際にはオープンソースに携わっているすべての人々を対象としている催し物だ。それは報道関係者に配られるプレスカードにも表れている。裏面にはおなじみのペンギン、Tuxではなく、BSD界のアイドルであるデーモンが描かれているのだ。これは、商用のBSD UNIXをリリースしているBSDi社がスポンサーに加わっているためだ。また同社のブースで配っていた、デーモンのツノ型カチューシャと尻尾を多くの来場者が付けていたので、展示会場内は妙にデーモンの存在感が強かった。LinuxWorldがLinuxだけでなく、オープンソース全体の祭典であるということの表れだろう。

## 基調講演

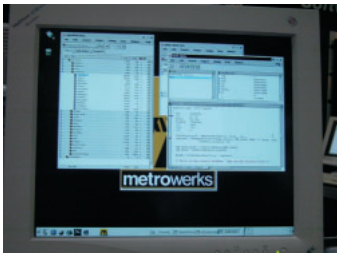
各企業のLinuxへの力の入れ具合がわかる基調講演は、31日にIBMの社長兼最高執行責任者（COO）であるSamuel J. Palmisano氏から始まった。IBMは昨年、Linux関連のビジネスへの10億ドルの投資を発表するなど、最もLinuxに力を入れているメーカーのひとつだ。

講演のキーワードは、「エンタープライズ」。Linuxのビジネス現場での利用は、小規模のサーバから始まり、徐々に大規模な用途に広がってきたが、ついには企業の基幹レベルのシステム構築にも使われるようになった。その実例として、石油メジャーの米国シェル社による石油採掘用の分析システムなどが紹介されていた。また、ダイエーグループのローソンの各店舗へのLinuxマシン導入も、大量導入の事例として紹介されていた。約7500の店舗に2台ずつ、合計1万5000台の導入実績は世界最大級ということだ（次号で紹介予定）。

IBMは“ eServer ”という統一ブランド名でサーバを揃えているが、今回の基調講演で最大64個のCPUを搭載できるx430シリーズを発表した。これは、比較的小規模用途向けのx86アーキテクチャサーバと、ハイエンドのメインフレームの中間の製品であり、これにより同社の全サーバ製品群でLinuxが動作することになった。

翌日2月1日には、インテルの「インテル・アーキテクチャ・グループ」副社長であるWilliam A. Swope氏が登壇した。

氏の講演でも、今後ハイエンド分野にまでLinuxの利用を広げていくこと



多くのプラットフォーム用に開発環境を提供しているメトロワークス社は、CodeWarriorのLinux版を出展していた。

BSDi社では、こんなにキュートなデーモンガールのお出迎えが受けられる！

BSDi社のブース。存在感の大きさには、Tuxペンギンも負けそうだ。


SGI社はLinuxでコントロールされたロボット2人が活躍していた。

が主な話題であった。Webサーバやメールサーバとして Linuxマシンが用いられている現状から、ミッション・クリティカルな用途にまでLinuxを導入していくことで、オープンソースコミュニティに大きなビジネスチャンスが生じると語った。

ハイエンド用途に話が進めば、もちろんインテルの64ビットプロセッサ、Itaniumが登場してくるわけで、Itaniumマシンを用いたクラスタからのビデオストリーム送出のデモが行われた。また、NCSAのゲストとともにItaniumを4個搭載したLinuxマシンで科学計算のシミュレーションを実演してみせた。

Itaniumの出荷は遅れに遅れているが、今年半ばにはリリースされると言われている。リリースと同時に利用できるOSは、Linuxしかないのが現状であり、インテルがLinuxに寄せる期待が大きいのも当然といえるだろう。

## オープンソースに関する10の神話

どちらかといえばお堅い雰囲気だったインテルやIBMの重役による講演に比べ、1日の午後から行われたVA Linux Systemsの創立者でありCEO（最高経営責任者）であるLarry M. Augustin氏の講演は、ユーモアに満ちたものだった。

氏の講演もLinuxでエンタープライズ市場を狙うというのが話題の中心だ。最初に、その妨げになりそうなことを十項目あげ、それが誤っていることをひとつひとつ示していった。題して「オープンソースに関する10の神話」（TOP 10 myths about open source）。どこかで聞いたようなタイトルだ。「Linuxにはアプリケーションがない」「プロジェクトリーダーが辞めてしまうと、そのプロジェクトが終わってしまう」など、根拠のない「神話」をひとつずつ写真付きで説明していくのだが、みごとな話術もあり客席は笑いにつつまれていた。中でも「オープンソースの製品は信頼性が低い」の反例として、マイクロソフト社のWebサイトが2日間も閲覧不能になった件が示されたときは、大爆笑が起きた。

最後の「オープンソースはエンタープライズ用途に対応できない」については、米オラクル社の役員や、VA Linux Systemsのユーザーをゲストに招き、いわゆるeビジネス分野ですでにLinuxの利用が進んでいることを示した。

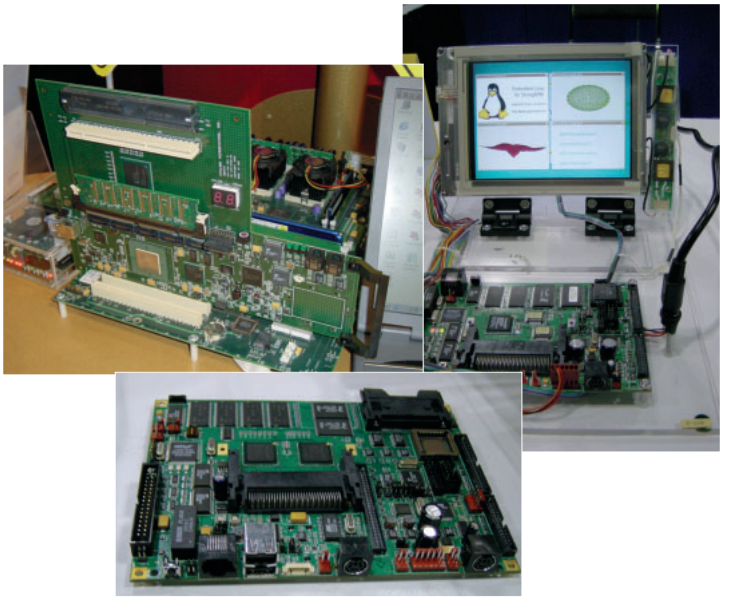
## キーワードはエンタープライズ

最終日2日の基調講演は、ヨーロッパで大きなシェアを持つSuSEのCTO（最高技術責任者）であり、XFree86プロジェクトの主要メンバーでもあるDirk Hohndel氏が行った。

ルフトハンザ航空など大手企業でのLinuxの採用実績が示され、エンタープライズ用途への着実な進出がヨーロッパでも進んでいることを明らかにした。

基調講演を行った4人のVIPがいずれも口にしたのは、「エンタープライズ」という言葉だ。小規模なグループ内のサーバから利用され始めたLinuxをはじめとするオープンソフトウェアは、

今やサーバといえばフルタワーではなく、1Uの薄型がスタンダードだ。多くの会社がこのタイプの製品を出展していた。



小型で低消費電力の組み込み用基板も多く見られた。Java VM搭載などで差異化が図られていた。

今や企業の情報の流れを一手に引き受けるような、失敗が許されない分野でも利用できるほどに成長した。そして今後もさらにこの流れは続いていくだろう。そのためには、オープンソースのコミュニティと企業とが協力していくことが不可欠である。もちろん各社ともそのことを承知しており、この流れにどのようにコミットしていけるのかを示したのが、今回の基調講演だ。「無料のOSでどうやって儲けるのか」などと言われていた時代とは、まさに隔世の感がある。

## IDG/Linus Torvalds Award

初日の基調講演後に、LinuxWorld恒例のIDG/Linus Torvalds Awardの受賞式が行われた。これは活発に活動しているオープンソースコミュニティに送られる賞で、前回はDebian Project、前々回はXFree86 Projectが受賞している。今回この栄誉に輝いたのは、Samba Teamで、賞金2万5000ドルを贈られた。

## 展示会場

展示会場には、大小合わせて約300社の企業のほかに、VA Linux Systemsがスポンサーになっている“.Orgパビリオン”にいくつかの団体が出展しており、世界中のコミュニティからメンバーが集まって、意見交換や親善の場となっていた。

プログラミングに関する議論を行っているところもあれば、Slashdotのブースのように、Playstation2と大型のモニタが設置され、一日中「鉄拳」をやっているところもあるなど、会場内でも特にコミュニティ色の強い場所となっていた。そのためこの周辺は、オープンソース関係の有名人に遭遇できる確率が非常に高い場所になっていた。実際、Red HatのCEOであるRobert Young氏、「伽藍とバザール」の作者であるESRこと、Eric S. Raymond氏、そして家族連れのLinus Torvalds氏なども見掛けられた。

企業の出展では、1Uサイズのラックマウントサーバや、組み込み用途向けの小型の基板が多く見られた。

大規模なクラスタサーバやホスティングサービスを行う企業では、いかに大量のサーバを設置できるかが重要であり、そのような用途では1Uの薄型サーバが主流になっている。この需要に合わせて、コバルトネットワークス（サンマイクロシステムズが買収）をはじめ、多数の企業が1Uサーバを出展していた。その中ではAPIから出展されていた、数値計算用途向けにAlphaプロセッサを2台搭載した1Uサーバが異彩を放っていた。

組み込み用途向けに、x86よりも低消費電力のCPUを用いた小型基板も多く展示されており、Javaアプレットを動作させたり、3Dの表示を行うなどさまざまなバリエーションが見られた。

1Uサーバや組み込み向け基板が多いのは日本でも米国も同じだが、国内の展示会でよく見掛ける、小型でカラフルな筐体に入った、SOHO向けのアプライアンスサーバはほとんどなかったようだ。オフィスや住宅の空間に余裕が

ついにAlphaプロセッサまでが、1Uサーバの筐体に！（しかも2個）833MHz×2のAlphaによって、他を圧倒する数値計算能力を誇っている。

x86-64プロセッサのシミュレータ“VirtuHammer”（Virtutech社製）。既存のシミュレータSimNow!の100倍の速度を誇り、x86-64の開発を加速する。

ギガビットネットワークによるビデオストリーミングのデモ。インテルのブースはハイエンドから組み込みまで、非常に多彩だ。

展示会にはつきものの、お姉さん。セキュリティ関連の商品を扱っているブースなので、「ミニ○カポリス」である。

あり、電気料金も日本より安い米国では、このような小型機器ではなく、フルタワーのマシンを用いるのだろうか。

## CPUメーカーがんばる

現在のx86 CPU市場では相変わらずインテルが強いが、AMDもPC DIY市場を中心に支持を集めている。その両社は、最新のx86 CPUを用いた製品の展示とともに、x86以降を見据えた技術のプレビューともいえる展示を行っていた。一見Linuxとは直接関係なさそうなCPUメーカーだが、両社のLinuxへの注力ぶりはかなりのものだ。

AMDは、来年以降x86-64テクノロジーに基づいたCPU「Hammerシリーズ」の出荷を計画している。これは既存のx86命令を拡張した64ビットのCPUだ。今回、このx86-64プロセッサの開発を、既存のx86 PCで行うための“VirtuHammer”シミュレータ（開発はVirtutech）が初めて公開された。これ以前にAMDから提供されていたシミュレータ（SimNow!）の100倍の実行速度を実現しているという。今のところ1.2GHzのAthlonマシン上でVirtuHammerを動作させた環境で、x86-64ネイティブコードのLinuxが4分で起動するそうだ。

VirtuHammerは、x86-64へのLinux移植を行っているコミュニティに提供される予定。また、ヨーロッパのディストリビュータSuSEとも提携を行い、x86-64 Linuxの開発を加速している。

AMDのx86-64もインテルのItaniumと同様に、リリースと同時に使えるOSは、Linuxだけになると思われる（Windowsファミリーがx86-64をサポートする予定は、今のところない）。もちろん、既存のx86命令も実行可能ではあるが、真の実力は64ビットモードネイティブなOSがあってはじめて発揮できる。AMDがLinuxに深くコミットするのも当然といえるだろう。

インテルのブースでは、Itaniumベースのクラスタシステムを PDAを使ってワイヤレスで制御するというデモが行われていたほか、ネットワーク、ソフト開発環境など多くのセクションが用意されていた。

Itaniumマシンやギガビットのネットワークといったハイエンドから、PDA用のStrongARMまで製品を持ち、全方位に注力できる半導体メーカーは、インテルくらいのものだろう。自社のハードウェアに対応したソフトウェアも含めて、システム全体でビジネスができるのが、同社の強みと言える。

## 新興の各社には試練の時代か

大手企業が次々に参入することで、オープンソース市場が活発になっていくのは歓迎すべきことだが、一方で比較的規模の小さい新興の企業は、今まで以上に生き残るための努力を強いられることになる。

おりしも、LinuxWorld開催の直前にカナダのStormix Technologies社が営業を停止しており、同社のブースがあるはずの場所は、がらんとしていた。新興のLinux企業にとっては、これからが正念場であると感じさせられた光景であった。

速報 Borland

# 統合開発環境DelphiのLinux対応版「Kylix」リリース

GUIのアプリケーションを簡単に作成できる統合開発環境としてWindowsで高い人気を誇っているBorland Delphiが、もうすぐLinuxでも利用できるようになる。Kylix（カイリックス）と呼ばれるこの製品のフィールドテスト版を入手したのでその概要を紹介しよう。

文：渡邉利和
*Text：Toshikazu Watanabe*
*(toshi-w@tt.rim.or.jp)*

2001年1月30日～2月2日の4日間に渡りニューヨークで開催されたLinuxWorld Conference & Expoで、ようやくKylixのリリースが発表された。KylixはBorlandが開発したソフトウェア開発ツールであり、言語にObject Pascalを採用したIDE（Integrated Development Environment：統合開発環境）である。

内容としては同社のDelphiと同等であり、Delphi for Linuxと呼んで差し支えないものだ。実のところ「Kylix」は開発コード名ということになっていたのだが、開発段階から大変な注目を集め、Kylixという名称がすっかり認知された状態でもあることから、このままの名称で製品リリースされることになったという。

なお、Kylixの製品情報は、ボーランドのWebサイト（http://www.borland.co.jp/kylix/）に掲載されている。

## Kylixの概要

Delphiは、Windows向けIDE市場で大変大きなシェアを持つ開発ツールである。このDelphiの使い勝手をそのままの形でLinuxに持ち込んだKylixは、これまではテキストエディタ＋コンパイラというCUI環境で行われるのが一般的だったLinuxでのプログラム開発作業をGUIベースに移行させる大きなきっかけになるものと期待されている。

Kylixには3種類のパッケージが用意される。最上位となる「Kylix Server Developer」は165種類以上のCLXコンポーネント、データベース接続のためのネイティブドライバ、SQLモニタなどを備え、大規模なデータベースアプリケーション開発にも対応するパッケージだ。米国での価格は1999USドルの予定。

「Kylix Desktop Developer」は標準版といった位置づけのパッケージだ。CLXコンポーネントの数が130以上とやや少なくなり、データベースアプリケーション開発機能の一部が省略される代わりに、価格も999USドルと約半分に抑えられている。

また、最近のBorlandの開発ツールではすべて同様の手法が採られているのだが、KylixでもWebで無償公開されるバージョンが用意される。「Kylix Open Edition」がそうで、メディアを購入する場合は99USドル、Webからのダウンロードであれば無償で入手可能だ。

出荷時期は、「Open Edition」が2001年中頃、「Server Developer」と「Desktop Developer」が2001年第1四半期となっている。開発段階から何度となく話題に上り、期待を集めたKylixだが、製品として登場するのも間近である。日本語版に関して、ターボリナックス ジャパンと技術面およびマーケティング面での戦略的提携を行うことが発表されている。ただし、現時点では日本語版の出荷時期は明らかになっていない。

## Kylixの特徴

Kylixの特徴は、実のところDelphiの特徴とほぼ同等である。Kylixの実体はDelphiをLinux上に移植したものと考えてよいため、これはある意味当然のことといえる。

Linuxで標準的な開発環境というと、やはりgccということになるだろうか。しかし、ちょっとしたフィルタコマンド程度のものを作るのであればテキストエディタでソースコードを記述し、コマンドラインコンパイラでコンパイルして作っても何の問題もないが、ウィンドウアプリケーションを作成するとなるとこれではいかにも効率が悪い。やはり、GUIアプリケーションを開発する場合は開発段階からGUI環境で作業するほうが便利である。

Kylixでは、Windows環境でのDelphiで定評のあるビジュアル開発環境をLinuxに持ち込むことで、GUI開発をLinux上で実行できるようにしている。最近のLinuxではGUI（X Window System）なしの環境のほうが珍しく感じられるほどで、ユーザーにとってはGUIベースのアプリケーションが好まれる状況がすでにできあがっていると言ってよい。

BorlandのGUI開発ツールは、単にGUIでコンポーネントの配置ができるというだけにとどまらず、ソースコードとGUIの連携が完全に取れている点

が特徴である。

これは、2Wayツールと呼ばれるもので、GUI表示での変更点は即座にソースコードに反映されるし、ソースコードの変更も同様にGUI画面にその場で反映される。従って、開発者はソースコードでもGUI画面でも、どちらでも好きな環境で作業を実行できる。ウィンドウ上のボタンの配置や色の決定などはGUI画面で行い、ボタンに割り当てるメソッドのコーディングはソースコードとして記述するが、この両者はどの時点においても矛盾なく完全に統合されているわけだ。この特徴により、GUIアプリケーション開発の効率が大きく向上している。

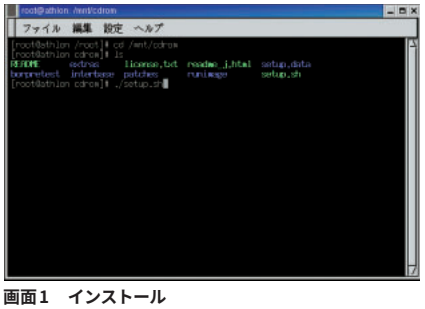
Kylixは完全なネイティブコードアプリケーションを出力するため、アプリケーションの動作時に特別なランタイムライブラリを用意したり、あるいはインタープリタをインストールしておく必要などはない。また、コンパイラの動作もきわめて高速であり、開発／テスト実行／デバッグというサイクルをストレスなく何度でも必要なだけ実行できる点も開発効率の向上に大きく貢献している。

デバッグはGUI環境のままビジュアルに実行でき、エラー時にソースコードの該当行に自動的にジャンプするといった基本的な機能はもちろん、ブレークポイントの設定やデバッグ時にのみ利用する特別な条件設定などがすべてGUI環境で利用可能だ。

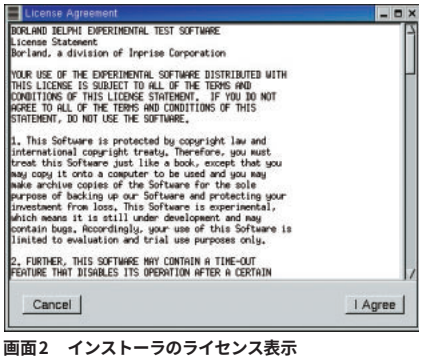
テスト実行は開発画面からボタンを1つクリックするだけで即座に行える。コンパイル作業が高速なこともあってこの作業にストレスを感じることはないし、テストのための環境は本番環境とまったく同一であり、妙な矛盾が生じたりすることはない。

Delphiではデータベースアプリケーション開発のサポートが充実しており、企業の業務システムの開発でも広く利用されているが、この性格はKylixにも引き継がれている。データベースのテーブルをビジュアルに表現しながらアプリケーション開発ができるデータアクセス対応コンポーネントが豊富に用意されているほか、Server DeveloperではInterBase、MySQL、Oracle8i、IBM DB2といった主要データベースに対応したネイティブドライバ「dbExpress」も提供されるので、業務アプリケーションをWindowsからLinuxに移植する、といった作業はとても容易になるはずだ。

Kylixでは、Webアプリケーション開発がサポートされている点にも注目すべきだろう。Apacheのモジュール開発がサポートされており、ウィザード形式で基本的な部分ができあがってし

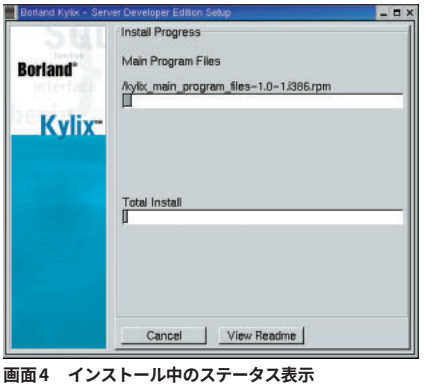


画面1　インストール
インストーラは、CD-ROM上でスクリプトを実行することで起動する。コマンドラインから実行すればよい。

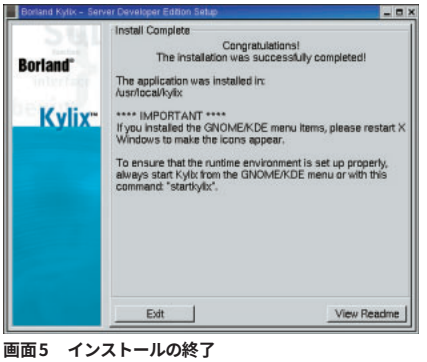



画面2　インストーラのライセンス表示
最初に表示されるのは、おなじみのライセンス表示。インストーラは、X Window Systemからだとテキスト画面で表示する。

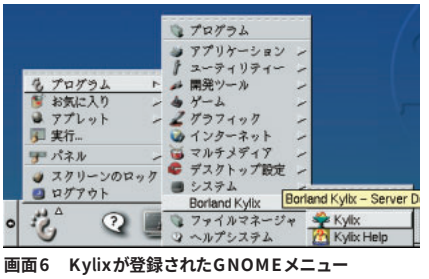

画面3　インストールオプションの設定
指定すべきオプションはほぼこれだけで、中でも重要なのはパスの指定だ。ドロップダウンリストによく使われそうなパス名がいくつか登録されているので、そこから選ぶだけで通常は十分なはずだ。インストールするファイル群の指定もここで行なう。




画面4　インストール中のステータス表示
あとは、ファイルのコピーが終わるのを待つだけだ。




画面5　インストールの終了
ここでX Window Systemをリスタートすると、GNOME／KDEのメニューにKylixが登録される。「X Windows」となっているのはご愛敬？




画面6　Kylixが登録されたGNOMEメニュー
GNOMEでは、こんな感じになる。「開発ツール」メニューの下に入ってもよいかとは思うが、このあたりは好みで調整すればよいだろう。

まうため、必要な処理のみを記述すればWebサーバ上で実行するロジックを作成するのが大変容易になる。

従来は、Webアプリケーションを作成しようと思うと、CGIにしてもApacheのモジュールとして作成する場合でも、Webサーバとのやりとりのためのインターフェイスのレベルから修得する必要があってなかなか面倒だったが、Kylixを利用すればそうした環境依存の定型処理の部分は自動的に作成されてしまうことになるので、迅速な開発ができるようになるだろう。

この機能とデータベースアクセスの機能を組み合わせると、「バックエンドに置いたデータベースにアクセスして処理を行うWebアプリケーション」が実に簡単に作れるようになる。この形式のアプリケーションは、オンラインショッピングサイトはもちろん、ちょっとした掲示板システムやWebメールなど、さまざまな場面で使われている。しかも、WebのシステムではLinuxサーバの利用頻度も高いため、このWebアプリケーション構築機能はKylixの大きな魅力と考えられるだろう。この機能だけを目当てにKylixの利用を開始しても十分に元が取れるほどではないだろうか。


画面7　インストール直後の起動時に表示される画面
最初の起動時には、フォントマトリックスを生成するとかで少々待たされる。このパネルは、2度目からは表示されない。


画面8　Kylix起動時に表示されるロゴ画面
起動時に表示されるウィンドウ。テスト版のものなので、製品リリース時には変更されると思われる。

## クロスプラットフォームライブラリCLX

GUIベースのIDEでは、マウス操作によって画面にコンポーネントを配置し、必要な処理を記述していく、という手法で開発を行う。ここで重要なのがコンポーネントの充実度合いである。使いやすく性能のよいコンポーネントが豊富に揃っている環境では開発は容易であり、逆に使えるコンポーネントが乏しい環境では開発者が独自にコーディングすべき要素が増えることになる。

Kylixは、Delphiで定評のあるコンポーネント群をLinux対応とすることで充実した開発環境を構築している。KylixでサポートされているコンポーネントはCLX（Component Library for Cross Platform）と呼ばれる機種依存性の低いコンポーネントライブラリだ。Delphiでは当然コンポーネントはWindows対応として作成されていたが、この仕様を変更せずにクロスプラットフォーム化を実現するのがCLXだと考えてよい。

現在のDelphiではまだコンポーネントはCLXではなくWindows専用のものなので、実際にはDelphiとKylixでは異なるコンポーネントを利用していることになる。しかし、次期バージョンのDelphiではCLXが採用されることになっているので、この時点でDelphiとKylixの基本的な環境の統合が行われることになる。つまり、WindowsとLinuxを自由に行き来しながら開発が行えるということだ。ただし、現在KylixでサポートされているCLXは、仕様としてはDelphiのコンポーネントを踏まえているため、Delphiで修得したコンポーネントに関する知識が無駄になることはない。

## Kylixのインストール

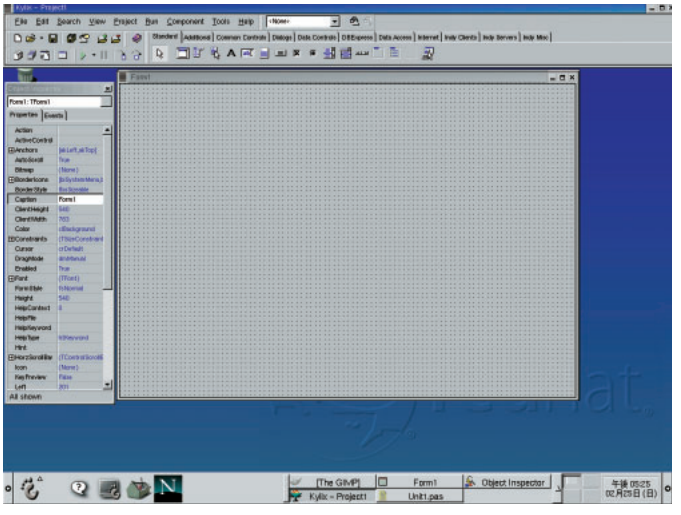
では、Kylixをインストールしてみた様子をご紹介しておこう。今回は、Red Hat Linux 7JにField Test#4というバージョンをインストールしてみた。

Kylixの動作環境は、

- **Linuxカーネル　2.2以上**
- **glibc　2.1.2以上**
- **libjpeg　6.2以上**
- **X Window System　XFree86など**

となっている。なお、Field Test#4で

画面9　Kylixのウィンドウ表示
起動直後のようす。画面をほぼいっぱいに使っているが、必要に応じてウィンドウを閉じたり開いたりできるので、特に問題はない。



配布されるすべてのファイルをインストールするためにはHDDに167Mバイトの空き容量が必要だった。ファイルシステムは、シンボリックリンクをサポートしたものである必要があり、FAT、FAT32、SMB、Novell File Systemへのインストールはサポートされていない。

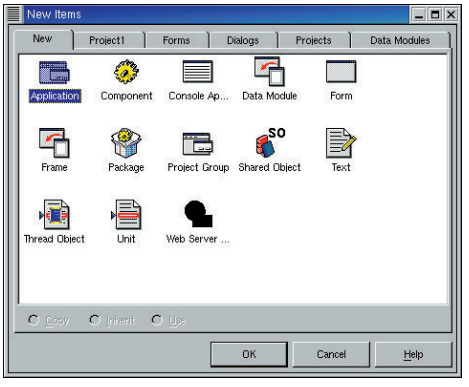
なお、現在一般的に利用されているディストリビューションで利用されているglibcには不具合が見つかっており、そのままではKylixが正常に動作しないことから、glibcのパッチも収録されていた。まずこのパッチを適用してglibcをアップデートした後にKylixをインストールしたのだが、パッチ自体はrpmファイルで提供されており、適用はごく簡単な作業だ。これはKylix特有の問題というわけではないので、将来のディストリビューションでは解決されると予想される。

インストール作業は、コマンドラインからインストールスクリプトを実行し、基本的な質問に回答していくだけの簡単な作業である。

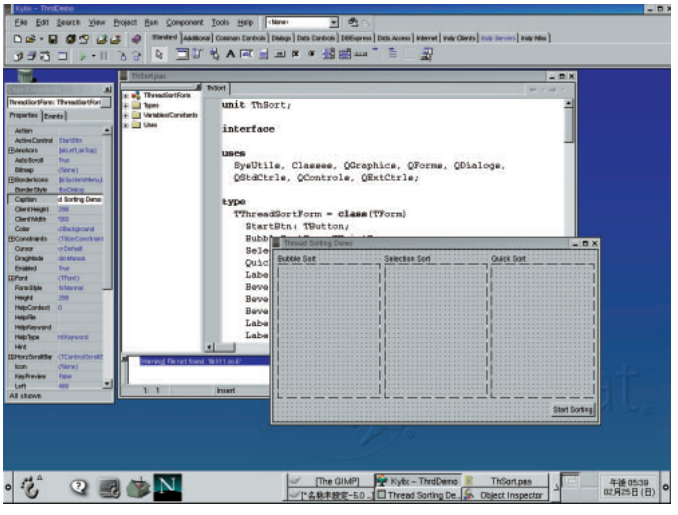
GUI環境でインストールを実行した場合にはインストーラもGUIで動作するため、特にわかりにくいこともない。現在のバージョンでは表示の日本語化などは行われていないが、見た目の印象はWindows上のDelphiと変わるところはない。IDEとしては標準的なインターフェイスであり、現在では多くのIDEツールが同様のルックアンドフィールを採用していることから、Delphiを使った経験がなくても特に迷うことはないだろう。

Linux上で動作するGUIアプリケーションを作成しようとすると、各種のGUIツールキットの使い方を憶えるなどといった準備が必要であり、やはりそれなりに敷居の高いものであった。しかし、Kylixを利用すればこのレベルの知識は持っていなくてもとりあえずウィンドウやボタンを組み合わせたアプリケーションウィンドウを生成し、処理を行うことができる。ObjectPascalというほかでは使われていない言語の文法を憶える必要はあるが、今時のオブジェクト指向言語をひとつでも知っていれば特段難しいものではないので、その点も心配ない。

Linuxはプログラミングのための環境としても強力な機能を持っているのだが、これまでは初心者でも簡単に使える開発ツールというものはあまり一般的ではなかったため、取り組むのはそう簡単なことではなかった。しかし、Kylixが無償で入手できるようになれば誰でもちょっとしたツールを簡単に作成できる環境が整うので、今後はホビープログラマーも順調に増加していくことになるだろう。

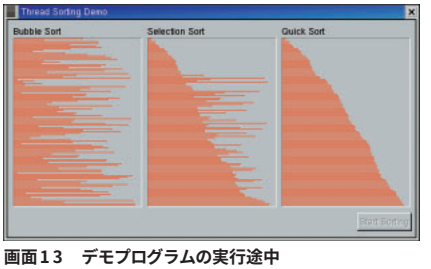



画面10　作成できるアイテムの一覧
“Web Server Application”というアイテムが用意されている点に注目したい。これを選ぶと、さらに「CGI Stand-alone Executable」と「Apache Shared Module (DSO)」のどちらかをさらに選択することになる。CGIにしておけば、Webサーバに依存しない汎用アプリケーションが構築できる。



画面11　アプリケーション実行例
標準で添付されているデモアプリケーションのプロジェクトを開き、実行してみたところ。テスト実行は、メインウィンドウの「Run」ボタンをクリックするだけで実行できる。


画面12　デモプログラムのソート実行前
このデモは、長さの異なるグラフを3種類の異なるアルゴリズムを使ってソートするもので、スレッドのデモでもある。




画面13　デモプログラムの実行途中
アルゴリズムの違いによってソートの実行時間が大きく異なることがわかる。なお、テスト機のCPUは650MHzのAthlonだが、アプリケーションの見た目の実行速度はかなり良好だった。

# Distribution

新着ディストリビューション

## Red Hat Linux ベータ版「Fisher」

バージョンナンバー7.0.90をもつFisherの配布が始まった。Fisherはカーネル2.4が採用されたRed Hat Linux次期バージョンのベータ版だ。4月にリリースが予定されている正規版の予告編として、Fisherのもつ新機能を見てみよう。

## Turbolinux Advanced Server 6

極限までのチューニング。大規模サイト用に特化したTurbolinux Advanced Server 6のカーネルには、LFSやExt3といった2.4カーネルの新機能が取り込まれている。バンドルされるHDE Linux controllerを使えば、高性能サーバをWebブラウザから簡単に管理可能だ。

## Miracle Linux for Samba Version 1.0

インストールしたらすぐにSamba。Oracleに特化したMiracle Linuxを開発するミラクル・リナックスが、今度はSambaに特化したディストリビューションをリリースした。オフィス文書をPDF化する機能をもち、Windows NTサーバからスムーズに移行可能だ。

## Miracle Linux for PostgreSQL Version 1.0

ミラクル・リナックスがリリースしたもう1つのディストリビューションは、オープンソースデータベースの代表格PostgreSQLに特化したバージョンだ。初期設定が済まされているので、インストール後すぐにデータベースと連携したWebサイトを構築可能だ。

## Meister Linux Mandrake 7.2

レーザーファイブによって日本語化されたフランスのディストリビューションMandrakeは、Windows（FAT）領域へのインストールをサポートし、ルック・アンド・フィールがWindowsに近いKDE 2.0.1を採用して、Windowsユーザーが Linuxへ移行しやすいように配慮されている。

# Red Hat Linuxベータ版「Fisher」

Linux最大ディストリビュータのひとつであるRed Hat, Inc.が次期バージョンのベータ版、コードネーム「Fisher」をリリースした。Fisherはインストーラやカーネルなどの基本システムが大幅に変更されたほかに、Intelが開発中の64ビットプロセッサItaniumを新たにサポートしている。

## より多機能になったインストーラ

Fisherのインストーラ（**画面1**）は、Red Hat Linux 7のものと比べて外見が若干異なるほかに、より多くの機能を備えている。

日本でも高速回線環境が整いつつある昨今、家庭内LANのゲートウェイとしてLinuxマシンを使っているユーザーが増えている。しかし、常時接続することは、外部からアタックを受ける可能性が増えることを意味する。

このあたりの事情が考慮されたのか、新しいインストーラでは、インストール中にセキュリティの設定が可能になっている（**画面1**）。たとえば、他のマシンからFisherを操作する場合は、クラックの対象となりやすいTelnetのチェックを外してSSHを使う、という具合に設定できる。

このほかにも、ISOイメージファイルを使ってのハードディスクインストールが新たにサポートされている。

## カーネル2.4の採用

Fisherでは基本システムとして、カーネルに2.4.0.99、Cライブラリにglibc2.2.1、XFree86 4.0.2、Cコンパイラにgcc 2.96-71が採用されている。

Fisherのカーネルはgcc 2.96でコンパイルされており、カーネルモードで動作するWebサーバプログラムTUXや、ギガビット用のネットワークアダプタIntel Pro/1000を利用可能になっている。

## 豊富なGUI設定ツール

FisherにはApache、BIND、プリンタのGUI設定ツールが新たに収録されている。たとえば、Apacheの設定ツール（**画面2**）では、クライアントの最大接続数やバーチャルホストの基本的な項目の設定が可能だ。このほかにも、ADSLサービスを利用するためのPPPoE（PPP over Ethernet）クライアントを利用できるので、LinuxでADSL接続する際に役立つだろう。

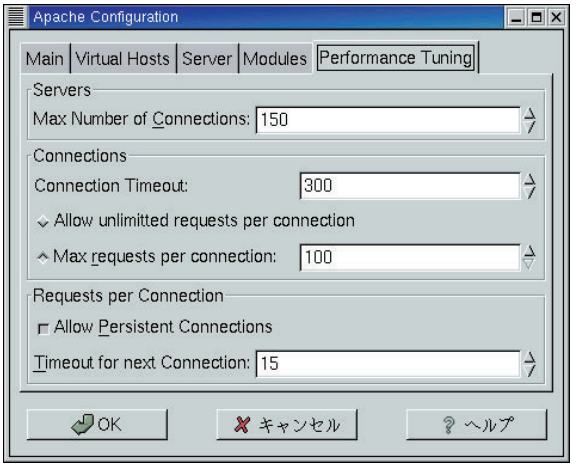
また、Debianに実装されているaptに相当するup2dateというパッケージアップデートツールが利用可能になっている。これにより、Red HatユーザーもRPMパッケージをインストールする際に、ライブラリの依存関係などで悩まなくてもよくなるだろう。

このように多くの新機能を搭載するFisherはまだ開発中のベータ版である。4月にリリース予定の正式版をバグの少ないものにするためにも、「バグレポートの宛先」を参考にして、どんどんバグレポートを送っていただきたい。

**バグレポートの宛先**

Fisherは正式リリース前のベータ版で、バグ出しを目的として公開されています。Fisherには少なからぬバグの存在が予想されるため、日常使用するデスクトップやサーバ用途のご使用はお勧めできません。Fisherの評価中にバグを発見した場合は、下記の宛先まで使用環境などを添えてバグレポートを送ってください（レポートに対する回答はありません）。
英語：http://bugzilla.redhat.com/bugzilla/
日本語：testers-jp-list@jp.redhat.com


**画面1　装いを新たにしたインストーラ**
Red Hat Linux 7と比べて設定項目に大きな変更はないが、ファイヤウォールマシンとしての設定が可能になっている。




**画面2　ApacheのGUI設定ツール**
このツールを使ってWebサーバApacheの基本的な設定ができる。このほかにもDNSサーバのBINDとプリンタのGUI設定ツールが収録されている。

# Turbolinux Advanced Server 6

ターボリナックスジャパンは大規模サイト用に開発したTurbolinux Advanced Server 6（以下、Advanced Server）を2月16日にリリースした。

ホライズン・デジタル・エンタープライズ（HDE）が開発する高性能Linux管理ツールHDE Linux Controller（以下、Linux Controller）のTurbolinux版をバンドルして、6万9800円で販売される（収録物は**表1**を参照）。

ユーザーはメールや電話などで、インストールとサーバの基本的な設定に関するサポートを90日間で3件受けられる。

## サーバ用に拡張されたカーネル

Advanced Serverは、基本システムにカーネル2.2.18、glibc2.1.3、XFree86 3.3.6が採用され、カーネルはサーバに必須の機能を2.4カーネルからバックポートして拡張されている。

### ■LFS（Large File Summit）

x86マシンで2.2系のLinuxを動作させる場合、Ext2ファイルシステムに2Gバイトを超えるファイルを作成できない。だが、LFSを採用したAdvanced Serverは最大4Tバイトのファイルを扱えるように拡張されているので、データベースのプラットホームとしても安心して使用できるだろう。

### ■LVM（Logical Volume Manager）

LVMは複数のパーティションとハードディスクをまとめて、仮想的に1つの大きなドライブを作る機能だ。

たとえば、マシンに2台のハードディスクが接続されている場合、IDEディスクにある/dev/hda1（20Gバイト）と、もう一方のSCSIディスクにある/dev/sda3（15Gバイト）という2つのパーティションをまとめて35Gバイトのドライブとして使える。

この要領で最大1P（＝1000T）バイトの領域を作成可能だ。ドライブのサイズ変更も容易なので、LVMによって柔軟にハーディスクを使えるだろう。

### ■Ext3

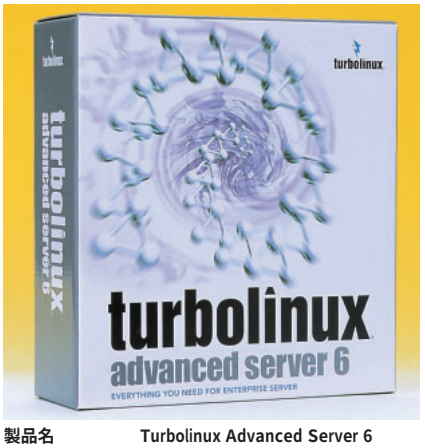
LFSやLVMを利用してExt2上に大きなパーティションやファイルを作成した場合、今度はシステム不正終了時のファイルシステムチェックにかかる時間が悩みのタネになる。

Ext3は、この問題を解消するジャーナリング機能を、現在標準のExt2に追加した新しいファイルシステムである。現在ジャーナリングファイルシステムとしてはReiserFSのほうが一般的だが、Ext2ファイルシステムを直接ReiserFSに変換できないので、Ext2との相性は良くない。

その点Ext3ファイルシステムはExt2との互換性が高く、稼動中のExt2をtune2fsというプログラムでExt3に変換できるのが嬉しい。ただ、0.0.5というバージョンを見てもExt3は安定稼動が望まれるシステムに導入するのは早計だろう。

### ■ReiserFS

ReierFSもExt3と同様ジャーナリング機能を備えるファイルシステムだ。ReiserFSはKondara、SuSE、Mandrakeといったディストリビューションで採用されており、カーネル2.4.1から正式に取り込まれているので、Ext3に比べて動作実績があるといえ



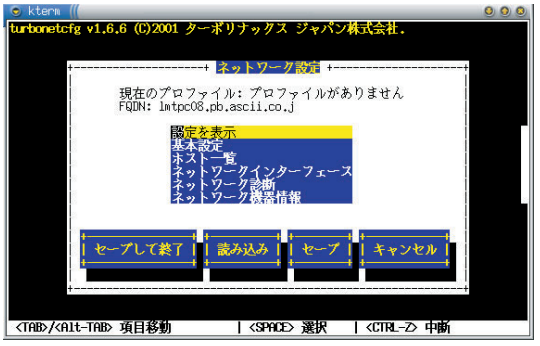

製品名　Turbolinux Advanced Server 6
価格　6万9800円
問い合わせ先　ターボリナックス ジャパン株式会社
03-5766-1660
http://www.turbolinux.co.jp/

| 収録物 | 概要 |
|---|---|
| インストールCD | Advanced Serverのインストール用CD |
| ソースCD | ソースパッケージを収録 |
| コンパニオンCD | Linux Controller Turbolinux Editionなどを収録 |
| インストールガイド | インストール用のマニュアル |
| サーバ構築・運用ガイド | サーバアプリケーションの設定方法を解説 |
| Linux Controllerマニュアル | Linux Controllerを使ったサーバ設定方法を解説 |

表1　Advanced Serverの収録物

| 新機能 | 概要 |
|---|---|
| LFS | サイズが2Gバイトを超えるファイルをサポート |
| LVM | 複数のドライブとパーティションを仮想的に1つのドライブとして扱う |
| Ext3 | Linux標準のExt2にジャーナリング機能を追加した新しいファイルシステム |
| ReiserFS | Linuxで最も実績のあるジャーナリングファイルシステム |
| kparam | いくつかのカーネルパラメータを動的に変化させるツール |

表2　Advanced Serverが備える新機能




**画面1　Turbolinux独自の設定ツール**
**HDE Linux Controllerがバンドルされているのでwebブラウザから管理できるが、Turbolinuxの独自ツールはターミナルでも使えるので、もしものときに安心だ。**

る。ただ、Ext2を直接ReiserFSへ変換できないので、Ext3と比べて導入に若干手間がかかるだろう。

### ■kparam

Linuxには動作中のカーネルの様子を表す/procという特別なファイルシステムが存在する。たとえば「ls /proc」として表示される数字は実行中のプロセスIDを示し、数字のディレクトリ内にはそのプロセスの情報が格納されている。/proc内の適当な数字のディレクトリへ移動して「cat cmdline」としてみれば、なんとなく/procの意味がわかるだろう。

さて、kparamは/proc内にあるいくつかのカーネルパラメータを、Linuxが動作中に調整するプログラムだ。/etc/sysconfig/に置かれる設定ファイルをもとにしてカーネルパラメータを設定する。Advanced ServerにはOracleデータベースのプラットホームに適した設定ファイルが用意されている。

## Webブラウザでサーバ管理

Advanced Serverには、Turbolinux独自の設定ツール（**画面1**）のほかに、HDEが開発するLinux設定ツールLinux Controller Turbolinux Editionがバンドルされる（**画面2、3**）。

Linux ControllerはWebブラウザからLinuxを操作する高機能なツールで、Advanced Serverにバンドルされる Linux Controller Turbolinux Editionは、バックアップ管理機能などに制限があるものの、製品版Linux Controllerの上位クラスLinux Controller 2.0 Professional Editionに相当するバージョンだ。

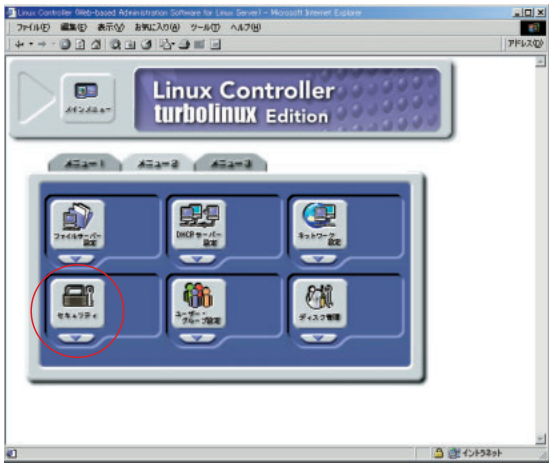
Linux Controllerのトップページ（**画面2**）にある「セキュリティ」から「パケットフィルタの一般設定」を選択すると、各サービスに対するパケットフィルタリングを設定できる（**画面3**）。パケットフィルタリングはipchainsコマンドを使って設定するのが一般的だが、日本語のメニューを読みながら設定できるのが初心者に嬉しい。

このほかにも、WebサーバやDNSなどのネットワーク設定はもちろん、パッケージ管理やディスク管理にも対応しており、Webブラウザからほぼすべてのサーバ管理作業が可能だ。

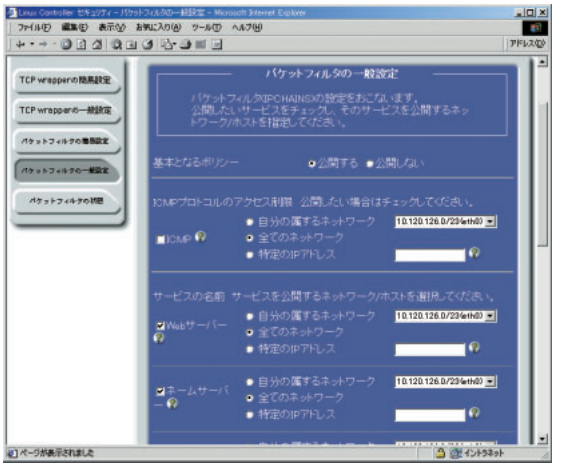
さて、Linux Controllerの最上位バージョンのEnterprise Editionは、Professional Editionがインストールされた多数のLinuxマシンを一括管理する機能をもつ。

このため、複数台のAdvanced ServerをWWW、DNS、Mailなどのサービスごとに稼動させ、Enterprise EditionがインストールされたLinuxマシンを仲介して、多数のサーバマシンで構成される大規模サイトをWebブラウザから一括管理することも可能だ。

このように、基本性能を高めるためのチューニングが施され、スケーラビリティの高い設定ツールを備えるAdvanced Serverは、大規模サイトで高いパフォーマンスを発揮するディストリビューションといえるだろう。




**画面2　Linux Controllerのトップ画面**
**Advanced Serverには、Professional Editionに相当するHDE Linux Controllerと詳細なマニュアルがバンドルされる。**




**画面3　パケットフィルタリングの設定画面**
**通常ipchainsを使って設定するパケットフィルタリングを、Webブラウザで設定する。**

# Miracle Linux for Samba Version 1.0

リレーショナルデータベースOracleに特化したMiracle Linuxを開発するミラクル・リナックスが、Samba用に特化したMiracle Linux for Samba Version 1.0（以下、Miracle for Samba）を3月1日にリリースした。

サーバ構築用のドキュメントとインストールCDなどが収録され、1万9800円で販売される。ユーザーは、インストールに関してFAXとメールで30日間、Sambaの起動と設定に関してメールで1年間3件までサポートを受けられる。

Miracle for Sambaは、カーネル2.2.16、glibc2.1.3、現時点で最新バージョンのSamba 2.0.7-ja2.2を採用しており、インストール直後からSambaサーバとして使えるのが特徴だ。

## PDFプリントサーバ

Miracle for Sambaは、Windowsマシンで作成したオフィス文書をPDFファイルとして出力する機能を備えている。

WindowsクライアントからMiracle for Sambaへアクセスすると**画面1**のように表示される。ここに表示される「PDFWRITER」をWindowsのコントロールパネルにある「プリンタ」（**画面2**）へドラッグすれば、PDFプリントサーバとして利用可能だ。

このあとWindowsで作成したExcelやWord文書を「PDFWRITER」を指定して印刷すれば、Sambaの共有フォルダにPDFファイルが作られるというわけだ。

この機能は、オフィスファイルを、Windows用のPostScript（PS）プリンタドライバを通してPSファイルに変換して、さらにps2pdfでPDFファイルに変換することで実現されている。

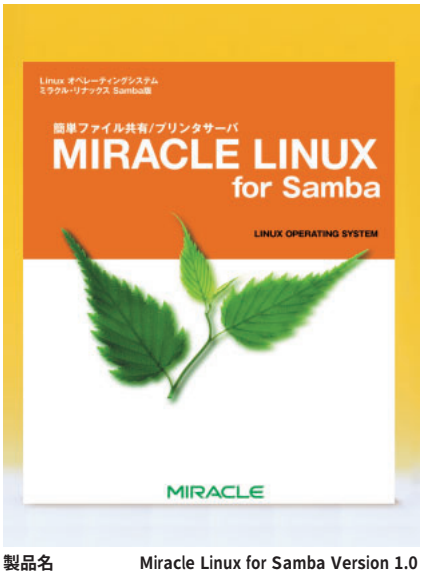
PDFファイルはWindowsやMac OSはもちろん、Linuxでも利用できるので、この機能を利用してPDFファイルにしておけば、作成した文書を有効に使えるだろう。

## NTサーバからスムーズに移行

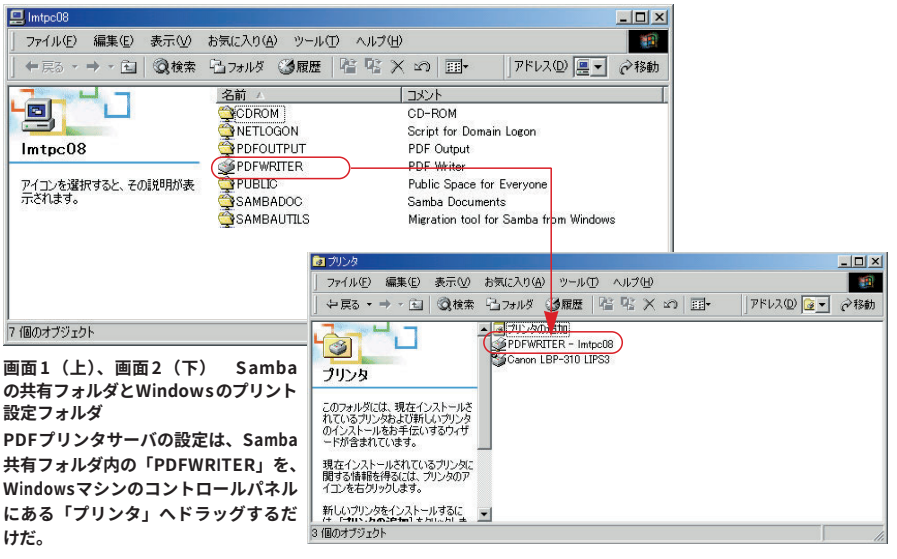
Sambaの共有フォルダ（**画面1**）にある「SAMBAUTILS」には、NTサーバに作成済みのユーザー情報を抜き出すためのpwdump.exeというツールが用意されている。Miracle for Sambaは、SambaとLinuxで共通のユーザー情報を使うように設定されているので、このツールを使えばNTサーバにあるユーザー情報を簡単にSambaへ移行できるだろう。

また、Webブラウザで操作する設定ツールSWATは、インストール直後からブラウザでアクセス可能で、SWATのトップ画面からは豊富な日本語ドキュメントを参照できるように設定されている。

このように、細かい箇所があらかじめ設定済みのMiracle for Sambaは、Linux初心者でもファイルサーバを簡単に構築できる製品である。




製品名　Miracle Linux for Samba Version 1.0
価格　1万9800円
問い合わせ先　ミラクル・リナックス株式会社
03-5562-8300
http://www.miraclelinux.com/




画面1（上）、画面2（下）　Sambaの共有フォルダとWindowsのプリント設定フォルダ
PDFプリンタサーバの設定は、Samba共有フォルダ内の「PDFWRITER」を、Windowsマシンのコントロールパネルにある「プリンタ」へドラッグするだけだ。

# Miracle Linux for PostgreSQL Version 1.0

Oracleデータベースに特化したMiracle Linuxを開発するミラクル・リナックスが、オープンソースデータベースの代表格であるPostgreSQL専用のディストリビューションMiracle Linux for PostgreSQL Version 1.0（以下、Miracle for PostgreSQL）をリリースした。

インストールCDやサーバ運用ガイドなどが収録され、1万9800円で3月1日から発売されている。ユーザーは、インストールに関してFAXと電話で30日間、PostgreSQLの起動と設定に関して1年間3件までのサポートを無償で受けられる。

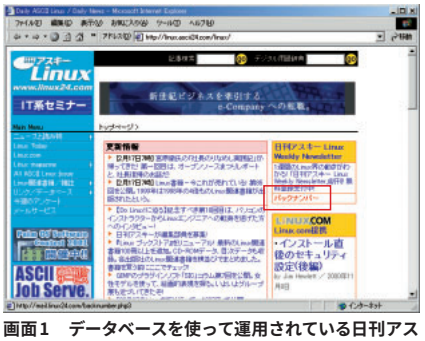
## データベースと連携するWebサイト

Miracle for PostgreSQLは、カーネル2.2.16、glibc2.1.3、PostgreSQL 7.0.3、PHP 3.0.18-i18n-ja-2、Apache 1.3.14で基本システムが構成されている。収録されるPostgreSQLにはjumbo-20010117パッチが適用され、PHPは国際化されたバージョンが収録されているので日本語への対応は万全だ。

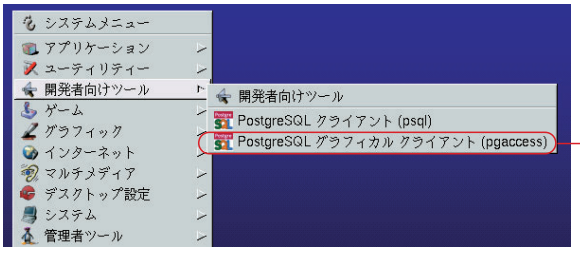
個人のホームページでは、掲示板や日記などの動的コンテンツの作成にCGIが使われるケースが多い。しかし、コンテンツが頻繁に変更されたり多くのデータを持つサイトでは、データベースを使ってファイルを管理し、PHPというスクリプト言語を仲介してそれらをコンテンツに反映させるのが有効だ。

たとえば、日刊アスキーLinuxのトップページ（**画面1**）にある「バックナンバー」をマウスでポイントすると、backnumber.php3というPHPスクリプトにリンクされているのがわかる。「バックナンバー」をクリックして表示される記事の一覧は、PostgreSQLとPHPを使って動的に作成されたものだ。

Miracle for PostgreSQLは、このようなWebサイト構築用に設定済みなので、インストール直後からデータベースと連携させたWebサイトを構築可能だ。GNOMEメニューにはPostgreSQLを操作するツールが登録済みだ（**画面2**）。




**画面1　データベースを使って運用されている日刊アスキーLinuxのWebサイト**
**拡張子.php3を持つファイルがPHPを利用しているサイトの目印だ。「バックナンバー」をクリックして表示されるのは、PHPが生成したHTMLページである。**



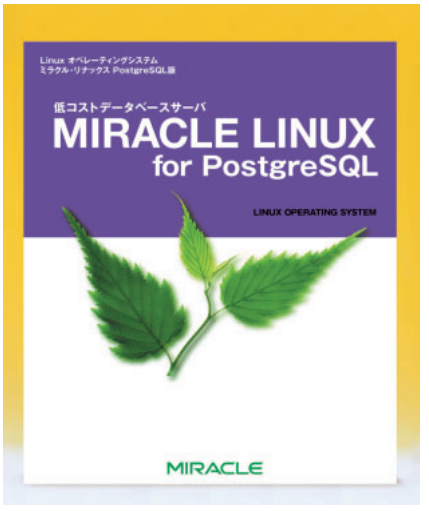


**画面2（左）、画面3（右）　Miracle for PostgreSQLのGNOMEメニュー**
**PostgreSQLに用意されるTcl/Tkインターフェイスを利用したpgaccess（右）はGNOMEパネルに登録済みだ。pgaccessは、データの検索や作成、GUIアプリケーションを作成できる高機能なツールだ。**

## WindowsでもJavaでも

PostgreSQLへのアクセス手段はPHPだけではない。いくつかのデータベースには、共通のアクセス手段を提供するためのODBC（Open Database Connectivity）というインターフェイスが用意されている。PostgreSQL用のODBCを利用すれば、AccessやExcelといったアプリケーションからもアクセスできるので、より多くのユーザーが利用できるデータベースを構築可能だ。

ODBCのほかにも、JavaやPerlなどからデータベースを利用するためのドライバ群が収録されている。データベースを利用したいがOracleを必要とするほどの規模ではない場合に、PostgreSQLの初期設定が不要なこのディストリビューションで開発を始めてはどうだろうか。




製品名　Miracle Linux for PostgreSQL Version 1.0
価格　1万9800円
問い合わせ先　ミラクル・リナックス株式会社
03-5562-8300
http://www.miraclelinux.com/

# Meister Linux Mandrake 7.2

LASER5 Linuxでおなじみのレーザーファイブから、Meister Linux Mandrake 7.2（以下、Mandrake 7.2）が2月23日にリリースされた。これまでMeisterシリーズは、五橋研究所から発売されていたが、今後はレーザーファイブよりリリースされる。

MandrakeはフランスのMandrake Soft SAが開発するディストリビューションで、今回紹介するMandrake 7.2は、レーザーファイブが独自に日本語化して、商用ソフトと日本語マニュアルをバンドルしたバージョンである。前バージョン7.1と比べて、バンドルされる商用ソフトは大幅に増えている。

## KDE 2.0.1を採用

Mandrake 7.2では、基本システムとしてカーネル2.2.17、glibc2.1.3、XFree86 4.0.1、gcc 2.95.3が採用されている。カーネルは多くのパッチで拡張されており、Mandrake 7.2のReiserFSへのインストールや、i815チップセットの利用が可能だ。

また、オフィスツールKOfficeを搭載したKDE 2.0.1の採用に加えて、Windows領域（FATファイルシステム）へのインストールもサポートするなど、WindowsユーザーがLinuxへスムーズに移行できるように配慮されている。

## GUI設定ツールでADSL接続

Mandrake独自のGUI設定ツールDrakConf（**画面1**）はさらに拡張されており、一般家庭へ普及しつつあるADSLやCATVといった常時接続ネットワークを設定できるようになっている。

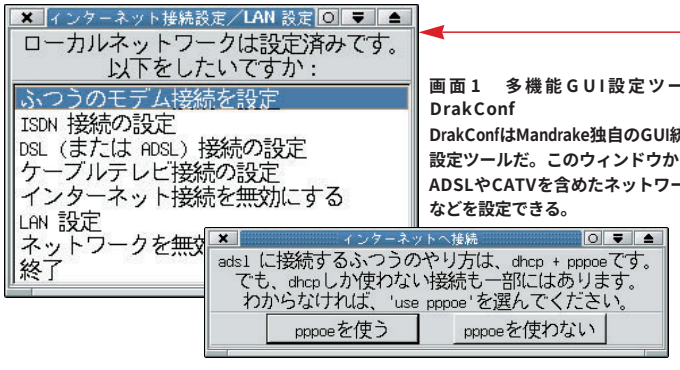
また、DrakConfを使えばMandrake標準のブートローダGRUBも設定できるので、GRUBに慣れていないユーザーでも快適なマルチブート環境を構築できるだろう。

インストール中にReiserFSを選べたり、簡単にADSL接続できたりと、新しいもの好きのユーザーに、特にワクワク感を与えるディストリビューションといえる。

| 収録物 | 概要 |
|---|---|
| インストールCD（2枚） | Mandrakeのインストール用CD |
| ソースCD | ソースパッケージを収録 |
| コントリビュートCD | あると便利な追加パッケージ集 |
| 日本語インストールマニュアル | 初心者にやさしい解説書 |
| 商用製品CD | |
| カーネル2.4関連パッケージ集 | 2.4カーネルへのアップグレードに必要なパッケージ集 |
| K筆 | Linuxで初めての年賀状ソフト |
| Wnn6 | UNIX系OSで定番の日本語入力プログラム |
| eWnn | 日本語エディタMuleから使う英文読書き支援ソフト |
| 翻訳魂 | 日英／英日翻訳ソフト |

**表1　Meister Linux Mandrake 7.2の収録物**
インストール用のCD（2枚）のほかに、レーザーファイブが販売するMeister Linux Mandrake 7.2には、コントリビュートパッケージを収録したCD、Wnn6などの商用ソフトを収録したCD、日本語のインストールマニュアルが付属する。


製品名　Meister Linux Mandrake 7.2
価格　6800円
問い合わせ先　レーザーファイブ株式会社
03-5818-6626
http://www.laser5.co.jp/





**画面1　多機能GUI設定ツールDrakConf**
DrakConfはMandrake独自のGUI統合設定ツールだ。このウィンドウから、ADSLやCATVを含めたネットワークなどを設定できる。

# Distribution ▶▶▶

## Miracle Linux with Oracleの新シリーズ発表

ミラクル・リナックスはリレーショナルデータベースOracleをバンドルしたMiracle Linux with Oracleシリーズの一環として、新たに3製品を3月中旬より出荷する。

このシリーズに新たに追加された製品は、

**Miracle Linux with Oracle8i Enterprise Edition**
**Miracle Linux with Oracle9i Application Server**
**Miracle Linux with Oracle Solution Pack**

の3製品で、すでに発売済みのMiracle Linux with Oracle8i Workgroup Serverとあわせて、with Oracleシリーズは合計4製品のラインナップとなる。

Miracle Linux with Oracle Solution Packは、ほかの2製品にバンドルされるOracle8i Enterprise EditionとOracle9i Application Serverの両方をバンドルする製品だ。

3製品ともデータベースサーバやアプリケーションサーバのセットアップの簡単さが特徴で、2月26日から先行受注が始まっている。

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

## Trubolinuxの次世代インストーラMongooseのスナップショット公開

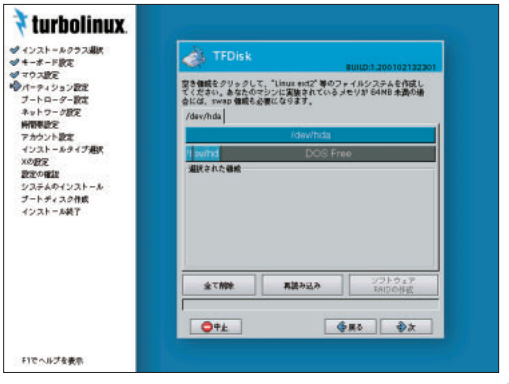
Trubolinuxの次世代インストーラとして開発中のMongoose（マングース）のスナップショットが届いた。

現行のTrubolinuxはテキストベースのインストーラを採用しているが、Mongooseの採用で初心者ユーザーがさらに使いやすいインターフェイスを提供することになる。

Mongooseはインターフェイスがグラフィカルになっただけでなく機能面もいくつか拡張されている。Mongooseは日本語やハングルを含む5ヵ国語の表示や、棒グラフでパーティション情報を表示しながらパーティションを編集するTFDiskを実装している。サーバ向けTrubolinuxのMongooseでは、このTFDiskを使って、Ext2ファイルシステムのほかに、ReiserFSとExt3ファイルシステムも作成できるとのこと。

このMongooseは、4月上旬～中旬に発売予定のTurbolinux Server 6.5から採用される予定だ。

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




Mongooseの画面は開発中のものです。

## NSA Security Enhanced Linux公開

NSA（アメリカ国家安全保障局）は、Linuxカーネルにセキュリティ拡張をほどこすSecurity Enhanced Linux（SE Linux）を公開した。

SE Linuxは、Linuxに実装されているリソースやアプリケーションへのアクセス制御を、さらに柔軟に設定するための拡張で、Red Hat Linuxのようなディストリビューションではなくカーネルパッチのような形態で配布されている。

同じような目的をもつRSBAC（Rule Set Based Access Control）for Linuxとともに、カーネル2.6や3.0などへの統合が、linux-kernelメーリングリストなどで議論されている。Security Enhanced LinuxはNSAのサイトからダウンロード可能だ。

**The National Security Agency（http://www.nsa.gov/selinux/）**

## Mobile Linux公開へ

CrusoeプロセッサでおなじみのTransmetaは、携帯Linuxマシン用に開発しているMobile Linuxの配布を開始する。配布ライセンスはGPLだ。

Mobile Linuxの配布開始は2月末をめどにしているので、本誌が発売される頃にはすでに配布が始まっているかもしれない。

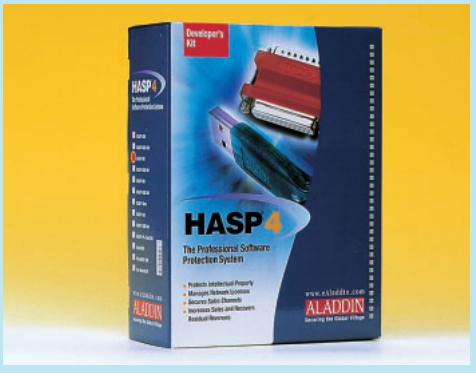
Mobile LinuxはIntelとTransmetaのCPUで動作するLinuxで、すでにGatewayや日立のインターネット専用情報端末で利用されている。省電力に優れた設計なので、Mobile Linuxを利用した小型ルータや携帯MP3プレイヤーなどを開発可能だという。

**Transmeta Corporation（http://www.transmeta.com/）**

# Products



## 商用アプリケーションを違法コピーから守るプロテクションシステム




### HASP

**Mac用の高価なDTPやグラフィックスソフト、建築／回路設計用CADソフトでは、コピープロテクトのためにプリンタポートなどに付けるハードウェアプロテクトキーが付属している。このたび、ALADDINが開発しているプロテクトキー「HASP」がLinuxに対応した。**

| | |
|---|---|
| 製品名 | HASP |
| 価格 | HASP開発キットは無料 |
| 問い合わせ先 | 株式会社アラジンジャパン<br>TEL 0426-60-7191<br>http://www.aladdin.co.jp/ |

アラジンジャパンから、違法コピーや不正使用を防ぐためのソフトウェアプロテクションシステム「HASPファミリー」が発売されている。アラジンジャパンは、イスラエルのアラジン・ナレッジシステムズの日本法人で、HASPのほかにインターネット向けセキュリティキー「eToken」やファイヤウォール統合アンチウイルス「eSafe Gateway」などのセキュリティ製品を販売・サポートしている。

HASPは、通称「ドングル」と呼ばれているハードウェアプロテクトキーで、**写真1**のようなデバイスをマシンのパラレルポートやUSBポートに接続する。ソフトウェアからHASPが接続されていることを検出し、HASPなしでは動作しないようにプロテクトを行うことができる。なお、LinuxでのUSB正式サポートはカーネル2.4からのため、いまのところHASP USBはLinuxでは利用できない。

HASPファミリーには、数百バイトのメモリを内蔵し、複数のアプリケーションで個別のアクセス管理を行ったり、デモ版やソフトウェアレンタル、リモートからアップグレードするなどの機能を持たせたHASP4 M1や、ネットワーク上で同時使用数を管理することができるHASP4 Net、PCカード用のHASP4 PC-CARD、Mac用のMacHASPなどが用意されている。

HASPは専用のASICチップを内蔵していて、ソフトウェアとの通信も暗号化されているため、リバースエンジニアリングは不可能である。

HASPのプロテクトは、大きく分けて2種類ある。ひとつは、HASP APIを利用するもので、HASPライブラリ（Windowsの場合にはDLL）が用意されている。これをリンクすることでユーザープログラムから、hasp（）

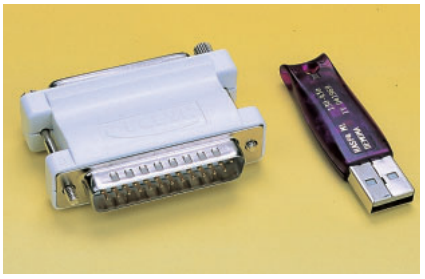


写真1　HASPプロテクションキー
左側が25ピンパラレルポートに接続するHASP4 M1、右側がUSBポートに接続するHASP4 USB。ただし、現在のところHASP4 USBはLinux環境未対応。


写真2　HASP開発キットの内容
開発キットには、350ページほどのプログラマーガイドとHASPソフトウェアCD、HASPデモキーが含まれる。このCDでWindowsとLinux用の開発が行える。

ルーチンをコールして、HASPキーと通信し、戻り値をチェックすることで、プロテクトが可能になる。

もうひとつはエンベロープで、HASP4から加わった機能だ。プログラムファイルをエンコードするため、デバッガなどで読み込んでトレースしたり、解析したりすることができなくなる。

この2つを併用することで、より強力なプロテクトが実現できる。ただしLinux用には、HASP APIだけが提供されており、エンベロープ機能は利用できない。

## Linuxに対応したHASP開発キット

ソフトハウスなどのアプリケーション開発者向けのHASP開発キット（**写真2**）は無料で提供されている。これを利用して、アプリケーションソフトへプロテクトする実験、動作検証を行う。

標準で含まれるCD-ROMには、WindowsとLinuxに対応するプログラムが入っている(Mac用は別のCDで提供される)。Windows用は、専用のインストーラとGUIのツールが付属するが、Linux用のモジュールは、hasplinux100.tgzというtar+gzipでアーカイブされたファイルで提供されている。

そのファイルを展開すると、カーネルのバージョンごとのドライバと、テストツール、デモプログラムのソース／バイナリファイルが含まれている。

それでは実際にLinuxで動作させてみよう。パラレルポート用HASPデモキーをLinuxマシンに装着する。パラレルポートにアクセスするために、

```
# modprobe parport_pc
```

として、Linuxに標準で含まれるパラレルポートドライバをロードする。

次に、HASPドライバモジュールをロードする。対応するカーネルは、2.0.38、2.2.14（-smp）、2.2.14-5.0（-smp）、2.2.16（-smp）、2.2.17（-smp）で、それぞれに対応したデバイスドライバモジュールaksparlnx.oが用意されている。

```
# insmod aksparlnx.o
```

そして、デバイスファイルを作成する。

```
# mknod /dev/Hardlock c 42 0
# chmod a+rw /dev/Hardlock
```

ここまでの作業で、準備は終了である。デモプログラムhaspdemoによって動作を確認することができる。ソースプログラムhaspdemo.cが同じディレクトリにあるので、これを参考にして、アプリケーションにHASP APIを組み込んでいけばよい。

付属する約350ページのプログラマーガイドには、HASPの概念からプロテクションのテクニック、APIサービスの使用方法が、日本語で詳細に記述されている。特にプロテクトされたプログラムへの攻撃（コピー外し）の方法と対策の例が数多く載っていて、それらを組み合わせて併用することで高度なプロテクションが実現されていることがわかる。

開発キットに含まれるHASPデモキーには、開発用の固定パスワードが組み込まれている。アラジンジャパンに注文することでソフトウェア開発者ごとに固有のパスワードが記録されて出荷される。そのキーを元に実際の製品に組み込み、ソフトウェアにHASPを同梱することで、その製品を購入した人以外がそのソフトウェアを使うことはできなくなる。なお、注文は最小5個から受け付けている。

HASPのAPIはOSによらず共通なので、プラットホームを移植することによる手間は増えない。HASPによってコピープロテクトが行えるようになったことで、MacやWindows用のDTP、CAD、グラフィックスソフト、業務用ソフトなどの本格的なアプリケーションが移植されることが期待できるのではないだろうか。

ただし、Windows用に比べるとLinux用の完成度はあと一歩といったところだ。先ほどの準備の部分をユーザーが手作業で行うようでは使い勝手がよくないので、カーネルのバージョンに合わせたデバイスドライバの自動インストールや、デバイスファイルの作成などをサポートするユーティリティの整備が望まれる。


# インターネット便利箱 for Linux

**インターネットへ常時接続するサービスが普及しつつある中で、サーバ用ではなくデスクトップ用の便利なツールが登場した。Webサイトから効率的に情報を収集するために、情報検索、自動収集、定点観測という3つの機能を備えたユーティリティだ。**

| | |
|---|---|
| 製品名 | インターネット便利箱 for Linux |
| 価格 | 5800円 |
| 問い合わせ先 | 株式会社クエスト<br>TEL 03-5789-4909<br>http://www.qst.co.jp/ |

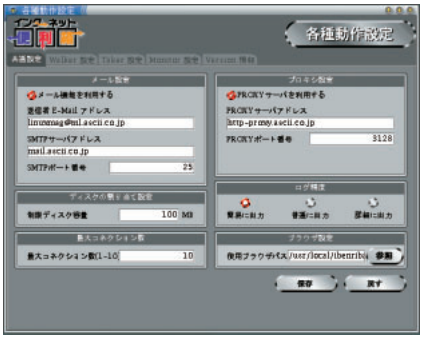
「インターネット便利箱 for Linux」は、インターネット上の情報を収集する作業を支援するユーティリティソフトだ。NTTコムウェアが開発している「Info Σ 2100」シリーズ（http://infosigma.nctech.nttcom.ne.jp/）のLinux版で、販売はクエストが行っている。

複数の検索エンジンに対して一括検索を行う情報検索機能「Walker」、Web上の必要な情報をダウンロードする自動収集機能「Taker」、指定したサイト（URL）を巡回し、情報更新を検出する定点観測機能「Monitor」の3つの機能を持っている。

インターネット便利箱は、Javaで書かれていて、X Window System上で動作する。Javaのランタイムである JRE 1.2.2を先にインストールする必要があるが、JRE自体も製品パッケージに含まれていて、インストールに際してはシェルスクリプトを実行するだけで簡単に行える。

起動すると、**画面1**のようなメニューが表示される。まず、設定ボタンを押して、動作設定を行う。共通設定（**画面2**）では、メールやプロキシサーバ、ディスク割り当て容量などを設定する。また、Walker設定などでは、結果を格納するディレクトリやインターネットへの接続のパラメータ（並列処理数、リトライ回数、間隔、タイムアウト時間）などを設定する。


画面1　インターネット便利箱の起動メニュー




画面2　各種動作設定の共通設定
3つの機能に共通する設定を行う。

## 見つけたい情報を複数の検索エンジンで一括検索

Walkerでは、検索したいキーワードを入力して検索ボタンを押すだけで、複数の検索エンジンを同時に使用（メタ検索）して情報を収集することができる。Yahoo! Japanやgooなど国内で14種類、AltaVistaなど海外では11種類の検索エンジンが登録されていて、自由に組み合わせて選択が可能である（**画面3**）。

検索サイトをひとつひとつ回って情報を見つけ出すには、手間と時間が掛かり、情報の取得漏れも発生しかねない。メタ検索で得られた結果は、すべてをまとめて表示することも、検索エンジンごとに表示することもでき、すべてを表示する場合には、重複する結果（同一サイト）を排除して表示するようになっている（**画面4**）。

もちろん、AND検索やOR検索などの検索式を利用できる。検索サイトによって検索式の指定方法が異なる場合があるが、その違いを意識しないで利用できるのは便利だろう。

検索結果は自動的に保存されるため、あとで何度も検索し直す必要が


画面3　Walker（情報検索）の検索実行画面
国内、海外の多くの検索エンジンが登録済み。


画面4　Walkerの検索結果表示
複数の検索エンジンでの検索結果を整理して、並べて見ることができる。

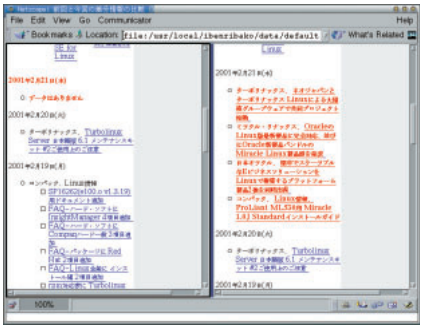
画面5　Taker（自動収集）の収集設定
Webサイトから一括して情報を収集する。情報の収集レベルによって階層を設定する。

画面6　Takerの収集結果画面
収集したURLごとにツリーで表示される。HTMLファイルを選び、Netscapeで閲覧できる。

画面7　Takerの詳細設定
収集するファイルの条件や種類を設定する。収集終了を通知するメールも送信可能。

画面8　Monitor（定点観測）の観測実行画面
特定のWebサイト（URL）を監視して、更新情報を取得することができる。

画面9　Monitorの観測結果一覧
更新状況が表示されるので、見たい情報を選ぶ。取得したページの差分の表示も可能。



画面10　Netscapeで前回と今回の差分を比較
差分比較ボタンを押すと、左側に前回分、右側に今回分を並べて、違う部分を赤で表示される。

ない。また、検索結果のURLに接続確認し、すでに存在しないWebページを検索結果リストから除外する機能も持っている。

## Webサイトの情報をローカルディスクに蓄積

Takerは、指定したWebページのHTMLを、そのままハードディスクに収集する。複数のページを登録可能で、指定したWebページからリンクされているページも、何階層まで（外部リンクは1階層だけ）を取り込むかの収集レベルを設定できる（**画面5**）。

ニュースサイトや情報サイトなどを収集しておけば、ダウンロード待ちをすることなく次々にページを見ていくことができる。収集した結果は、**画面6**のようにURLごとに表示されるので、HTMLファイルを選択して表示ボタンを押すと、Netscapeが起動されて見ることができる。

ファイルの日付やサイズの制限を設定することもでき、HTMLファイル以外に、tarやzipなどのアーカイブ、画像、音声、PDFやテキスト文書、そして収集したいファイルの種類を指定して取り込むことも可能である（**画面7**）。

## Webページの更新情報を検出

Monitorは、あらかじめ更新状況を知りたいサイトを登録しておくことで、そのページに定期的にアクセスして更新情報をチェックする。オークションサイトの価格情報や気になる情報が掲載されているWebページなど、常に最新の情報が必要な場合に活躍する（**画面8、9**）。

監視したいWebページのURLとタイトルを入力して、観測開始ボタンをクリックすれば、すぐにチェックされる。また、観測サイクルとして、毎日／毎週／毎月の何時からという設定と、何時間ごとにという設定が可能だ。

更新された箇所を強調表示する差分表示や、更新前と更新後を2つ並べて表示する差分比較（**画面10**）も行える。

また、Taker、Walker、Monitorのそれぞれの作業が終了したことを、メールで知らせるように設定できる。動作結果の概要も書かれているので、必要に応じてiモード携帯電話などに通知すれば便利に使えるだろう。

この類のソフトは、Windows用としては販売されていたが、Linux用としては初めてだ。Linuxをサーバではなくデスクトップで利用するユーザーにとって、有意義なソフトといえるだろう。

## LinuxとWindowsでつくるLAN環境

# 入門・Linux LANの設定

**TCP/IP LANはじめの一歩。**
**Windows環境を便利にするSambaサーバを構築しよう。**

LANを導入してみませんか？　Windowsクライアントを強力にサポートするLinuxサーバの作り方お教えします。
LANでできること、必要な機材、TCP/IPのキホンを学んだら、Windowsをいっそう便利にするLinuxサーバを作りましょう。Linuxマシンにフリーソフトの Sambaをインストールすれば、Windowsで使えるファイル／プリントサーバに早変わり！　インターネットだって家族みんなでつなげちゃう！

*illustration : Aki*


Linux

# WindowsとLinuxでLANを構築しよう

文：編集部／Text：Linux magazine

みなさんはLANを使っているだろうか。職場で、あるいは家庭にLANを導入しているという方も多いだろう。本特集では、LANの構築をしたことがない読者にもわかるように、WindowsとLinuxが混在するLANの作り方を解説する。日常の作業をWindowsで行っている環境にLinuxサーバを置くことで、より便利になること間違いなしだ！

## LANってなんだろう

そもそもLANとはどのようなものだろうか。LANはLocal Area Networkの頭文字をつないだもので、ビル内など、比較的狭い範囲のコンピュータをつないだネットワークのことを指す。

LANを構築するためには、ソフトウェア、ハードウェア両方の対応が必要となる。数年前までは、PCの処理能力がきわめて低かったため、LANといえばワークステーションで作るものと相場が決まっていたが、PCの能力向上とソフトウェアサポートの充実とともにPCをLANに接続することが多くなった。

また、Windows NTやWindows 95は、OSがLANに対応したため、それまで必要だったサードパーティ製のLAN接続ソフトが不要になり、LANの普及に拍車をかけた。Linuxは、Windowsに先だって非常に早い時期からLANに対応しており、さまざまなネットワークで動作するプログラムが数多く開発されている。

このように、PCの世界にもLANが普及し、ハードウェアの価格も劇的に低下した。

今日では、2台以上のPCがあれば、なにはともあれLANでつなぐのが常識といっても過言ではないほどだ。

## LANでできること

さて、LANで複数のコンピュータをつなぐといったいナニが便利になるのだろうか？　LANでできることは非常にたくさんあるのだが、今回は基本的な使い方に的を絞って説明する。

### ■ファイル共有

複数のPCがある場合に、それぞれのPC間でデータを受け渡したくなったらどうすればよいだろうか？　このような場合、フロッピーディスクにデータをコピーして、ほかのPCで読み出すという方法がポピュラーだった。しかし、1枚のフロッピーディスクに収まらないような大きなデータを移すには、データを分割して複数のフロッピーディスクに書き込むか、MOやCD-Rなどの大容量リムーバブルメディアを使わなければならない。こんなとき、LANでつながっていればPC間でデータを送るのも簡単だ。転送速度もフロッピーディスクよりもずっと速い。

Linuxマシンをファイルサーバにして、ワープロや表計算ソフトなどのデータはこのサーバに保存して一元管理すれば、バックアップの作成も楽になる。Windows NT、Windows 95以降では簡単な設定でマシン同士をつなぐことができるが、ファイルサーバとして使うにはWindows 95、98、Meでは信頼性に疑問が残るし、Windows NT、2000では高価なサーバ版を用意しないとライセンス上問題がある。Linuxなら信頼性も十分だし、ライセンスの問題も発生しない。

Linuxでファイル共有を行うには、NFS、FTP、Sambaなどが利用できる。本特集では、Windowsとつなぐのに便利なSambaというサーバアプリケーションを使ってLinuxサーバを構築する。

### ■プリンタ共有

LANを介して1台のプリンタを共有することも可能だ。プリンタの共有方法には大きく分けて2つの方法がある。ひとつは、プリンタを直接ネットワークに接続するというものだ。比較的高級なページプリンタには、ネットワークインターフェイスが内蔵されていたり、別売りオプションとして用意されていることが多い。また、周辺機器メーカー各社は、プリンタのパラレルコネクタに接続するプリンタサーバを販売している。これを使うと、ネットワークに対応していないプリンタでもイーサネット経由で印刷できるようになるのだ。プリントデータをスプールするバッファを持たない製品なら、数千円で販売されている。

もうひとつの方法は、Linuxサーバにシリアルやパラレル接続したプリンタをほかのマシンから使う方法だ。この方法では、プリンタがネットワークインターフェイスを持つ必要がない。さらに、プリントデータはサーバがスプールするので、印刷を実行したマシンはサーバにデータを送るとすぐに解



# 入門・Linux LANの設定

放される。スプールされたデータは、プリンタの速度に合わせてサーバからプリンタへと送られる。印刷速度の遅いプリンタを利用する場合や大量の印刷を行う場合は、かなり快適になるだろう。ただし、初期コストがかからないかわりに、サーバが動いていないとプリンタを利用できないという欠点もある。

本特集では、Linuxサーバにパラレル接続したプリンタをWindowsマシンで共有する方法を解説しよう。

### ■IPマスカレード

フレッツ・ISDNや、ADSLによるインターネット常時接続サービスが始まり、日本の劣悪なインターネット接続環境も少しずつマシになってきた。ISDNで接続する場合、NAT（Network Address Translation）という機能を持ったISDNルータを使うことで、LANで接続されたPCから同時にインターネットへ接続できる。しかし、モデムやターミナルアダプタ（TA）、多くのDSLモデムはNAT機能を持たないため、複数のPCでインターネットを利用することはできない。

Linuxは、NATの機能（IPマスカレード）を持っているため、Linuxサーバにモデムや TAを接続し、LANでつながれたすべてのマシンからインターネットを利用できるようになる。もちろん、回線の自動接続／切断もできるので、通常のダイヤルアップ環境でも大丈夫だ。

本特集では、IPマスカレードの設定と、モデムやTAでプロバイダにつなぐための設定を取り上げる。

### ■基本的なLAN

以上3つのサービスはLANでできることの中でも基本的なものだ。複数のPCがあるなら、これらのサービスが使えるだけでも劇的に便利になる。**図1**はLANを導入した環境のイメージを表している。LANで接続したマシン同士でファイルのやりとりができ、どのPCからもプリンタを利用したり、インターネットへ接続することが可能なのだ。さらに、無線LANを導入すれば、ワイヤレスでこれらのサービスを使うこともできるようになる。

### ■応用

今回は紹介しないが、LinuxマシンにNetatalkというサーバアプリケーションをインストールして、Macintoshとファイル／プリンタを共有することも可能だ。SambaとNetatalkの両方を動作させれば、Linuxサーバを介してWindows、Macintosh両方でファイルを共有することもできる。

また、telnetやSSHを使い、Linuxマシンにリモートログインしてサーバを管理すると便利だ（インターネットに接続する場合は、telnetではなく、パスワードやデータが暗号化されるSSHが必須だ）。

Linuxとネットワークの経験を積めば、複数のプロバイダからメールを自動受信させて一元管理したり、Webメールシステム実現させることも不可能ではない。

Linuxで動作するサーバアプリケーションは数え切れないほどある。きっとおもしろいもの、役に立つものが見つかるに違いない。

## この特集の構成

LANを導入することで、何ができるのか、どのように便利になるのかがわかっていただけただろうか。

本特集の前半部は、基礎知識の学習編という位置づけだ。サーバのソフトウェア設定を行う前に、どのようなハードウェアが必要なのかを知っておこう。また、LANやインターネットで使われる通信プロトコルについて、基本的なところを説明する。プロトコルの基礎知識があれば、トラブルシュートも少しは楽になるだろう。

後半部が実践編で、Linuxサーバを Windowsマシンとつなぐためのソフトウェア Samba、モデムやTAでプロバイダにつなぐためのソフトウェアPPxPのインストールと設定方法や、IPマスカレード機能を使うための設定を解説する。あわせて、Windowsマシンに必要な設定も説明しよう。

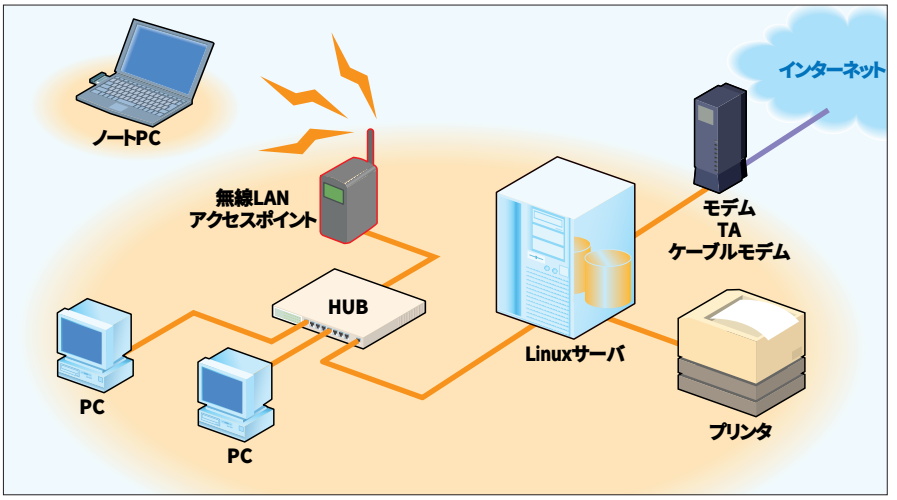



図1　基本的なLANのサービス

# LANの構築に必要な機材

文：編集部／Text：Linux magazine

LANを構築するためには、ハードウェアとソフトウェア両方の準備が必要だ。このセクションではハードウェアについて解説しよう。

LANで使われるネットワークにはいくつかの種類があるが、現在、もっともよく利用されているのがイーサネットだ。イーサネットには、接続方法や伝送速度などによっていくつかの規格があるが、その中でもケーブルの取り回しが楽で、オフィスや家庭で利用するのに向いているのが10BASE-T、100BASE-TXという規格のものだ。これらはハブという機械を介してPCなどの機器をつなぐ方式で、各PCはハブと接続することになる。そのため、ある機器をネットワークから切り離しても、他の機器のネットワーク接続に影響を与えない。

10BASE-Tは伝送速度が10Mbps、100BASE-TXは100Mbpsの規格で、これらを同じネットワーク内で混在させることもできる。bpsとは、ビット／秒（bit per second）の略で伝送速度の単位だ。10BASE-T、あるいは100BASE-TXのイーサネットでLANを組むために必要な機材を順に紹介する。

## イーサネットカード

PCをイーサネットでつなぐためには、個々のPCにイーサネット用のネットワークインターフェイスカード（NIC）が必要だ。パソコンショップには、さまざまな種類のNICが並べられており、どれを購入すればよいのか迷うかもしれない。機器選択のポイントをチェックしてみよう。

### ■PCI接続のカード

PCI接続の拡張バスを備えたデスクトップマシンでは、PCIバスに接続するNICを購入しよう（**写真1**）。ISAバスに接続するNICは設定が面倒なうえに、性能も期待できない。

### ■PCカード

ノートPCの場合は、PCカードスロットに挿すNICを使う。PCカードとは、名刺くらいの大きさの拡張カード規格だ（**写真2**）。かつてはPCMCIAカードと呼ばれることが多かった。PCカードには、データをやりとりするバスが16ビットの規格と、CardBusという32ビットの規格がある。後者はPCがCardBusに対応していないと使えない。また、PCカードには、動作電圧が5Vと3.3Vのものがあり、3.3Vのカードは対応したPCにしか挿せないので注意してほしい。

### ■10BASE-Tと100BASE-TX

数年前まで、100BASE-TXのNICは高価だったが、現在では低価格化が進んでいる。PCI接続のカードは1000円台から、PCカードは3000円台から販売されているので、いまさら10BASE-Tのカードを購入することはないだろう。100BASE-TXに対応するNICは10BASE-Tで動作させることもできるので、迷うことなく100BASE-TX対応カードを購入しよう。

### ■Linuxで動作するか

Linuxサーバで使うNICを購入する場合にはもっとも重要なポイントだ。現在販売されている製品は、ほとんどがWindowsでのみ動作保証をしてお

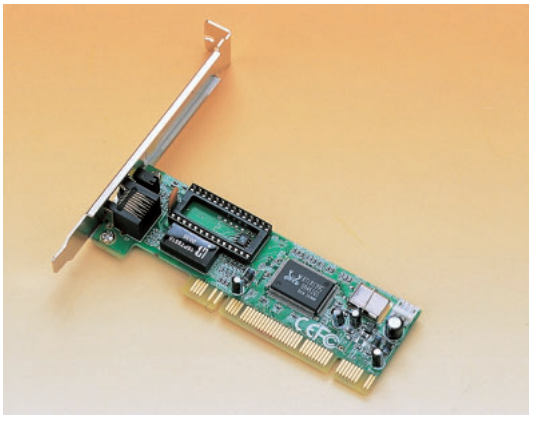


写真1　PCIバスに接続するNIC
1580円で購入したGREEN HOUSE GH-EL 100/LRは、RealtekのRTL8139Cを採用した100BASE-TX対応イーサネットカードだ。


写真2　PCカードのNIC
コレガ FEther CB-TXLは、100BASE-TXに対応する。3.3V駆動のCardBus製品で、低消費電力、32ビットのバス接続がウリだ。



# 入門・Linux LANの設定

り、Linuxでの動作を保証しているものは少ない。とはいえ、これらのカードがLinuxで利用できないわけではない。メルコ、コレガ、プラネックスなどの周辺機器メーカーは、WebページでLinuxでの動作状況を公表しているので購入前にチェックしておくとよいだろう。

Linuxでは、製品ごとではなく、カードに搭載されているイーサネットコントローラチップごとにドライバが用意されている。このため、購入を検討している製品にどのようなチップが使われているのかを把握することが重要となる。製品によってはパッケージにコントローラチップ名が明記されていたり、店頭にあるプライスタグに書かれていることもあるので注意深く見てみよう。

Linuxはさまざまな種類のコントローラチップに対応しているが、ディストリビューションによって用意されているドライバはまちまちだ。利用するディストリビューションが未対応の場合でも、カーネルやドライバのソースコードを入手して、自分でコンパイルすればたいていのNICは利用可能になる。しかし、これは初心者には難しい作業だろう。

市販されているNICの中でも、3Com、IntelのカードとDEC（またはIntel）の21140、21143というチップを搭載したカードは、Linuxユーザーの間では定番となっている。これらのカードはほとんどのディストリビューションでインストール時から認識され、動作時の安定度も高い。ただし、少々値が張るのが難点だ。ノーブランドのものを中心に、低価格のイーサネットカードには、RealtekのRTL8139やVIAのチップを採用したものが多いようだ。RTL8139は、チップにカニのマークが描かれているチップで、Linuxでの動作もそこそこ安定している。一方、VIAのチップを使っているものはLinuxで利用するには設定が面倒なことが多く、動作も安定しないのでお勧めしない。

## ハブ

先に述べたとおり、10BASE-T、100BASE-TXでLANを構築するにはハブが必要となる。2台のPCをつなぐだけなら、クロスケーブルというケーブルを用いてPC同士を接続することも可能だが、場合によっては通信条件をうまく設定できずにデータのやりとりがうまくいかないこともあるのでできるだけハブを使うようにしよう。

ハブにもさまざまなものがあるが、家庭やSOHOで利用する場合、現状では100BASE-TXに対応したスイッチングハブを購入するのがよいだろう。ハブにはポートと呼ばれる接続端子があり、接続する機器の数に合わせてさまざまなポート数のものが市販されている。PCやISDNルータなど、ネットワークに接続したい機器の数に合わせて選ぶことになる。最近では8ポートの製品が主流で、コストパフォーマンスも一番良い（**写真3**、**4**）。

ポート数のほかにもハブを選ぶポイントがある。そのひとつは対応する規格だ。100BASE-TXでネットワークを構築するなら、100BASE-TXに対応するハブが必要になる。また、ハブにはリピータハブとスイッチングハブというものがある。それぞれの違いについて簡単に説明しよう。

### ■リピータハブ

ちょっと細かい話になるが、イーサネットの基本的な通信方式はCSMA/CDというもので、これはデータを送信する機器がほかのすべての機器に対してデータを送りつける方法だ。データを受け取る側では、流れてくるデータの中から、自分宛のものだけを拾い出すことになる。リピータハブは、このルールに則り、あるポートに流れてきたデータを、ほかのすべてのポートに中継する。この方法では、1つの伝送路をすべての機器で共有するために、

写真3　スイッチングハブ
編集部で使用しているプラネックス FX-08SMCは、筐体の背面に8つのイーサネットポートを備える。製品によっては、ポートが前面のものもある。状況に合わせて選択しよう。

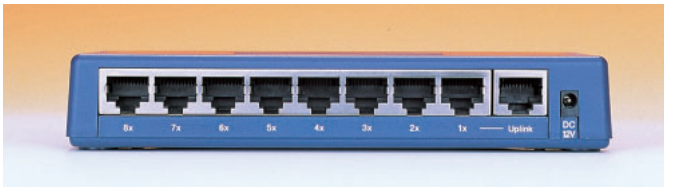


写真4　FX-08SMCの背面
左から8つのイーサネットポートが並んでいる。9つめはカスケードポートといい、ハブを多段接続する場合に利用する。このポートは8番目のポートと排他利用しなければならない。

複数の機器が同時にデータを送信することはできない。そのため、各機器はデータを送信する前にほかの機器がデータを送信していないかどうかを調べて、伝送路が空いていればデータを送信する。

たとえるなら、しきりの悪い会議のようなものといえる。発言する前に聞き耳をたてて、ほかの人がしゃべっていないことを確認してから参加者全員に向けて発言する。運悪く発言が重なることもあるので、その場合は発言がダブった両者がいったん黙り、次の機会をうかがう。全員がこんなことをすると、参加者が多くなるほど発言はぶつかりやすくなる。これと同じように、リピータハブを使ったネットワークでは通信量が増えるほど衝突（コリジョンという）が増えるので、その結果、転送効率が低下する。

また、リピータハブを用いると10BASE-Tと100BASE-TXを混在させることはできない。導入コストの低さが利点ではあるが、次に説明するスイッチングハブの価格が下がった現在ではあえて選ぶ必要はないだろう。

### ■スイッチングハブ

リピータハブがデータを全ポートに流すのに対し、スイッチングハブはデータの宛先を見て必要なポートにだけデータを送る。これによって2台の機器が直接接続されたのと同じような形となりネットワーク全体のスループットが向上する。先ほどと同じように会議にたとえれば、話をしたい相手と発言者の間を電話でつないだようなものだ。このようにすることで、複数の会話が同時に行えることになる。

また、多くのスイッチングハブには10BASE-T、100BASE-TX変換機能があるので、このようなハブを用意すれば両者の混在が可能になる。現在では8ポートの製品が数千円から手に入るので、スイッチングハブの購入をお勧めしたい。

## イーサネットケーブル

10BASE-T、100BASE-TXのイーサネットでは、電話のモジュラケーブルに似たケーブルで機器を接続する（**写真5、6**）。結線の仕方によってストレートケーブルとクロスケーブルという2種類のものがあるが、PCなどの機器とハブの間はストレートケーブルで接続するので、間違えてクロスケーブルを購入しないようにしよう。クロスケーブルは、ハブ同士を接続する場合などに利用するもので、ハブが1台のネットワークでは必要ない。

また、通信速度によってケーブルのグレードが規定されており、10BASE-Tではカテゴリ3以上、100BASE-TX

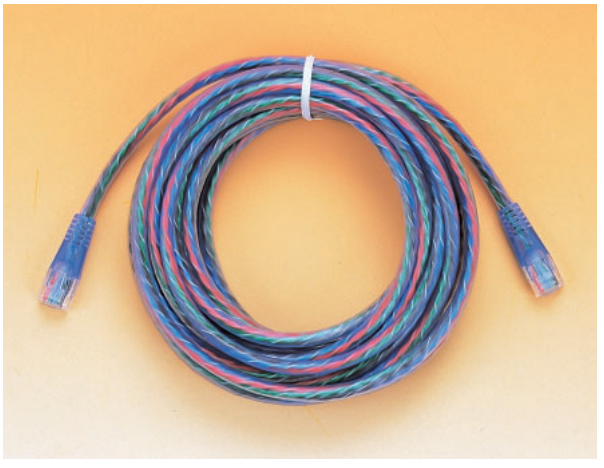


写真5　イーサネットケーブル
5mのカテゴリ5ストレートケーブル。ケーブルにはさまざまな色のものがあるので、接続機器ごとに色分けしておくと配線ミスを起こしにくくなるだろう。

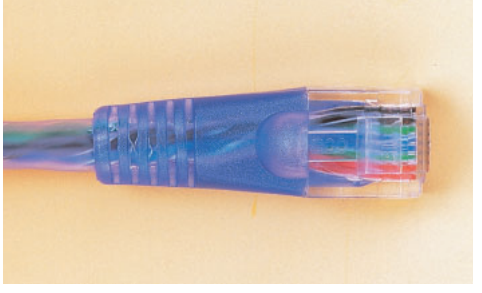


写真6　イーサネットケーブルのコネクタ
10BASE-T／100BASE-TXでは、通称RJ-45と呼ばれるコネクタを利用する。信号線は、ノイズ耐性を向上させるために、1対ずつ撚り合わせてある。

## Column

### Linuxと無線LAN

最近ではIEEE 802.11bという規格に対応した伝送速度が11Mbpsの無線LAN機器が各社より発売され、人気を呼んでいる。これはノートPCなどのPCカードスロットに写真のようなNICを挿し、2.4GHz帯の電波を使って通信するものだ。ケーブルをつなぐことなくネットワークを利用できる環境は思いのほか便利だ。

現在のところLinuxへの対応を正式にサポートしているメーカーはないようだが、LinuxのPCカードサービス（pcmcia-cs）ドライバを使うことで多くのカードが利用可能だ（http://www.pcmcia-cs.sourceforge.net/）。pcmcia-csのインストールや設定の方法については、『Linux PCMCIA HOWTO』が参考になる。Linux JF（Japanese FAQ）Projectによる『Linux PCMCIA HOWTO』の日本語訳が用意されているので、英語が苦手な方はこちらが役に立つだろう（http://www.linux.or.jp/JF/JFdocs/PCMCIA-HOWTO.html）。また、本誌2000年8月号で無線LANの特集をしているので、お持ちの方は参考にしてほしい（メルコ、コレガ、プラネックス、アイコムの製品を検証し、すべてが動作した）。



# 入門・Linux LANの設定

ではカテゴリ5が必要だ。とはいえ、国内で販売されているイーサネットケーブルはほぼすべてがカテゴリ5なので特に気をつけなくてもよいだろう。店頭にはさまざまな長さ、色のものが置かれているので、必要な長さで好みの色のものを購入すればよい。

## インターネットへの接続

Linuxサーバを介してインターネットに接続する場合、通信機器や接続サービスの種類によって接続方法が変わってくる。これらを大別すると**図2**に示した3種類になるだろう。

### ■モデム／TAで接続する

モデムやターミナルアダプタ（TA）をLinuxサーバのシリアルポートに接続し、LinuxのIPマスカレード機能を用いることで、LAN内の他のマシンからもインターネットへ接続することができる。Linuxサーバは、PPP（Point to Point Protocol）でプロバイダに接続する。本特集では、設定がわかりやすく、機能も豊富なPPxPというPPP接続ソフトウェアを紹介する。

### ■ISDNルータなどで接続する（1）

IPマスカレード機能を持たないケーブルモデムやDSLモデムを使ってインターネットへ接続する場合は、Linuxサーバに2枚のNICを挿し、1枚をインターネット用に、もう1枚をLAN用に設定する。この状態でLinuxのIPマスカレード機能を使えば、LAN内のPCからインターネットへ接続できるようになる。ISDNルータのIPマスカレード機能を使わずに、LinuxでIPマスカレードを行う場合もこのような接続方法になる。

この接続方法では、IPマスカレードのほか、パケットフィルタリングによる接続制御もLinuxで行うことになる。そのぶん設定は面倒になるが、Linuxサーバを簡易ファイアウォールにするなど、細かい制御をできるのがメリットだ。

### ■ISDNルータなどで接続する（2）

ISDNルータやDSLモデムのIPマスカレード機能を用いてインターネットへ接続する場合は、この接続方式も利用可能だ。この場合、Linuxサーバはほかの PCと同じように接続されることになり、インターネットからは直接見えない。そのため、インターネットから接続されるサービスを行うのは難しい。また、LANの中からインターネットへの接続にLinuxサーバは関与しないので、Linuxサーバをファイルサーバ、プリンタサーバとしてのみ使う場合にお勧めする方式だといえる。

このほか、複数のPCを同時に接続できる契約を行っているときもこのように接続できる。ただし、Linuxサーバを含め、すべてのPCが外部ネットワークに直接さらされることになるので、個々のマシンでのセキュリティ対策は不可欠だ。できることなら、LinuxサーバにNICを2枚挿して、ファイアウォールとして設置したい。

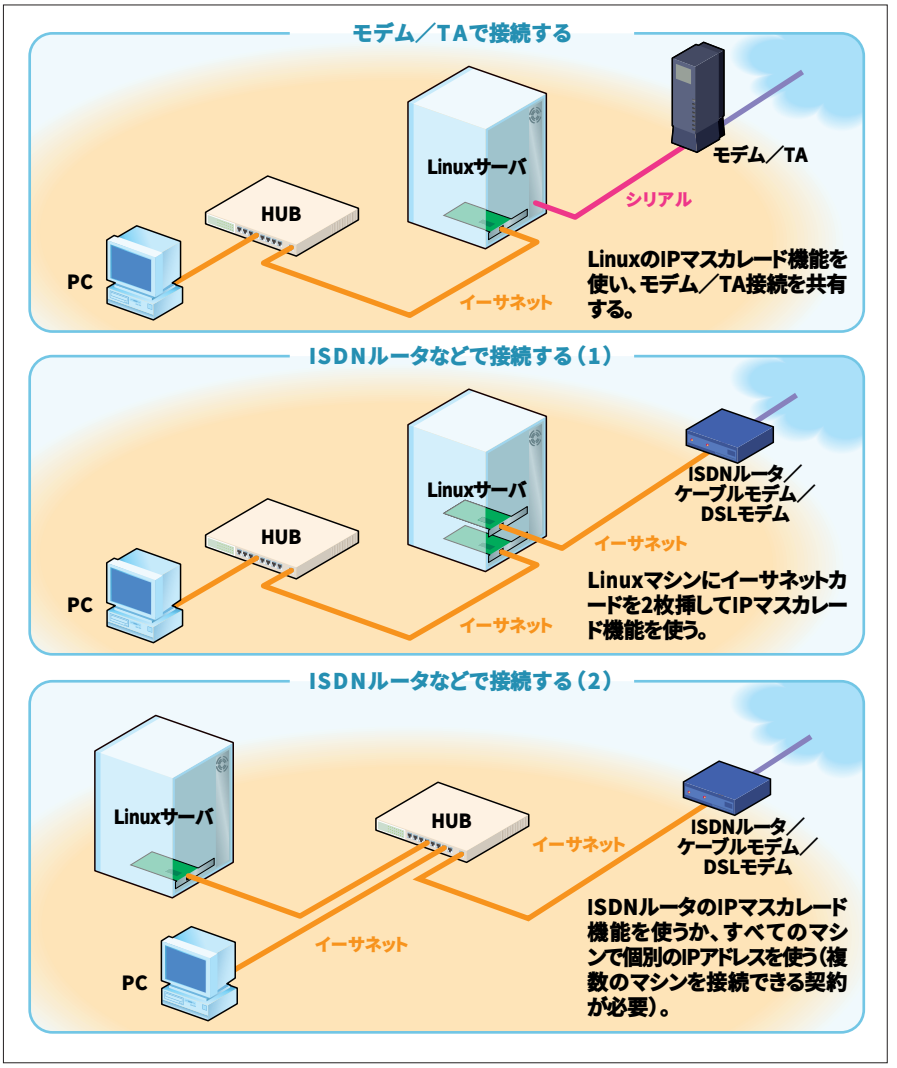



図2　Linuxサーバをインターネットにつなぐ方法

# TCP/IP初歩の初歩

文：編集部／Text：Linux magazine

LANでコンピュータ同士がデータをやりとりする仕組みをおおまかに知っておこう。もちろん、そんなことを知らなくてもLANを利用することは可能だが、Linuxサーバを立ち上げるなら知っておいて損はない。

## プロトコルってなんだ？

LANやインターネットを利用する場合、「プロトコル」という言葉をよく見かけるはずだ。通信の世界でいうプロトコルとは、データの送受信を行う手順を定めた約束事のことを指す。実生活の場面にたとえてみよう。電話をかけるときには、最初に「もしもし」「はい、吉田です」などという会話を交わすことが多いだろう。こうして会話が始まり、用件が済むと「さようなら」といって電話を切る。これは電話をかける際のプロトコルのようなものだ。もちろん、人間はコンピュータに比べると柔軟性が高いので、このような手順通りに話さなくても電話はかけられる。逆にいえば、コンピュータを始めとする機械同士が通信するためには、厳密なプロトコルを決めておかないとうまくやりとりができないのだ。

プロトコルには実にさまざまな種類のものがある。HTTP、FTP、TCP/IP、NetBEUI、PPPなど、一度は目にしたり聞いたりしたことがあるだろう。プロトコルは、役割ごとに作られており、その数は膨大なものになっている。

また、ソフトウェアメーカーなどは、各社独自の通信プロトコルを作り、自社製品に採用してきた。たとえば、DOSのLAN-ManagerやWindowsではNetBEUI、NetWareではIPX/SPX、MacintoshではAppleTalk、多くのUNIXではTCP/IPというプロトコルが標準的に使われていた。これらには互換性がなく、相互に通信することはできない。いろいろな種類のコンピュータを導入すると、それらをネットワークでつなぐのはたやすいことではなかったのだ。

結局、マルチベンダープロトコルの事実上の標準は、仕様が公開されており、実装を重視したTCP/IPとなって今日に至っている。TCP/IPは、インターネットの原型であるARPANETのために開発されたプロトコルだ。ここでは、TCP/IPの基礎について解説することにする。

## TCP/IPで使われるプロトコルの種類

TCP/IPとは、Transmission Control Protocol/Internet Protocolを省略したものだが、一般には、データの送受信手順から、各種のサービスを行うための通信手順まで、数多くのプロトコル群をまとめて総称したものだ。たとえば、データの送受信には、TCP、UDP（User Datagram Protocol）、IPといったプロトコルが使われている。また、サービスについては、ファイル転送サービスのFTP（File Transfer Protocol）、WebサービスのHTTP（Hyper Text Transfer Protocol）などが用意されている。

TCP/IPは、アプリケーション層、トランスポート層、ネットワーク層、データリンク層という4つの層から成り立っており、各層が連携して働く（**図3**）。ユーザーに一番近い層がアプリケーション層で、通常のネットワーク利用では主にこの層を担当するアプリケーションソフトウェアを利用することになる。トランスポート層やネットワーク層にユーザーが直接関わることはあまりないが、管理者がネットワーク障害の原因を究明するときなどにはこれらの層に関係するソフトウェアを使

| TCP/IP層 | プロトコル | | 通信相手を認識する方法の例 |
|---|---|---|---|
| アプリケーション層 | HTTP、FTP、telnet、SMTPなど | DNS、NFSなど | URLやメールアドレス |
| トランスポート層 | TCP | UDP | ポート番号 |
| ネットワーク層（インターネット層） | IP、(ICMP) | | IPアドレス |
| データリンク層（ネットワークインターフェイス層） | イーサネット | PPP、SLIP / PLIP | MACアドレス |
| 物理層 | 10BASE-T、100BASE-TX | シリアル / パラレル | |

図3　TCP/IPの階層



用することもある。また、データリンク層に関しては、ダイヤルアップソフトウェアくらいしかユーザーが操作するものはない。**図3**にある物理層は、TCP/IP層ではないのだが、イーサネットやシリアルケーブル、あるいはパラレルケーブルなど、いろいろな種類の通信媒体でTCP/IPが利用できることを示している。

## 階層化することのメリット

TCP/IPの各階層には役割が決められており、それぞれの層は隣の層と連携して動作する。つまり、分業化されているわけだ。したがって、ネットワーク関連のソフトウェアは、担当する層の処理だけを行い、隣接する層の仕事はその層を担当するソフトウェアに任せるように作られている。もちろん、隣接する層のプログラムとデータを受け渡しする方法もあらかじめ決められている。

たとえば、アプリケーション層を担当するWebブラウザは、HTTPの処理と、トランスポート層のソフトウェアとのやりとりだけを行う（もちろん、テキストや画像の表示といった処理も必要だが、ここではネットワークに関する処理に話を絞る）。もし、分業ができていないなら、Webブラウザ自身が通信経路を決めたり、イーサネットインターフェイスやシリアルポートの制御までやらなければならない。Webブラウザだけではない、FTPクライアントやtelnetクライアントなど、ネットワークに関わるアプリケーションソフトすべてが同様の実装をしなければならなくなってしまう。

では、各階層の役割を順番にに見ていこう。

### ■アプリケーション層

一般のユーザーが利用するネットワークアプリケーションの層だ。ここまでに出てきた、HTTP（Web）、FTP（ファイル転送）、telnet（遠隔ログイン）のほか、メール配信を行うSMTP（Simple Mail Transfer Protocol）やPOP3（Post Office Protocol version 3）などのプロトコルがこのアプリケーション層に分類される。

アプリケーション層のプロトコルは、単純なテキストによるコマンドと応答で規定されているものも多い。たとえば、メールサーバからメールをダウンロードするのに使われるPOP3というプロトコルは、**表1**のようなコマンドを使う。PCなどのクライアントがコマンドを送信し、それに応じてサーバが応答

| コマンド | 意味 |
|---|---|
| USER *mailbox* | メールボックス名の送信 |
| PASS *password* | パスワードの送信 |
| LIST ［*message No.*］ | 特定／全体の情報問い合わせ |
| RETR ［*message No.*］ | メッセージのダウンロード要求 |
| DELE ［*message No.*］ | メールの削除要求 |
| QUIT | 完了通知 |

表1　POP3の主なコマンド

▼リスト1　POP3による通信の例

```
+OK QPOP (version 2.52) at mail.asciilinux.com starting.
USER q-chan
+OK Password required for q-chan.
PASS ********
+OK q-chan has 1 message (675 octets).
LIST
+OK 1 messages (675 octets)
1 675
.
RETR 1
+OK 675 octets
Received: from lmpc02.asciilinux.com by mail.asciilinux.com
(8.9.3/3.7W)
        id WAA19805 for <q-chan@asciilinux.com>; Sat, 24 Feb 2001
22:17:38 +0900 (JST)
Message-Id: <200102241317.AA00157@lmpc02.asciilinux.com>
From: P-suke <p-suke@asciilinux.com>
Date: Sat, 24 Feb 2001 22:17:39 +0900
To: q-chan@asciilinux.com
Subject: Sample mail
MIME-Version: 1.0
Content-Type: text/plain; charset=iso-2022-jp
X-UIDL: 572a58e7594aedbe1bdb402eaefe13b4

Linux magazineは毎月8日発売です。
            :
            :

.
DELE 1
+OK Message 1 has been deleted.
QUIT
+OK Pop server at mail.asciilinux.com signing off.
```

クライアントが接続するとサーバは挨拶メッセージを返す
メールボックス名を送信
パスワードを送信
届いているメールの情報の問い合わせ
サーバから、「メッセージ番号 サイズ」というフォーマットでメールの情報が送られる
メッセージ番号1のメールをダウンロード要求
サーバからメッセージ番号1のメールが送られる
メッセージ番号1のメールを削除要求
終了

する。サーバからの応答も「＋OK」、「－ERR」といった簡単なテキストとなっている。**リスト1**は、POP3でメールをダウンロードしたときのやりとりだ。

### ■トランスポート層

トランスポート層には、通信する相互のマシンで動作しているアプリケーション同士をつなぐ通信プロトコルが属する。代表的なプロトコルはTCP（Transmission Control Protocol）とUDP（User Datagram Protocol）だ。アプリケーション層のソフトウェアから渡されたデータは、小さなデータに分割され、宛先と送信元のポート番号などが書かれたヘッダが付加される。ポート番号とは、アプリケーションを特定するための番号で、主なアプリケーションには、個別の番号が予約されている。たとえば、HTTPは80番、telnetは23番が割り当てられている。それ以外の割り当ては、/etc/servicesに書かれているので参照してほしい。

逆に、ネットワーク層から渡された細切れのデータは再結合され、宛先のポート番号に応じたアプリケーションに渡される。

UDPは、データグラム型、あるいはコネクションレス型と呼ばれるプロトコルで、信頼性確保の処理を行わずに分割したデータ（UDPデータグラム）を伝送する。相手にデータが届いたことの確認や再送処理などは行わないため、信頼性は低いがそのぶん高速な処理が可能だ。映像や音声など、リアルタイム性の高いデータを送る場合によく使われる。

一方、TCPは信頼性のあるコネクション型通信を行うプロトコルで、通信に先立って通信路の確立を行う。TCPでは分割したデータ（TCPセグメント）の送達確認や、エラー時の再送を行うほか、データが順序よく相手に到達するように制御する。このため、TCPを使うアプリケーションは、データ転送に関してはほとんどエラーフリーとなる。

### ■ネットワーク層

ネットワーク層に属するIP（Internet Protocol）は、コネクションレス型の通信プロトコルで、宛先や経路の指定に使うアドレス処理と、フラグメント処理を行う。

トランスポート層から渡されたデータには、宛先と送信元のIPアドレスなどが書かれたヘッダが付加される。IPアドレスとは、最終目的地にデータを伝送するための経路選択に使用されるアドレスで、現在広く使われているIPv4（IPバージョン4）では32ビットの数値が使われている。表記する際には、192.168.8.30のように、8ビットずつ区切って10進数にしたものを「.」ドットでつないだ形にすることが多い。ルータなどのネットワーク機器は、宛先のIPアドレスを元にデータグラムの送り先を決定する。これが繰り返されることによって最終的には宛先IPアドレスを持つマシンにデータグラムが到着する。

また、伝送パケット長が制限されているネットワークでデータを送る際には、データグラムの分割／再組み立て（フラグメント／リアセンブリ）をする。これがフラグメント処理だ。

IPは、単なるデータ転送の機能しか持たないため、これを補完するプロトコルとしてICMP（Internet Control Message Protocol）が用意されている。ICMPは、経路決定や伝送についてのエラーや情報の通知を行う。

なお、IPアドレスを使うには、重複することがないように割り当てを受ける必要があるが、インターネットなど外部のネットワークに接続しないネットワークでは、プライベートアドレスという特別なIPアドレスを使うことができる（**表2**）。本特集では、構築するLANで192.168.0.0～192.168.0.255というプライベートアドレスを使うことにしよう。IPマスカレードを利用してインターネットに接続する場合には、Linuxサーバがプロバイダから割り当てられたIPアドレスとプライベートアドレスを相互に変換するので心配しないでほしい。

### ■データリンク層

イーサネットにも独自のプロトコルがある。IPではIPアドレスを使いIPデータグラムを伝送するが、イーサネットでは、カードごとに割り振られているMAC（Media Access Control）アドレスを使ってイーサネットフレームを伝送する。データリンク層では、IPアドレスとMACアドレスを相互に変換したり、IPデータグラムとイーサネットフレームの対応を取ることにより、イーサネットでIPデータグラムを運ぶことを可能にしている。ダイヤルアップでプロバイダに接続する場合には、PPP（Point to Point Protocol）というプロトコルを使ってIPデータグラムを運べるようにする。

## TCP/IP通信の流れ

Webを例にTCP/IPでの通信を説明しよう。ブラウザから送られるHTTPのコマンドやデータは、TCPセグメントに分割される。それぞれのTCPセグメントには、宛先ポート80という情報を含むTCPヘッダが付加される。さらに、WebサーバのIPアドレスを含んだIPヘッダが付加されたIPデータグラムとなり、イーサネットフレームに納め



られてイーサネットケーブルを伝わる。受信したサーバでは、逆の手順で処理を進め、アプリケーション層のWebサーバプログラムにコマンドやデータを渡すのである。

このように、TCP/IPでの通信においては、IPアドレスで通信相手を指定し、ポート番号で利用するサービスの種類を指定する。しかし、Webをブラウズするのに IPアドレスやポート番号を意識することはほとんどない。ご存じのとおり、Webサービスでは「http://www.ascii.co.jp/」のようにURL（Uniform Resource Locators）を用いる。この例では、HTTPを使い、www.ascii.co.jpという名前のサーバに接続するという意味になる。HTTPでは、デフォルトで80番のポートを使用するので指定しなくてよい。しかし、サーバの名前がわかっていても、IPアドレスがわからなければ接続できない。TCP/IPには、DNS（Domain Name System）というサービスがあり、ホストの名前とIPアドレスを相互に変換できる（**図4**）。最寄りのDNSサーバにwww.ascii.co.jpのIPアドレスを問い合わせると、DNSサーバはほかのDNSサーバに問い合わせるなどしてIPアドレスを調べ、クライアントに回答する。

1台のサーバで複数のサービスを提供することも可能だ。**図4**のserver.asciilinux.comでFTPサーバとWebサーバを動かしているとしよう。ポート番号21への接続はFTPサービス、ポート80番ならWebサービスを提供することになるわけだ。

DNSは、メールサーバを調べることもできる。たとえば、「q-chan@asciilinux.com」というメールアドレスにメールを送りたいとしよう。ユーザーは宛先にこのアドレスを指定してメールを送信する。メールの配送を頼まれたサーバは、DNSサーバに対して「asciilinux.com」というドメインを担当するメールサーバをたずね、教えられたサーバの25番ポートに接続してSMTPでメールを渡すことになる。

## WindowsとLinuxをつなぐSamba

前にも述べたように、LAN-Managerや初期のWindowsでは、TCP/IPではなくNetBEUIというプロトコルが使われてきた。NetBEUIには、動作が軽い、設定が簡単という利点がある反面、ルーティングができないなどの欠点がある。そこで、NetBEUI上のアプリケーションインターフェイスであるNetBIOSをTCP/IP上で使えるようにしたのがNetBIOS over TCP/IPだ。TCP/IPのルーティング機能などを利用し、Windowsネットワークを利用できるのである。

このNetBIOS over TCP/IPをUNIX系OS上に実装したのがSambaというソフトウェアで、これをLinuxサーバで動かすことにより、Windowsからファイルを読み書きしたり、プリンタを共有することができる。Windows 95／98／Me、Windows NT／2000は、NetBEUIとNetBIOS over TCP/IPのどちらも利用可能だが、SambaはNetBIOS over TCP/IPにしか対応していないので、Sambaを利用する際には、WindowsマシンでNetBIOS over TCP/IPを使えるようにしておく必要がある。

TCP/IPには、ここで紹介した以外にもたくさんのプロトコルがある。また、Linuxで動作するネットワークソフトウェアも数え切れないほど用意されている。興味を持った方は、書籍をひもといたり、Webで検索するなどして勉強してみるのもよいだろう。

では、いよいよLinuxサーバの設定を始めよう！

| IPアドレス |
|---|
| 10.0.0.0 ～10.255.255.255 |
| 172.16.0.0 ～172.31.255.255 |
| 192.168.0.0 ～192.168.255.255 |

表2　プライベートアドレス

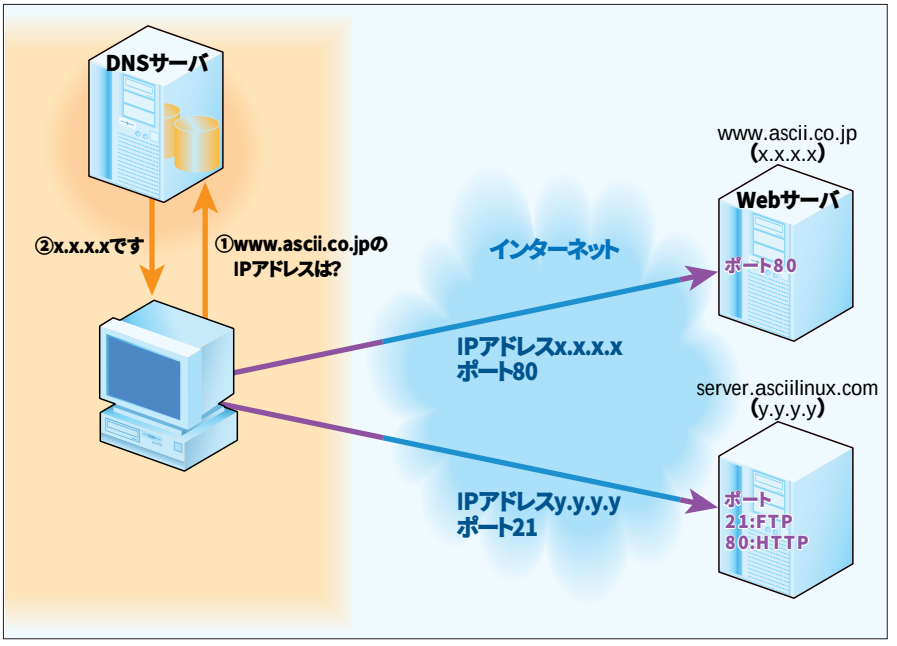



図4　インターネットサーバへの接続

# ネットワークへの接続と設定

文：竹内 充彦／Text：Michihiko Takeuchi

## モデム／TAの装着と接続確認

xDSLやCATVが徐々に普及してきているとはいえ、まだまだ、アナログ電話回線やISDNによるダイヤルアップ接続を利用しているユーザーのほうが圧倒的に多いはずだ。そこで、まずは、Linuxマシンをモデムや TAでダイヤルアップ接続するための設定を解説しよう。

### ■ モデム／TAの取り付け

モデム／TAをLinuxマシンに接続する。シリアルケーブルで接続する場合は、PCの背面にあるシリアルポートに接続する。内蔵モデムの場合は、PCの電源を切り、筐体を開け、マザーボード上のスロットにモデムカードを挿す。

LinuxはUNIXを手本にしているため、すべてのデバイスに対して、ファイルと同様にアクセスできるように設計されている。これをデバイスファイルといい、/devというディレクトリに置かれる。シリアルポートも例外ではない。たとえば、シリアルポートは、/dev/ttyS＊というデバイスファイルが対応する。Windowsでいうところの COM1は/dev/ttyS0、COM2は/dev/ttyS1にあたる。末尾の数字が1つずつずれるので注意しよう。内蔵モデムはCOM3やCOM4として認識されることもあるが、その場合は/dev/ttyS2や/dev/ttyS3が対応する。

### ■ 接続の確認

接続したモデムが、デバイスファイルを介して正しく通信できるかどうか確認してみよう。ktermやxtermを起動して、コマンドラインからcuコマンドを使う。

ところで、Linuxのファイルにはすべてアクセス権が設定される。デバイスファイル/dev/ttyS＊は、rootとユーザーグループuucpに属するユーザーにのみ読み書きが許可されている。それ以外のユーザーにはアクセスできない。ここでは、rootで行おう。

cuコマンドの引数－lの後には、ずばりデバイスファイル名を指定する。－sの後には、通信速度をbps単位で指定する。

```
# cu -l ttyS0 -s 38400
```

モデム／TAの接続されたシリアルポートに正しく接続されれば、モデム／TAと対話ができる。atと入力すればOKと返事が返ってくるはずだ（**画面1**）。接続されていない場合は、atと入力しても何も応答がないし、入力そのものも画面に表示されない。どちらの場合も終了するには「~ .」を入力する。モデム／TAが接続されていないポートの場合、cuコマンドが終了するまでに時間がかかることもある。

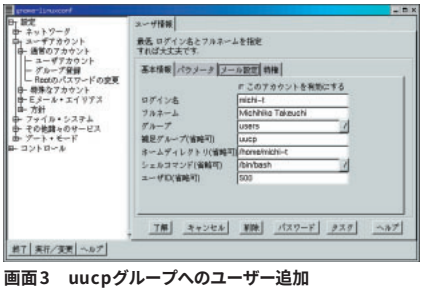
モデムの接続されたポートを見つけられただろうか？ 見つけたら、そのポ

```
# cu -l ttyS0 -s 38400          ← 試しにttyS0に接続してみる
Connected.
at                              ← モデム／TAにatコマンドを送出するためatと入力する
OK                              ← 返事が返ってくればモデム／TAが接続されている
~.                              ← cuを終了するには「~.」を入力する
Disconnected.
#
```

**画面1　モデム／TAの接続テスト**
正しく接続されているかどうか確認しておこう。

```
# rm /dev/modem                 ← modemをいったん削除する
# ln -s /dev/ttyS0 /dev/modem   ← ttyS0をmodemにシンボリックリンクする
# ls -l /dev/modem              ← リンクを確認する
lrwxrwxrwx    1 root    root      10 Feb 16 03:24 /dev/modem -> /dev/ttyS0
#
```

**画面2　デバイスファイルの設定**
モデムを接続したポートから/dev/modemにシンボリックリンクを張っておく。



**画面3　uucpグループへのユーザー追加**
linuxconfで、モデムを利用するユーザーをuucpグループにも所属させる。



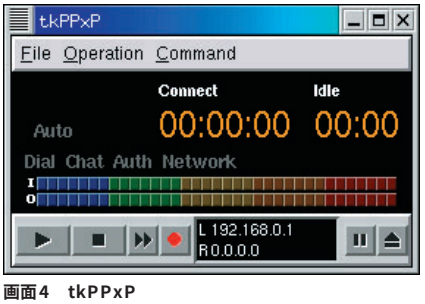
ートを/dev/modemにシンボリックリンクしておこう。すでに/dev/modemというリンクファイルが存在する場合は、いったんそれを削除してからリンクし直す（**画面2**）。

ついでといっては何だが、ここでモデムを利用するユーザーをuucpグループにも所属させておこう。linuxconfのユーザーアカウントで、補足グループにuucpを追加する（**画面3**）。

## PPxPによるダイヤルアップ接続

モデム／TAを利用して、LinuxからISP（インターネットサービスプロバイダ）にダイヤルアップ接続するには、Windowsの場合と同様、PPP（Point to Point Protocol）というプロトコルを利用する。Linuxで、PPPを利用したダイヤルアップ接続を行うソフトウェアにはいくつかあるが、日本で現在主流なのは真鍋敬士氏の作成したPPxPである。

PPxPには、（1）ユーザーが必要に応じて起動可能、（2）LANからの接続要求に応え自動的に接続可能、（3）パケットフィルタリングが可能、（4）簡易IPマスカレード機能を持つ、といった特徴がある。またユーザーインターフェイスが独立しているため、CUIにもGUIにも適切なインターフェイスが用意されている。ここではインターフェイスに、GUIのtkPPxPを使う（**画面4**）。




画面4　tkPPxP
PPxPのインターフェイスの1つであるtkPPxP。X Window System上で動作する。

Linuxのネットワーク機能は、前節でも紹介したTCP/IPをベースに実装されている。しかし、モデム／TAを介したプロバイダとの接続にはPPPが使われる。したがって、ダイヤルアップ接続を利用するには、TCP/IPのパケットをいったんカプセル化してPPPに乗せてデータ交換した後、カプセルからTCP/IPのパケットを取り出すという仕組みが必要になる。こうすることで、見かけ上、TCP/IPがPPP接続間をくぐり抜ける。このような仕組みをトンネリングという。

トンネリングを利用するには、Linuxにトンネルデバイスを用意しなければならない。BSD系のOSでは、カーネルに標準的に実装されているのだが、Linuxではカーネル2.4からの実装となっており、現在主流の2.2.＊では、ドライバモジュールとして組み込む必要がある。

PPxPで利用可能なトンネルデバイスには、以下のものがある。userlinkはカーネル2.1～2.2.＊で、ethertapは2.2.＊で利用可能だ。

```
ul0     userlinkで提供
tap0    ethertapで提供
tun0    カーネル2.4に実装
```

なお、カーネル2.4以降で実装されるtun0については、PPxPの開発中のバージョンでしか対応していないため、解説は省略する。ここでは、カーネル2.2.＊でtap0（ethertap）を使うことにする。

### ■ PPxPとtkPPxPのインストール

まずは、PPxPとtkPPxPがインストールされているかどうかを確認しよう。実行ファイルがどこに存在するかを調べるには、whichコマンドを使う。whichは、コマンド検索パスに従って、シェルから“どの”ファイルが実行されるかを調べるためのコマンドだ。コマンド実行後、以下のようにパスが表示されれば実行可能な状態にあるので、次のトンネルドライバの組み込みに飛ぼう。

```
# which ppxp
/usr/bin/ppxp
# which tkppxp
/usr/X11R6/bin/tkppxp
```

コマンド検索パスに見つからない場合は、which実行後以下のようなメッセージが表示される。

```
# which ppxp
bash: ppxp: command not found
```

このように表示された場合は、インストールされていないか、コマンド検索パス上に存在しない。あとからインストールした場合は、/usr/local/binにあるかもしれないので注意。

もし見つからなければ、新たにインストールしよう。PPxPは、インターネットから……と言いたいところだが、まだ接続できない。今月号の付属CD-

```
# lsmod
Module                  Size  Used by
ethertap                2388   1  (autoclean)
#
```

画面5　lsmodの実行結果
ethertapが組み込まれていることが分かる。

ROMにtarボールが収録されているので、そこからインストールしよう。

CD-ROMをドライブにセットして、以下の手順でインストールする。

```
# mount /mnt/cdrom
# cd /mnt/cdrom/Linuxmag/Special/PPxP
# cp ppxp-0.99120923.tar.gz /tmp
# cd /tmp
# tar xvzf ppxp-0.99120923.tar.gz
# ./configure
# make
# make install
```

### ■ トンネルデバイスドライバの組み込み

次に、トンネルデバイスのドライバを確認しよう。最近のディストリビューションなら、最初からethertapが用意されており、PPxPを起動すると、ethertapが組み込まれる。

試しに、tkPPxPを起動し、lsmodコマンドで組み込まれているモジュールを確認してみよう。**画面5**のようにethertapが組み込まれていれば何もすることはない。

組み込まれていない場合は、まず、モジュールethertap.oが存在するかどうか確認してみよう。ethertap.oは、ディレクトリ/lib/modules/2.2.*/netの下にある（*はバージョンにより異なる）ので、手作業で組み込んでみよう。

```
# depmod -a
# modprobe ethertap
```

組み込まれたかどうかlsmodで確認しよう。うまくいったら、起動時にethertapが読み込まれるように、ファイル/etc/conf.modules（/etc/modules.confの場合もある）に以下の1行を追加するといい。

```
alias tap ethertap.o
```

また、デバイスファイル/dev/tap0が存在するかもあわせて確認しておこう。なければ作成し、グループをuucpに、アクセス権を660に変更する。

```
# mknod /dev/tap0 c 36 16
# chgrp uucp /dev/tap0
# chmod 660 /dev/tap0
```

以上で、トンネルデバイスの準備ができた。

### ■ ダイヤルアップ接続の設定

PPxPを利用するためには、あらかじめプロバイダの情報や電話番号など、接続に必要な設定を行う必要がある。これはtkPPxPから行うことができる。

コマンドラインから、tkPPxPを起動しよう。ppxpは一般ユーザーが起動することもできる。通常はそうするべきだ。とくに、tkPPxPをrootで起動するのは避けるべきだ。インターネットに接続されるマシンにおいて、root権限で、しかもX Window Systemを起動するというのは、セキュリティ上問題だ。常時接続ならなおさらだ。

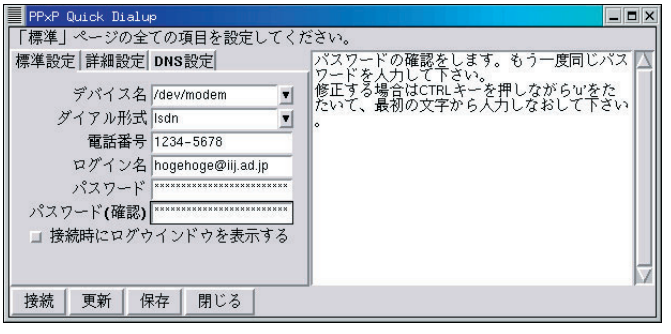
```
$ tkppxp
```

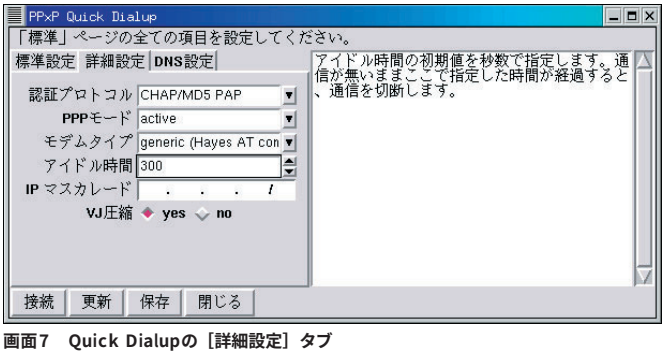
tkPPxPの［Operation］メニューから［Quick Dialup］を選択する。すると、［PPxP Quick Dialup］ダイアログボックスが表示される。

まず、［標準設定］タブでの各項目を設定する（**画面6**）。

「デバイス名」には、モデムやTA

画面6　Quick Dialupの［標準設定］タブ




画面7　Quick Dialupの［詳細設定］タブ


画面8　Quick Dialupの［DNS設定］タブ

の接続されたデバイス名を指定する。ここでは、先ほど設定した/dev/modemをリストから選択しておけばいい。「ダイヤル形式」は、パルスやトーンなど、利用している回線に応じてリストから選択する。「電話番号」は、プロバイダのダイヤルアップ接続先の電話番号だ。「ログイン名」は、プロバイダから割り振られたダイヤルアップ接続用のログイン名を入力する。「パスワード」と「パスワード（確認）」は、先の「ログイン名」に対応するパスワードを入力する。2つの欄には、同じものを入力しなければいけない。

次に［詳細設定］タブの各項目を設定する（**画面7**）。

「認証プロトコル」は、接続先のプロバイダがどのユーザー認証プロトコルを使っているかによる。CHAPはパスワードを暗号化して送出するが、PAPはそのまま送出する。「PPPモード」はactiveでいい。［モデムタイプ］は、もしリストに使用しているモデムと同じ型番があればそれを選べばいいし、なければgenericを選んでおけばいいだろう。「アイドル時間」は、データのやりとりがなくなってから回線を自動切断するまでの時間を秒で指定する。0にしておくと自動切断しない。「IPマスカレード」は、ここでは設定しない。IPマスカレードについては後述する。［VJ圧縮］は、プロバイダにもよるが、通常はyesでいいだろう。

さらに［DNS設定］タブの各項目を設定する（**画面8**）。

［接続時のresolv.confの書き換え］は、PPP接続時にプロバイダからDNSサーバのアドレスが取得できる場合、それを/etc/resolv.confの内容と書き換えるかどうかだ。これも接続先のプロバイダ次第なのだが、通常はyesにしておき、下の［DNSサーバの指定］を記述しておくのがいいだろう。サーバのアドレスはプロバイダによって異なるので調べておこう。

設定したら、［保存］ボタンをクリックして、ファイル名を指定して保存する。このとき、rootの場合は/etc/ppxp/conf/に、一般ユーザーの場合は~/.ppxp/conf/に、保存することになる。ここでは一般ユーザーなので、後者になる。接続先がわかりやすいよう名前を付けておこう（**画面9**）。

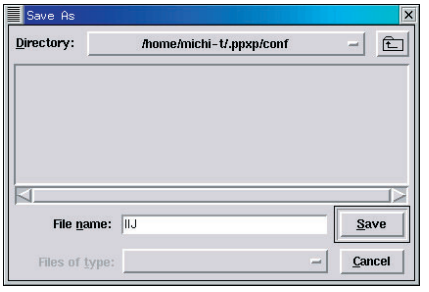
### ■ 接続テスト

PPxPの設定が終了したら、接続テストをしてみよう。tkPPxPを起動する。保存した設定ファイル名を引数として指定する。

```
$ tkppxp IIJ
```

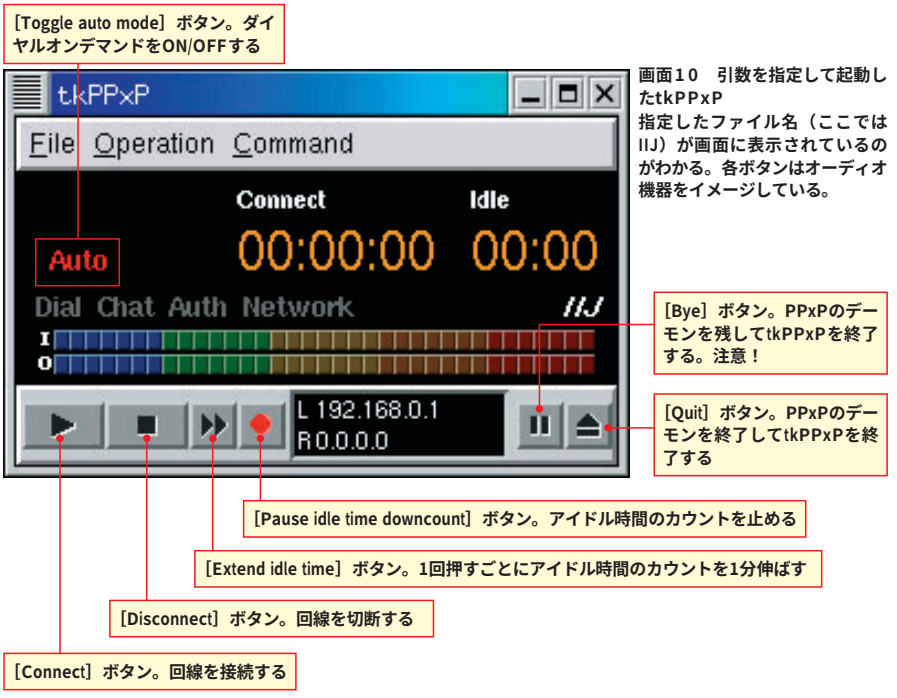
すると、tkPPxPが、指定した設定ファイルの内容を読み込んだ状態で起動する。ここで［Connect］ボタン（オーディオのPLAYボタンを模したもの）をクリックすれば接続する（**画面10**）。

Dial→Chat→Auth→Networkと、文字が順に点灯していき、正しく接続されると、プロバイダのゲートウェイのIPアドレス（R）と、こちらに割り




画面9　設定ファイルの保存
一般ユーザーの設定ファイルは、ホームディレクトリの.ppxp/conf/に保存される。ここでは、分かりやすいようにプロバイダ名（IIJ）を付けている。


画面11　接続した状態のtkPPxP
データが流れると、バーグラフにデータ量が表示される。Remote（R）ピアのアドレスとLocal（L）ピアのアドレスが表示されている。




画面10　引数を指定して起動したtkPPxP
指定したファイル名（ここではIIJ）が画面に表示されているのがわかる。各ボタンはオーディオ機器をイメージしている。

振られたIPアドレス（L）が表示される（**画面11**）。

この状態になれば、Linuxマシンからはインターネットにアクセスが可能だ。たとえばNetscape Navigatorを起動すればWebのブラウズが可能になる。

終了時に注意が必要なのは、［Bye］ボタンで回線も切断されたものと間違えないこと。このボタンは、tkPPxPが終了するだけで、PPxPのデーモンは機能したまま（つまり接続中ならば、回線は接続されたままになる）である。間違って押した場合は、もういちどtkPPxPを起動して正しく終了しよう。回線を切断して終了する場合には［Quit］ボタンを押す。

### ■ ダイヤルオンデマンド

tkPPxPで、ダイヤルオンデマンドをONにしておくと、インターネットへの接続要求があったときに自動的に接続することができる。アイドル時間を設定しておけば、勝手に接続して勝手に切断してくれるので便利だ。もちろん、ルータとして利用する際にも、LAN側からの接続要求に応じて、ダイヤルするようになる。

またフレッツISDNを利用している場合、常時接続とはいっても、たまに回線が切れることがある。こうした場合にも、ダイヤルオンデマンドをONにしておけばいちいち手動で再接続しなくて済む。

ダイヤルオンデマンドは、tkPPxPの［Toggle auto mode］の「Auto」という文字をクリックすることでON／OFFが可能だが、設定ファイルの1行目に、

```
auto on
```

と記述しておけば、設定読み込み時に自動的にONになる。

## Linuxマシンへの NICの装着と設定

次に、LinuxマシンにNICを装着しよう。電源を切ってから本体を開け、スロットにNICを装着する。その後電源を入れる。59ページで紹介したNICなど、最近の製品ならば、自動的に認識してくれるはずだ。

しかし、ハードウェアとしては認識していても、まだネットワークプログラムからは正しく認識されていない状態だ。そこで次は、ネットワークの設定を行おう。

この設定はlinuxconfなどのGUIの設定ツールを使っても設定できるのだが、今回はテキストエディタで設定ファイルを編集していく。GUIだから簡単というわけではなく、かえって煩雑に感じるからだ。

▼リスト1　/etc/sysconfig/network

```
NETWORKING="yes"          ← 必ずyesに
HOSTNAME="afb.localdomain"  ← このLinuxマシンの名前
DOMAINNAME="localdomain"    ← このLANのドメイン名
FORWARD_IPV4="yes"          ← Linuxをルータとして利用する場合。そうでなければ"no"に
GATEWAYDEV="eth0"           ← LAN側のNICのデバイスeth0を指定
GATEWAY="192.168.0.1"       ← ゲートウェイのアドレスを指定
```

▼リスト2　/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE="eth0"             ← デバイス名
IPADDR="192.168.0.1"      ← このNICのIPアドレス。他の機器と重複しないように
NETMASK="255.255.255.0"   ← 末尾0～255を利用するためのマスク
NETWORK="192.168.0.0"     ← 末尾0はネットワーク全体
BROADCAST="192.168.0.255" ← 末尾255はブロードキャスト
ONBOOT="yes"              ← ここをyesに
BOOTPROTO="none"
```

▼リスト3　/etc/HOSTNAME

```
afb.localdomain   ← このLinuxマシンの名前
```

▼リスト4　/etc/hosts

```
127.0.0.1       localhost  localhost.localdomain   ← 必要
192.168.0.1     afb.localdomain  afb   ← このLinuxマシンの定義
192,168.0.2     win.localdomain  win    ← このあとつなぐWindowsマシンの定義
```

▼リスト5　/etc/host.conf

```
order hosts,bind   ← まずhosts、次いでbindの順で名前解決
multi on
```

▼リスト6　/etc/resolv.conf

```
nameserver 210.130.x.x   ← プロバイダのネームサーバのIPアドレス
nameserver 210.130.x.x   ← プロバイダのネームサーバのIPアドレス
```



# 入門・Linux LANの設定

### ■ ネットワーク設定

編集する必要があるファイルは、すべてディレクトリ/etc以下に収められている。

```
/etc/sysconfig/network
/etc/sysconfig/network-scripts/
ifcfg-eth0
/etc/hosts
/etc/host.conf
/etc/HOSTNAME
/etc/resolv.conf
```

まず全体の設定として、/etc/sysconfig/networkを**リスト1**のように設定する。ここで、HOSTNAMEは、そのLinuxマシンに付ける名前である。例では「afb.localdomain」としているが、自由に付けていい。FORWARD_IPV4については、Linuxマシンをルータとして利用する場合はyesにするが、そうでない場合はnoでいい。ルータの設定については次節で紹介する。

GATEWAYDEVはLAN側のNICのデバイスeth0を、GATEWAYにはゲートウェイのIPアドレスを指定する。ISDNダイヤルアップルータやxDSLルータを利用している場合は、そのアドレスを、このLinuxマシンをルータとして利用する場合は、このマシンに割り振るアドレスを指定する。

次にNICのデバイスeth0について、/etc/sysconfig/network-scripts/ifcfg-eth0を**リスト2**のように設定する。ここでは、IPADDRを192.168.0.1にしているが、すでにルータを利用している場合は注意が必要だ。それはルータ自身のIPアドレスが192.168.0.1に割り当てられていることが多いからだ。そのような時は別のアドレス、たとえば192.168.0.2などに設定する必要がある。

/etc/HOSTNAMEには、このLinuxマシンの名前を書く（**リスト3**）。/etc/hostsには、IPアドレスと名前の関係付けを記述する（**リスト4**）。LANに接続されるマシンのIPアドレスと、名前、省略形をタブ区切りで記述する。一番上のlocalhostの記述は必ず残しておくこと。

/etc/host.confでは、名前解決にhostsファイルを先に使用するように**リスト5**のように設定しておく。

resolv.confにはプロバイダのネームサーバアドレスを記述しておこう（**リスト6**）。

設定が終わったらネットワークを再起動する。Windowsと違って、システムを再起動する必要はない。デバイスloに続き、eth0が機能し始める。

```
# /etc/rc.d/init.d/network restart
Bringing up interface lo [  OK  ]
Bringing up interface eth0 [  OK  ]
```

ifconfigコマンドを実行してデバイスeth0を確認してみよう。eth0が表示されて、正しくIPアドレスが振られていればOKだ。

## 2枚目のNICの装着と設定

現在、xDSLで使われるxDSLモデムはPCとの接続にEthernetを使う。ただし上位プロトコルはPPPである。したがって、PPP over Ethernet（PPPoE）という仕組みを利用することになる。

CATVやxDSLの場合、Linuxをルータとして利用するためには、LAN用とモデム接続用の2枚のNICを装着する必要がある。2枚目のNICはデバイスeth1として機能する。eth1のIPアドレスは、ダイヤルアップ時と同様、プロバイダからダイナミックに割り振られることが多い。

設定で必要なのはファイル/etc/sysconfig/network-scripts/ifcfg-eth1だ（**リスト7**）。IPアドレスやネットマスクなどは動的に割り振られるので空欄。BOOTPROTOにはdhcpを指定しておく。

### ■ PPPoEの組み込み

デバイスeth1を介して、PPPoEを利用するためのソフトウェアにはいくつかあるが、ここでは、Roaring Penguin Software社のRP-PPPoEを使う。RP-PPPoEは、pppdを利用して動作するPPPoE接続用のプログラムである。pppdはほとんどのディストリビューションに標準でインストールされているので、問題はないだろう。

同社のWebサイトからrp-pppoe-2.8-1.i386.rpmをダウンロードし、以下の手順でインストールする。

```
# rpm -Uvh rp-pppoe-2.8-1.i386.rpm
```

インストールしたら、ISPとの認証に関する設定を行う。

```
# /usr/sbin/adsl-setup
```

▼リスト7　/etc/sysconfig/network-scripts/ifcfg-eth1

```
DEVICE="eth1"          ← デバイス名
IPADDR=
NETMASK=
NETWORK=
BROADCAST=
ONBOOT="yes"           ← ここをyesに
BOOTPROTO="dhcp"       ← ここをdhcpに
```

と入力し、順に質問される、ISPとの接続に使うユーザー名、NICのデバイス名（ここではeth1）、アイドル時間の設定（常時接続ならno）、プロバイダのネームサーバ名、パスワード、ファイアウォールの設定について入力する。入力が終了すると、設定ファイルに書き込まれる。準備ができたら、

```
# /usr/sbin/adsl-start
```

で、接続開始だ。回線を切断する場合は、

```
# /usr/sbin/adsl-stop
```

と入力する。

なお、RP-PPPoEは、インストール時に/etc/rc.d/init.d/adslというスクリプトを作成する。Linux起動時に自動的に接続したい場合は、これを利用することができる。

## Windowsマシンのネットワーク設定

WindowsマシンへのNICの装着は、Linuxマシンと同じだ。また、ハードウェアの認識については、最近の製品ならプラグアンドプレイで自動認識されるので悩むことはないだろう。ここでは、ハードウェアの認識まで済んだ後のネットワーク設定と接続の確認について触れておく。

### ■ Windows 9x／Meの設定

Windows 9x／Meのコントロールパネルから［ネットワーク］を起動する。［現在のネットワークコンポーネント］のリストから「TCP/IP->（NICの名前）」を選択して、［プロパティ］ボタンをクリックすると［TCP/IPのプロパティ］のダイアログボックスが表示される。

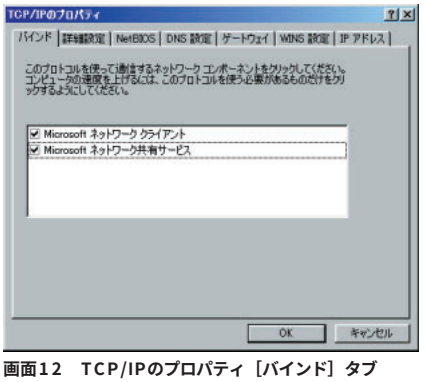
［バインド］タブで［Microsoftネットワーククライアント］と［Microsoftネットワーク共有サービスにチェック］しておく（**画面12**）。

［DNS設定］タブには、ネームサーバを指定する。ISDNダイヤルアップルータやxDSLルータを利用している場合はルータのアドレスを、Linuxをルータとして利用する場合は、今回、DNS（BIND）の設定を行わないので、プロバイダのネームサーバアドレスを指定しておこう（**画面13**）。

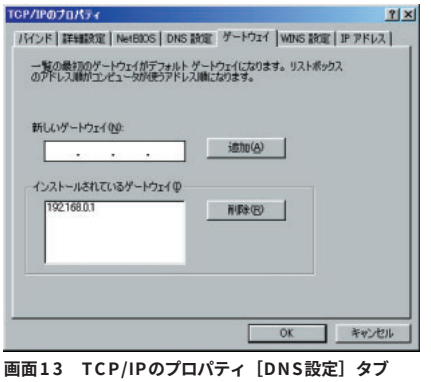
［ゲートウェイ］には、ルータのアドレスを指定する。ISDNダイヤルアップルータやxDSLルータを利用している場合はそのアドレスを、Linuxをルータとして利用する場合はLinuxマシンに割り振ったアドレスを、ここに指定する（**画面14**）。

［IPアドレス］には、プライベートアドレスを指定する（**画面15**）。

設定が終わったら、［OK］ボタンをクリックしてWindowsを再起動する。

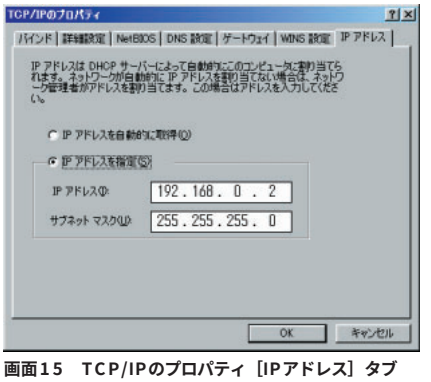



画面12　TCP/IPのプロパティ［バインド］タブ

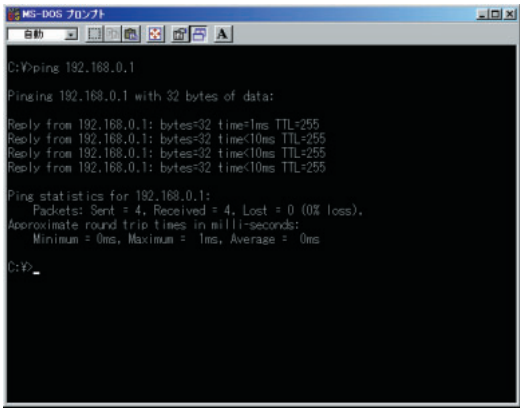



画面13　TCP/IPのプロパティ［DNS設定］タブ


画面14　TCP/IPのプロパティ［ゲートウェイ］タブ




画面15　TCP/IPのプロパティ［IPアドレス］タブ




画面16　MS-DOSプロンプトでpingコマンドを実行する。名前を指定してもダメなときは、hostsファイルを用意する必要がある。



# 入門・Linux LANの設定

再起動したら、ネットワーク接続チェックをしてみよう。［スタート］メニューから［プログラム］－［MS-DOS］プロンプトを起動し、コマンドラインから以下のように入力する。

```
C:¥> ping 192.168.0.1
```

**画面16**のように、結果が返ってくれば接続されている。パケットロスが0%ならば、完璧だ。

ダイヤルアップルータを利用している場合、LAN内の名前解決ができないことがある。たとえば、

```
C:¥> ping afb
```

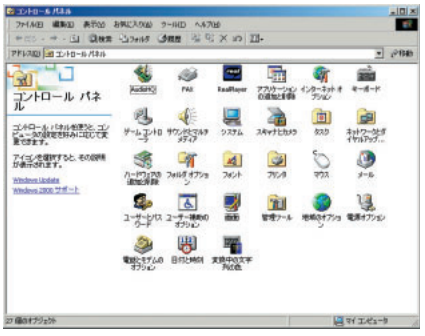
などと、LANに接続されたLinuxマシンの名前を指定しても、解決できずに応答がない（Linuxマシンをルータにしている場合は可能）。これは、LinuxマシンでDNSサービスを設定すればいいのだが、本特集では解説していない。DNSサービスを設定するまでは、Windows起動ドライブのディレクトリ\Windowsの下にhostsという名のファイルを置くといい。内容は**リスト4**と同じものだ。

## ■ Windows 2000の設定

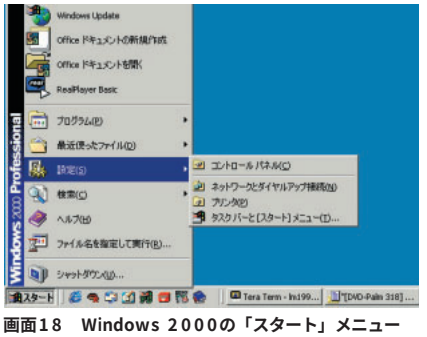
Windows 2000の場合も、まずコントロールパネル（**画面17**）から［ネットワークとダイヤルアップ接続］のフォルダを開く。スタートメニューの［設定］からも同じフォルダを開くことができる（**画面18**）。

LANの設定は、［ローカル エリア接続］（**画面19**）から行う。アイコンをダブルクリックすると、LANの接続状態が表示される（**画面20**）。ここで［プロパティ］ボタンをクリックすると、利用できるサービスとプロトコルが表示される（**画面21**）。必要なコンポーネントは、［Microsoftネットワーク用クライアント］、［Microsoftネットワーク用ファイルとプリンタ共有］、そして［インターネットプロトコル（TCP/IP）］の3つだ。

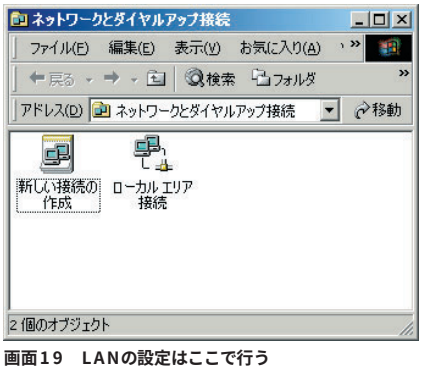
さらに［インターネットプロトコル（TCP/IP）］を選択し、右下の［プロパティ］ボタンをクリックして、IPアドレスの設定を行う（**画面22**）。［IPアドレス］には、プライベートアドレスを指定する。［デフォルトゲートウェイ］には、ルータのアドレスを指定する。ISDNダイヤルアップルータやxDSLルータを利用している場合はそのアドレスを、Linuxをルータとして利用する場合はLinuxマシンに割り振ったアドレスを、ここに指定する。［DNSサーバー］は、今回は自前のDNSを利用しないので、プロバイダのネームサーバアドレスを指定しておく。



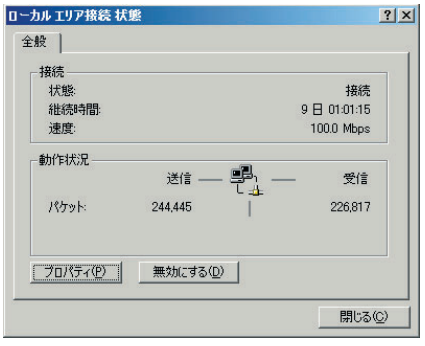
画面17　Windows 2000のコントロールパネルフォルダ
「ネットワークとダイヤルアップ接続」フォルダを開く。



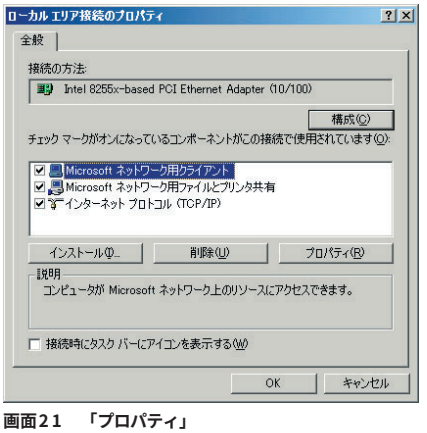
画面18　Windows 2000の「スタート」メニュー
こちらからも選択が可能。



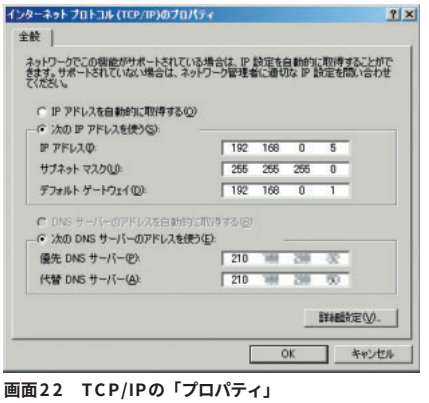
画面19　LANの設定はここで行う



画面20　「ローカルエリア接続」状態
「プロパティ」ボタンをクリックして、設定を行う。



画面21　「プロパティ」
Windows 9xと同様のコンポーネントが必要だ。



画面22　TCP/IPの「プロパティ」
IPアドレス、ゲートウェイ（ルータ）、DNSサーバを設定する。

# ルーティングの設定

文：竹内 充彦／Text：Michihiko Takeuchi

複数のネットワークを接続する際、ネットワーク間のパケットの流れを制御することをルーティングと言う。LANとインターネットを接続する場合には、ルーティングをする機材、すなわちルータが必要となる。

たとえば、同一のLANに接続されたコンピュータ同士でパケットを交換する際に、それをインターネットに流す必要はない。しかし、LANに接続されたコンピュータとインターネット上のコンピュータの間でパケットを交換する際には、LANとインターネットの間を中継する必要がある。このように、パケットの宛先や内容によって、経路制御を動的に行うのがルータの仕事だ。現在では、ISDNダイヤルアップルータが最も身近なルータだろう。

そうした専用機を使わずとも、ルーティングはLinuxで行うことができる。前節で紹介したように、モデム／TAとNICを装着したLinuxマシンは、まさにLANとインターネットの中継地点に位置することになる。

ここでは、モデム／TAを装着したLinuxマシンで、ルーティングを行うための諸設定を解説する。

また、パケットの種類により出入りを制限する簡易ファイアウォール、パケットフィルタリングについても簡単に紹介する。

そして、端末型ダイヤルアップ接続を利用して、LANをインターネットに接続するIPマスカレードについても解説する。

## パケット転送

前節でモデム／TAとNICを装着し、それぞれ機能するように設定したが、まだその2つのネットワークインターフェイス間は中継されていない。

互いのインターフェイス間をパケットが転送されるように設定する必要があるのだ。

### ■ パケット転送の開始

パケット転送を開始するには、70ページで紹介した/etc/sysconfig/network内の以下の項目を設定をする。

```
FORWARD_IPV4="yes"
```

また、Red Hat Linux 6.2以降をベースにしたディストリビューションの場合は、/etc/sysctl.confの内容のうち次の1行を書き換える必要がある。デフォルトは0となっているので1に変更しよう。

```
net.ipv4.ip_forward = 1
```

設定を終えたらシステムを再起動し、/proc/sys/net/ipv4/ip_forwardの内容を表示してみよう。1と表示されれば、パケット転送がされている。0と表示されれば、転送されていない。もし0のままならば最後の手段として手動で、

```
# echo 1>/proc/sys/net/ipv4/ip_forward
```

と、強制的に1に書き換える方法もある。この方法でしか書き換えられない場合は、毎回起動のたびに実行するのも面倒なので、/etc/rc.d/rc.localなどに記述しておくといい。

### ■ 転送されないプロトコル

上記の設定で、HTTP、Telnetプロトコルなどは転送される。しかし、後述するIPマスカレードがネックとなり、いくつかのプロトコルが通らない。

たとえば、FTPや、IRC（チャット）、RealPlayer（ビデオストリーミング）、Quake（ゲーム）などのプロトコルは転送してくれない。これらのうちいくつかは、プロトコルを通すためのモジュールが用意されているので、利用形態に合わせて適宜組み込む必要がある。

モジュールは、ディレクトリ/lib/modules/2.2.＊/ipv4の下に置かれてい

```
# lsmod
Module                  Size  Used by
ip_masq_vdolive         1336   0  (unused)
ip_masq_raudio          3000   0  (unused)
ip_masq_quake           1360   0  (unused)
ip_masq_ftp             4184   0  (unused)
ethertap                2388   1  (autoclean)
lockd                  31176   1  (autoclean)
sunrpc                 52964   1  (autoclean) [lockd]
ne                      6736   1  (autoclean)
8390                    6076   0  (autoclean) [ne]
```

画面23　lsmodの結果
モジュールip_masq_＊が組み込まれている。



# 入門・Linux LANの設定

る（＊はバージョンにより異なる）。代表的なモジュールを以下に示す。

```
ip_masq_cuseeme.o
ip_masq_ftp.o
ip_masq_irc.o
ip_masq_quake.o
ip_masq_raudio.o
ip_masq_vdolive.o
```

上から順に、CuSeeMe、FTP、IRC、Quake、RealPlayer、VdoLiveに対応している。組み込み手順は以下の通り。最後の“.o”はいらないことに注意しよう。

```
# depmod -a
# modprobe ip_masq_ftp
```

組み込んだらlsmodコマンドで確認してみよう（**画面23**）。これで、LAN上のWindowsマシンからFTPやRealPlayerが利用できるようになった。

モジュールの組み込みは、/etc/rc.d/rc.localなどに記述して、Linuxが起動したら自動的に組み込まれるようにしておくと便利だ。

## パケットフィルタリング

いよいよLANとインターネットの間でパケットが交換されるわけだが、そうなると直面する問題がある。それは内部からの無用なパケットの流出と、外部からの歓迎しないパケットの流入だ。

ダイヤルアップ接続環境で内部からの無用なパケットの流出が起きると、必要もないのに電話をかけることになり、無駄に電話代が増えてしまう。典型的な例としては、WindowsがLANで利用するNetBIOS over TCP/IPが挙げられる。次節で解説するSambaは、このプロトコルを利用する。NetBIOS over TCP/IPのパケットを外へ流そうとして、電話をかけてしてしまうことがあるのだ。

外部からのパケットの流入に関しては、LAN上のサーバへのアクセスとクラッキングがある。今回は、LAN上に外部からアクセス可能なサーバを置かないので、外部からのアクセスはすべて拒否すればいい。もちろん内部から外部へのリクエストに対する応答は、IPマスカレードにより受け取れるので問題はない。

このように、パケットの出入りをふるいにかけることをパケットフィルタリングと言う。パケットフィルタリングの設定には、ipchainsというコマンドを使う。

### ■ ipchains

ipchainsは、かなり細かくパケットの出入りを制御できる。ipchainsコマンドの典型的な書式は以下のとおりだ。

```
ipchains -[ADC] <チェーン> <ルール>
```

実際には、たとえば次のようにコマンドを実行する。

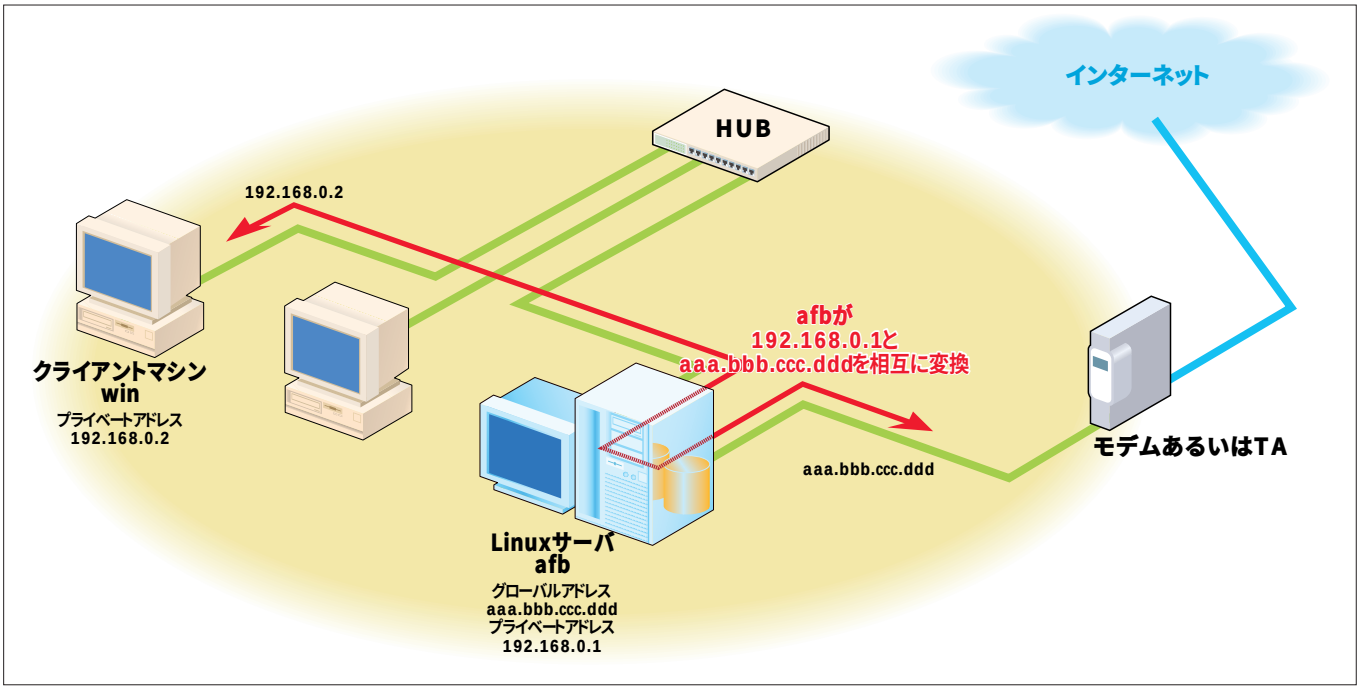



図1　IPマスカレードを利用したインターネット接続

```
ipchains -A input -s 192.168.0.0/24
-j DENY
```

この例では、input（というチェーン）に－s以降（というルール）を追加（－A）するという意味だ。順に解説しよう。

チェーンは、ルールを持つフィルタで、複数のチェーンを鎖のようにつなぐことで、複雑なフィルタリングを実現する。ipchainsに標準で用意されているチェーンには以下のものがある。

```
input   入力
output  出力
forward 転送
```

チェーンはこれ以外にも自分で作成することが可能だが、初歩的なLANではそこまで必要ないので、ここでは省略する。これらのチェーンにルールを追加／削除して、パケットの出入りを制御する。

ルールの追加削除は以下のオプションで指定する。

```
－A     追加
－D     削除
－C     コピー
```

ルールには、パケットの種類や宛先、送信元などを元にして、動作を指定する。先の例ではルールは2つの項目からなる。1つは、

```
-s 192.168.0.0/24
```

で、これは送信元のアドレスが192.168.0.1～255という意味だ（/24はサブネットマスクの表記法）。もう1つは、

```
-j DENY
```

で、これは拒否するという意味だ。つまり総合すると、「192.168.0.1～255から送出されたパケットは拒否する」ということになる。これをinputというチェーンに追加しているのだ。ルール記述子を**表1**に示しておく。

しかし、すべての項目をipchainsで1行ずつ設定していくのは大変である。そこで、セキュリティポリシーというものが登場する。これは、「すべて禁止で特定のものだけ認める」か「すべて許可して特定のものだけ禁止するか」というセキュリティに対する根本的な考え方だ。

ipchainsでは－Pを指定することで、セキュリティポリシーを変更できる。デフォルトはすべて許可である。たとえば、以下のように設定する。

```
ipchains -P forward DENY
```

すると、forwardチェーン（つまり転送）に関しては、すべて禁止になる。こうしておいてから、転送したいパケットについてのルールを、1つずつACCEPTしていくという手順になる。

ipchainsで指定した設定は、－Lオプションで確認することもできる。

```
# ipchains -L
```

### ■ つなげる時は忘れずに

ipchainsの基本を知ったところで、今回のダイヤルアップ接続型のLANで、これだけは設定しておきたいというフィルタリングを紹介しておく（**リスト8**）。

1行目で、外部からプライベートIPアドレスを騙って侵入してくるパケットを拒否する（主にクラッキング対策）。2行目で、パケット転送のポリシーをすべて禁止に。次の2行は、NetBIOS（プロトコルtcpとudpにおいて、ポート137～139に対してLANから送信されたパケット）の外部（tap0）への転送を禁止している。なおWindows 2000を使っている場合は、最後の2行を追加し、ポート445にも同様の設定をする必要がある。

この設定を/etc/rc.d/rc.localに記述しておけば、起動時に有効になる。

| | |
|---|---|
| -s | ソースアドレス（送信元のIPアドレス） |
| -d | デスティネーションアドレス（宛先のIPアドレス） |
| -p | プロトコル名（tcp、udp、icmp等） |
| -j | ターゲット（ACCEPT、DENY、REJECT、MASQ、別のチェーンなど） |
| -i | インターフェイス（eth0、tap0など） |

表1　ipchainsのルール記述子

▼リスト8　ipchainsによるパケットフィルタリング

```
ipchains -A input -i tap0 -s 192.168.0.0/24 -d 0/0 -j DENY
ipchains -P forward DENY
ipchains -A forward -i tap0 -p tcp -s 192.168.0.0/24 -d 0/0 137:139 -j DENY
ipchains -A forward -i tap0 -p udp -s 192.168.0.0/24 -d 0/0 137:139 -j DENY
ipchains -A forward -i tap0 -p tcp -s 192.168.0.0/24 -d 0/0 445 -j DENY
ipchains -A forward -i tap0 -p udp -s 192.168.0.0/24 -d 0/0 445 -j DENY
```

## IPマスカレード

コンピュータをインターネットに接続するには、世界中で唯一のIPアドレス（グローバルIPアドレス）を割り当てなければならない、というのが大前



# 入門・Linux LANの設定

提だ。本来ならばLANに接続されたすべてのコンピュータも、インターネットに接続するのであれば、それぞれを識別できるようにグローバルIPアドレスを割り当てなければならない。しかし、グローバルIPアドレスは、JPNICやISPにより管理されているため、勝手に割り当てることはできない。

ダイヤルアップ接続ユーザーとしては、LANに接続された複数のコンピュータからインターネットにアクセスできれば便利だ。しかしグローバルIPアドレスは、ISPから割り当てられた1個しかない。この悩みを解決するのがIPマスカレードだ。

これは、LAN上のコンピュータが外部へアクセスしようとした場合に、ルータがプライベートアドレスをマスクして、ISPから割り当てられたグローバルIPアドレスに変換して送出し、それに対する応答を受け取ったら、アドレスのマスクを解除してLAN上のコンピュータに戻すというものだ。

こうすることで、LAN側から見れば、すべてのコンピュータがインターネットにアクセスでき、インターネット側から見れば、1台のコンピュータ（ルータ）だけがインターネットにアクセスしているように見える。

IPマスカレードを実現するためのコマンドも、前出のipchainsである。ターゲットにMASQを指定する。

192.168.0.1～255のアドレスをマスクして外部にアクセス可能にするコマンドラインは**リスト9**のようになる。

例によって、これを/etc/rc.d/rc.localに追加しておこう。これで、LANに接続されたWindowsマシンから自由にインターネットにアクセスできるようになる。

## カーネル2.4とiptables

2001年初頭に安定版カーネルの最新版、2.4が発表された。2001年2月末現在、カーネル2.4の正式版を採用したディストリビューションはないが、今後は、ほとんどのディストリビューションがカーネル2.2から2.4へ移行することは明らかだ。そこで、カーネル2.4でパケットフィルタ／IPマスカレードする際の注意点をまとめておこう。

カーネル2.4では、パケットフィルタやIPマスカレードを実現する仕組みが全面的に書き直され、DoSアタック（サービス不能攻撃）への対応などもできるようになった。これにあわせ、設定コマンドも、ipchainsからiptablesというパッケージに変更された。

iptablesの書式は、ipchainsとよく似ているが、違っている点も多いので気をつけよう。たとえば、パケットを拒否するときに、ipchainsでは“ -j DENY ”と指定していたのが、iptablesでは“ -j DROP ”にする。また、ipchainsでは、対象とするインターフェイスを“ -i eth0 ”のように指定した。iはinterfaceを表しているのだろう。しかし、iptablesでは、パケットが入ってくるインターフェイスと出ていくインターフェイスを個別に指定できるようになったため、それぞれ、-iオプション、-oオプションを使うことになった。これにより、より細かい設定が可能になったわけだが、“ -o eth0 ”とするべきところで、“ -i eth0 ”と指定すると思った通りに動作しないので注意しよう。

**リスト10**は、iptablesを使って、**リスト8**、**9**と同じ内容を設定する方法だ。PPxPは、カーネル2.4に正式対応していないため、この例では外向けのインターフェイスとしてtun0を指定している。1行目から6行目までがパケットフィルタの設定となっており、最後の行はIPマスカレードの設定だ。先ほど説明した変更点のほか、宛先ポートを指定するには、--dportオプションが必要になっている（送信元のポートは--sportオプションで指定する）。IPマスカレードの行では、送信元のIPアドレスが192.168.0.0/24で、tun0というインターフェイスから出ていくパケットがIPマスカレードされるように設定している。

▼リスト9　IPマスカレードの設定

```
ipchains -A forward -s 192.168.0.0/24 -d 0.0.0.0/0 -j MASQ
```

▼リスト10　iptablesによるパケットフィルタリング／IPマスカレードの設定

```
iptables -A input -i tun0 -s 192.168.0.0/24 -d 0/0 -j DROP
iptables -P forward DROP
iptables -A forward -o tun0 -s 192.168.0.0/24 -p tcp -d 0/0 --dport 137:139 -j DROP
iptables -A forward -o tun0 -s 192.168.0.0/24 -p udp -d 0/0 --dport 137:139 -j DROP
iptables -A forward -o tun0 -s 192.168.0.0/24 -p tcp -d 0/0 --dport 445 -j DROP
iptables -A forward -o tun0 -s 192.168.0.0/24 -p udp -d 0/0 --dport 445 -j DROP

iptables -t nat -A POSTROUTING -o -tun0 -s 192.168.0.0/24 -j MASQUERADE
```

# Sambaの導入

文：竹内 充彦／Text：Michihiko Takeuchi

Sambaは、UNIX系OS上でWindowsのファイル共有やプリンタ共有を提供するサーバアプリケーションだ。LANに接続されたLinuxにSambaを導入することで、Linuxマシンのハードディスク上にあるファイルをWindowsから読み書きしたり、Linuxマシンに接続されているプリンタをWindowsから利用したりできるようになる。

## Sambaのインストール

Sambaは、いまや標準インストールされていないディストリビューションはほとんどないのではと思わせるほど、定番となったサーバアプリケーションである。もし、最近のディストリビューションパッケージをインストールしたのであれば、きっとインストールされているはずだ。ただし、バージョンは若干古い可能性がある。

Samba日本語版の最新版は、samba-2.0.7-ja-2.2である。バージョン2.0.7は、それ以前のバージョンで見られた同一LAN上でWindows 2000を利用している際に起きていた問題点を、すべて修正している。まずは、自分のシステムにインストールされているSambaのバージョンを確認してみよう。Ktermを開いて、コマンドラインからrpmコマンドの-qオプションを指定する。

```
# rpm -q samba
```

バージョンが2.0.7-ja-2.2より古いものであれば、最新バージョンにアップデートしよう。最新版のtarボールは、今月号のCD-ROMに収録されているので、それを使うことにする。

```
# mount /mnt/cdrom
# cd /mnt/cdrom/Linuxmag/Special/Samba
# cp samba-2.0.7-ja-2.2.tar.gz /tmp
# cd /tmp
# tar xvzf samba-2.0.7-ja-2.2.tar.gz
# ./configure --with-i18n-swat
# make
# make install
```

## Sambaの設定

Sambaの設定ファイルは/etc/smb.confだ。これはテキストエディタで書き換えることもできるし、SWATというツールを利用して、GUIから書き換えることもできる。SWATを利用すると、Webブラウザ上からsmb.confの内容を書き換えることができる。

ここでは、SWATを利用してみよう。

▼リスト11　/etc/inetd.confのSWATの設定

```
swat    stream  tcp     nowait.400      root    /usr/sbin/tcpd  /usr/sbin/swat
```

▼リスト12　/etc/servicesのSWATの設定

```
swat    901/tcp         # Add swat service used via inetd
```

### ■ SWATの起動準備

SWATを利用するためには、ちょっとした設定が必要だ。SWATを利用するには、Webブラウザでポート901をアクセスする。したがって、Webブラウザからポート901に接続要求があったときに、SWATを起動するように設定しておくことと、ポート901自体を有効にしておくことが必要だ。

そのためにはまず、ファイル/etc/inetd.confに、**リスト11**の1行を追加する。次に、ファイル/etc/servicesの中から行頭の#でコメントアウトされている**リスト12**の行を探し（swatをキーにエディタで検索するといい）、行頭の#を削除し、有効にする。コメントアウトされた行が、見つからないときは、リストのように1行追加すればいい。

2つのファイルを修正したら、inetdを再起動する。psコマンドでinetdのプロセスIDを調べて、killコマンドで、そのプロセスIDにHUPシグナルを送る。

```
# ps ax | grep inet
  476 ?        S      0:00 inetd
 1875 pts/0    S      0:00 grep inet
# kill -HUP 476
```

### ■ SWATでの設定

SWATの設定ができたら、さっそくNetscape Navigatorを起動し、URLに以下を指定しよう。

```
http://localhost:901
```



ユーザー名とパスワードが求められるので、ユーザー名rootと、rootのパスワードをそれぞれ入力する。正しく入力すると、SWATの設定画面が表示されるはずだ（**画面23**）。

ちなみに、この画面にはWindowsからIEを使ってもアクセスできる。その場合は、SWATの動作するLinuxマシンのポート901にアクセスする。Linuxマシンに付けた名前がafbならば、

```
http://afb:901
```

のようにすればいい。IPアドレスを直接指定してもアクセスできる。

設定する項目を順に紹介していこう。とはいえ、SWATは非常によくできたツールで、設定項目には必ず説明が用意されている。そこでここでは、最低限の設定しなければいけない項目について紹介するにとどめる。より詳しく知りたい場合は、各項目の説明を参照してほしい。

Sambaの設定は、タイトルのすぐ下にある7つのボタンアイコンで行う。ホーム、全体設定、共有設定、プリンタ、動作状況、設定表示、パスワードだ。

### ■全体設定

全体設定では基本的な項目を設定する（**画面24**）。ここでは、少なくとも以下の項目を設定しよう。設定値については、**リスト12**の［global］セクションを参照してほしい。

“ workgroup ”は、Windowsネットワークに付ける名前だ。自由に付けていいが、ここで指定したものと後述するWindows側の設定「ワークグループ名」を一致させておく必要がある。デフォルトは“ SAMBA ”だ。

“ netbios name ”は、Windowsネットワーク上でのサーバ名だ。これも自由に付けていい。

“ server string ”には、このサーバの簡単な説明を書く。Windowsマシンのエクスプローラ上から参照できる。

“ log file ”には、Sambaの動作記録（ログ）を保存するファイル名を指定する。また“ max log size ”で、ログファイルのサイズの上限をKバイト単位で指定する。

“ coding system ”には、日本語ファイル名などの文字コードの扱いを指定する。

“ guest account ”で、ゲストユーザーのアカウントを指定する。既定値nobodyのままでかまわない。

“ security ”は、既定値“ user ”のままにしておくのが適当だ。同様に“ encrypt passwords ”も既定値“ yes ”のままにしておき、暗号化パスワードを使うようにしよう。

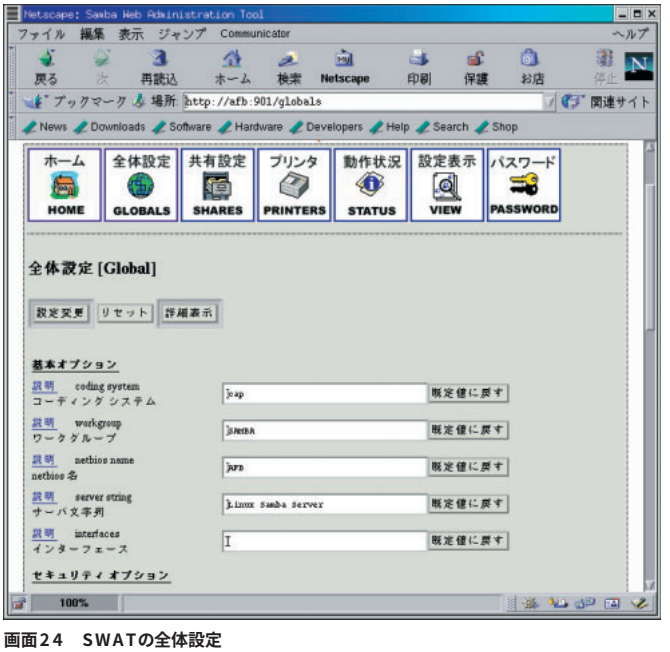
“ null passwords ”は、パスワードを設定しない（空文字）ユーザーを認めるかどうかを指定する。既定値は“ no ”だ。

### ■ 共有設定

ここでは、共有フォルダの設定を行う。まず、共有名を指定して、［新規作成］ボタンをクリックする。すると、共有フォルダの設定画面に変わる（**画面25**）。

“ comment ”には、Windowsのエクスプローラから見える、このフォルダ


画面23　SWATの設定画面
ページの上に並ぶ7つのボタンで画面を切り替えて各項目を設定する。




画面24　SWATの全体設定
［global］セクションを設定する。

に関する簡単な説明を指定する。自由に付けていい。

“ path ”は、共有するフォルダのパスを指定する。例外的にhomesの場合は各ユーザーのホームディレクトリになるので必要ない。

“ writable ”は、このフォルダを書き込み可能にするかどうかを指定する。

“ guest ok ”は、ゲストのアクセスを許可するかどうかを指定する。既定値は“ no ”だ。

“ browseable ”は、ワークグループ内の共有フォルダを一覧したときに、このフォルダを表示させるかどうかを指定する。既定値の“ Yes ”のままにしておいたほうが、Windowsのエクスプローラ上から、このディレクトリを探しやすい。

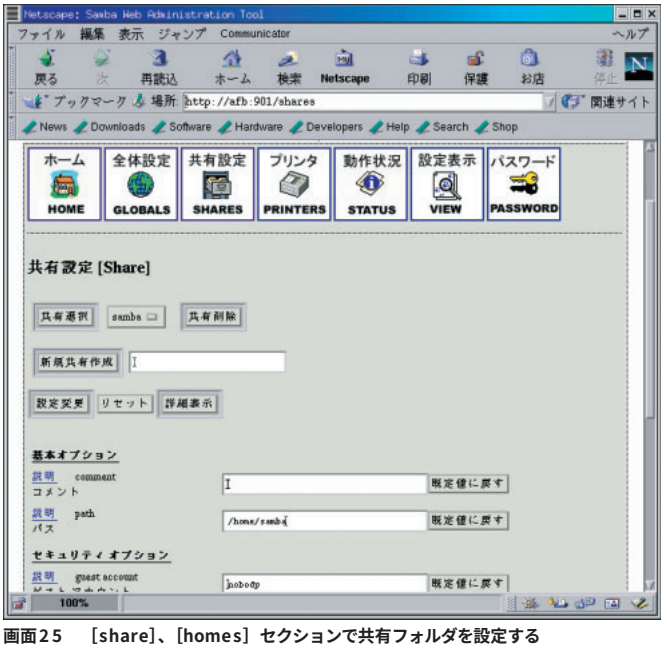
その他の設定内容は、**リスト13**の［homes］セクションと［tmp］セクションを参照してほしい。［homes］セクションは、各ユーザーのホームディレクトリ（/homesの下）の扱いを決めたものであり、［tmp］セクションでは/tmpを、ゲストも含めて誰でもアクセスできるようにしている。

### ■ プリンタ

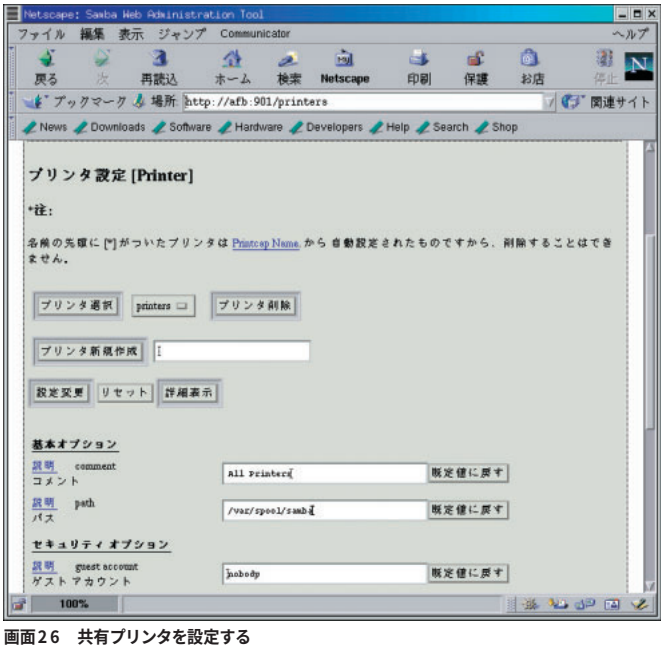
ここでは、共有するプリンタの設定を行う。まずプリンタ名を指定し、［プリンタ新規作成］ボタンをクリックする。このときプリンタ名に“ printers ”を指定すると、共有プリンタの既定値

▼リスト13　/etc/smb.conf（最低必要な部分のみ）

```
[global]
    workgroup = SAMBA                    ← ワークグループ名
    netbios name = afb
    server string = Linux Samba Server
    log file = /usr/local/samba/var/log.%m
    max log size = 50
    coding system = cap
    guest account = nobody
    security = user                      ← 今回のLANでは必ずuserに
    encrypt passwords = yes              ← 暗号化パスワードを使う
    null passwords = yes

[homes]                                  ← homesセクションではフォルダのpathを省略可
   comment = Home Directories
   read only = No                        ← writable = yesと同じ意味
   browseable = no                       ← 他のユーザーに見られないようにしている

[printers]
   comment = All Printers
   path = /var/spool/samba
   printable = yes

[tmp]
   comment = Temporary file space
   path = /tmp
   read only = no
   guest ok = yes                        ← ゲストユーザーもアクセスできる
   browseable = yes
```




画面25　［share］、［homes］セクションで共有フォルダを設定する




画面26　共有プリンタを設定する



# 入門・Linux LANの設定

である［printers］セクションができあがる（**リスト12**参照）。そして、［printers］の設定画面に変わる（**画面26**）。

“ comment ”には、Windowsのエクスプローラから見える、このプリンタに関する簡単な説明を指定する。自由に付けていい。

“ path ”には、Sambaが印刷データのスプールファイルを置くディレクトリを指定する。

“ printable ”は、Sambaを経由してのプリントを許可するかどうかを指定する。当然“ Yes ”だ。

その他、プリンタについても、“ guest ok ”が指定できるので、誰でも印刷できるように設定するならば“ Yes ”にするといい。

### ■ ユーザーの設定

Sambaのユーザーは、Linuxのユーザーとは別に管理される。したがって、Sambaのユーザーは必ずしもLinuxのユーザーと一致している必要はない。

とはいえ多くの場合、Linuxに登録してあるユーザーと、Sambaユーザーは重なるだろう。そこでまず、Linuxのユーザー登録ファイル/etc/passwdからSamba利用者のユーザー登録ファイル/etc/smbpasswdを作成する。続いてSambaの新規ユーザーの登録方法を紹介しよう。

Linuxのpasswdからsmbpasswdを作成するには、/usr/bin/mksmbpasswd.shを使う。使い方は以下のようになる。

```
# cat /etc/passwd | mksmbpasswd.sh
> /etc/smbpasswd
# chmod 660 /etc/smbpasswd
```

smbpasswdに新規にユーザー登録をするには、smbpasswdコマンドに−aオプションとユーザー名を指定して行う。ユーザー“ foo-n ”の登録は以下のようになる。

```
# smbpasswd -a foo-n
New SMB password: パスワードを入力
Retype new SMB password: パスワードを入力
Added user foo-n.
```

もちろんここで設定するSambaユーザーのパスワードは、Windowsにログオンするときのパスワードと一致していなければならない。

### ■ プリンタの設定

プリンタの設定は簡単だ。Linuxマシンのパラレルポートにプリンタを接続し、プリンタ設定ファイル/etc/printcapを用意する。printcapは、テキストエディタを使って**リスト14**のように記述しておけばいい。ほぼデフォルトの設定のままである。

リスト中の：sdで始まる行は、印刷データを一時的に溜め込むスプールファイルを指定している。この場合は、/var/spool/lpd/lpというファイルにスプールする。：mxで始まる行は、印刷データのサイズ制限で、#0にしておくと無制限である。

Sambaでは、Windows上のプリンタドライバで処理済みのデータをそのまま加工せずにプリンタに送り出すだけなので、この設定だけで十分だ。

## Sambaの起動

パケットフィルタリングのところでも触れたが、Sambaが利用するNetBIOS over TCP/IPというプロトコルは、137～139、445（Windows 2000以降）のポートを利用する。このポートが有効になっているかどうかは、ファイル/etc/servicesで確認しておこう。**リスト15**のような行があり、かつ、コメントアウト（行頭に#）されていなければ有効だ。

では、Sambaを起動しよう。Sambaは、2つのプログラムからなる。/usr/sbin/smbdと/usr/sbin/nmbdだ。これらを直接実行するならば以下のようにする。

```
# /usr/sbin/smbd -D
# /usr/sbin/nmbd -D
```

しかし、通常はスクリプトファイル/etc/rc.d/init.d/smbをstartオプションを指定して実行する。

▼リスト14　/etc/printcap

```
lp:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:
```

▼リスト15　/etc/servicesのSambaに関する記述

```
netbios-ns      137/tcp         # NETBIOS Name Service
netbios-ns      137/udp
netbios-dgm     138/tcp         # NETBIOS Datagram Service
netbios-dgm     138/udp
netbios-ssn     139/tcp         # NETBIOS session service
netbios-ssn     139/udp
```

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

標準でSambaがインストールされている場合は、/etc/rc.d/rc5.dの下に起動時に実行されるスクリプト（init.dへのリンク）ファイルが用意されており、起動時に自動的に起動される。

## Windowsの設定

まず**72ページ**の**画面12**で、「Microsoftネットワーククライアント」と「Microsoftネットワーク共有サービス」がバインドされていることを確認する。

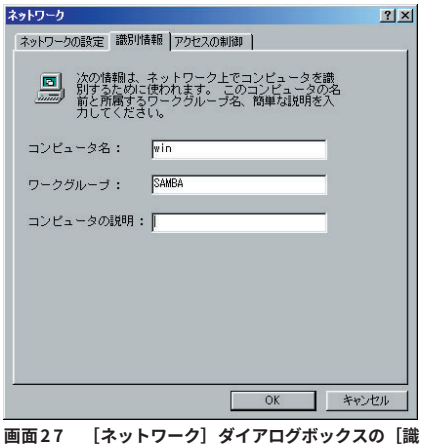
STARTメニューの［設定］－［コントロールパネル］を開き、［ネットワーク］アイコンをダブルクリックして、［ネットワーク］ダイアログボックスを表示する。［識別情報］タブで［コンピュータ名：］と［ワークグループ：］を設定する。コンピュータ名は任意。ワークグループは、Sambaの設定で指定したものと同じものを指定する（**画面27**）。

設定が終わったら、Windowsを再起動しよう。デスクトップにある［ネットワークコンピュータ］（Windows 9x）または［マイネットワーク］（Windows Me／2000）のアイコンからネットワークにアクセスする。Linuxマシン上の自分のホームディレクトリにアクセスできるはずだ（**画面28**、**29**）。

次にプリンタを設定しよう。STARTメニューの［設定］－［プリンタ］を開き、［プリンタの追加］アイコンをダブルクリックして、［プリンタの追加］ウィザードを起動する。プリンタの接続に「ネットワークプリンタ」を選択する。プリンタのネットワークパスで、Linuxマシンのプリンタ「lp」を指定する（**画面30**）。あとは、通常の［プリンタの追加］ウィザードと同じ手順だ。

参考までに、設定済みのプリンタの［EPSON MJ-930Cのプロパティ］の［詳細］タブを載せておく（**画面31**）。Windows用のドライバがそのまま利用できることがわかるだろう。

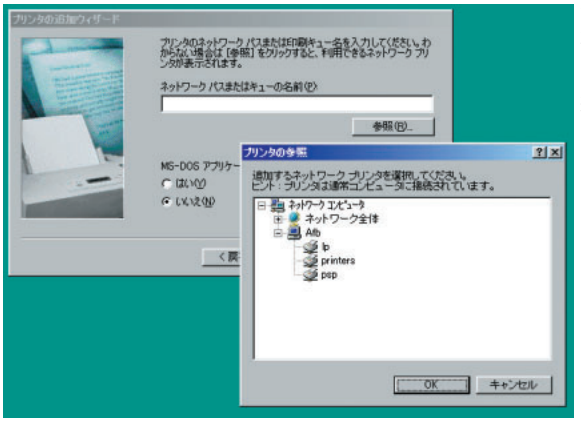
他人とデータのやりとりをすることを考慮すると、現状ではWindows環境を無視してしまうのは困難なことが多い。LinuxとWindowsを適材適所に利用することで、快適なLAN環境を実現してみよう。



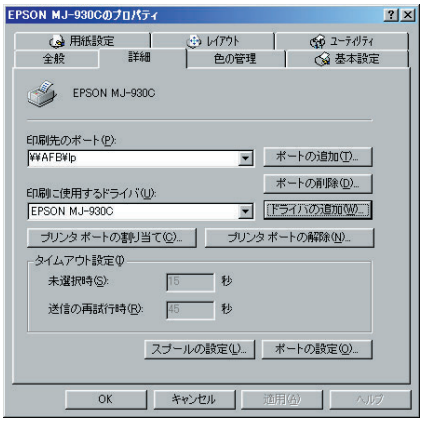

画面27　［ネットワーク］ダイアログボックスの［識別情報］タブ


画面28　［ネットワークコンピュータ］を開くと、同じワークグループのマシンが見える


画面29　Linuxマシンafbにアクセスすると、共有フォルダやプリンタが見える




画面30　［プリンタの追加ウィザード］でネットワークプリンタのパスを設定する。Linuxマシンのlpを選択する




画面31　［EPSON MJ-930Cのプロパティ］の［詳細］タブ

この際だから断言してしまおう。もはや「Linuxでゲーム？」などと疑問形で語る段階ではない。XFree86が3Dアクセラレーションを標準サポートし、Linuxでも高性能ビデオカードの性能を十分に発揮できる環境は整っている。Loki Entertainment Softwareの移植により、本格的なゲームタイトルも出揃った。機は熟したのだ。いまこそLinuxでゲームを！

# Linux ゲーム ユニックス

# Linuxでゲームしましょ

皆さんゲームしてますか？

本誌読者の大部分である正統派Linuxerの皆さんの中にも、相当なゲームマニアがいることと思う。そうでなくても、「家庭用ゲーム機は持っている」という方は多いはずだ。というワケで今回の特集はゲーム！　しかもLinux！　合言葉は「みんなゲームが好きなんだ」ということでよろしく。

## ゲームがLinuxにやってきた

まっとうな暮らしを送っている人々には、「ド○クエのせいで会社に遅刻した」とか「F○ Ⅷのデモムービーを会社のサーバにダウンロードして管理者に怒られた」といったゲーム好きの生態を理解できないかもしれない。

ただ、これくらいは序の口で「ゲームキャラクターのファンクラブを結成」したり、「日本に残っている『燃え○プ○野球』をすべて集めて」いたりするなど、意味不明なくらい「濃い」マニアが実在するのも事実だ。

こういった傾向は日本よりもアメリカのほうが顕著である。もともとがスケールの大きな国なので、マニアのスケールもデカいのだ（偏見？）。そしてアメリカには、PCゲームの文化がある。ゲームマニア達の存在とPCゲーム文化が背景にあったからこそゲームソフトをLinuxへ移植しようという動きが商業レベルで起こり得たのだろう。

その中心にいるのが、本特集で紹介するゲームの移植を手がけたLoki Entertainment Softwareだ。同社が移植・販売しているゲームは全12タイトル、開発中のものが6タイトルである（**表1**）。どのゲームもオリジナルが高く評価された名作ばかりだ。

Loki以外にも移植作業を進めているメーカーがいくつかある。

### Tribsoft

http://www.tribsoft.com/

SirtechのアクションRPG『Jagged Alliance 2』を移植。現在、Paradox Entertainmentの『Europa Universalis』、Cyberloreの『Majesty：The Fantasy Kingdom Sim』の2タイトルを開発中。

### Hyperion Entertainment

http://www.hyperion-software.com/

Monolith Productionsの『Shogo：Mobile Armor Division』、Ritual Entertainmentの『Sin』を移植。Shogoは2月11日にリリースされた。

### Vatical Vision

http://www.vvisions.com/

同社の『Terminus』は、1つのパッケージにWindows版／Machintosh版／Linux版が含まれている。市販ゲームソフトとしては、おそらく初めてのケース。

日本でも、ターボリナックスジャパンがLokiのゲームにTurbolinux Workstation 6.0 FTP版をバンドルしたパッケージ製品の販売を開始している。大手ディストリビューターの参入ということで、ゲームマニアはもちろん、これまでゲームに関心を持たなかったLinuxユーザーにも刺激を与えたはずだ。では、さっそく現在ターボリナックスジャパンよりリリース中の4タイトルを紹介していこう。

| 発売中のタイトル | オリジナル開発元 | ちょこっと解説 |
|---|---|---|
| Civilization：Call to Power | Activision | 文明を育てる「箱庭世界」シミュレーション |
| Myth II：Soulblighter | Bungie Software | ファンタジックな舞台で戦う戦略シミュレーション（詳細は88ページ） |
| Railroad Tycoon II | PopTop Software | 鉄道会社の会長となって会社経営を体験できるシミュレーション（詳細は90ページ） |
| Eric's Ultimate Solitaire | Delta Tao Software | 23種類のトランプゲームを集めた「究極のソリティア」 |
| Heretic II | Activision／Raven Software | Quake IIのレンダリングエンジンをベースにした3Dアクションアドベンチャー |
| Heroes of Might and Magic III | 3DO | RPGの名作『Might and Magic』のヒローが活躍するシミュレーションRPG（詳細は89ページ） |
| Quake III Arena | id Software | ネットワーク対戦が熱いバイオレント3Dアクション（詳細は87ページ） |
| Heavy Gear II | Activision | 巨大メカを操って戦うロボットもの3Dシミュレーション |
| SimCity 3000 Unlimited | Maxis | 家庭用ゲーム機でもおなじみの都市開発シミュレーションの最新版 |
| Soldier of Fortune | Activision／Raven Software | ミッションクリア型の3Dアクションガンシューティング |
| Descent3 | Interplay Entertainment | 美麗なグラフィック世界を360度視点で動きまわるスペースバトルアクション |
| Unreal Tournament | Epic Games | PlayStation2にも移植された大人気3Dバトルアクション |
| **開発中のタイトル** | | |
| Sid Meier's Alpha Centauri with Alien Crossfire expansion pack、MindRover、Kohan：Immortal Sovereigns、Tribes 2、Rune、Heavy Metal：F.A.K.K.2 | | |

表1　Loki Entertainment Softwareの移植ゲームタイトル

# Quake III Arena


**ターボリナックス直販価格：7900円**

SPEC 3D必須 OSS互換 20MB HDD 64MB RAM ネット対戦

アメコミチックなキャラクターがド派手なエフェクト（炸裂する重火器、弾け飛ぶ肉片！）にのって暴れまわる。「洋ゲー」ならではのバイオレンス感覚に溢れた3Dアクションゲーム。

本国アメリカでは、その暴力表現が世のお母様方を震撼させ（お子様たちは大喜び）、ゲームにおける暴力表現の是非をめぐって社会問題にまで発展したといういわくつきのゲーム。

## 3Dアクセラレーションが必須

オリジナルの開発元は、『Doom』シリーズでネットワーク対戦ゲームの先鞭をつけたメーカーとして知られるid Software。Quake III Arena（以下、Quake III）もDoomの流れを汲み、ネットワーク対戦（LAN、インターネット）が「売り」のひとつになっている。世界中に対戦サーバが設置され、日々熱き戦いが繰り広げられているのだ。

Quake IIIのもうひとつの見所は3Dグラフィックス。これはLinux版でも見事に再現されている。そのため、動作にはビデオカードによるハードウェア3Dアクセラレーションが必須となる。OpenGL互換のドライバがセットアップされていないと起動することもできないので注意したい。

製品パッケージには、3dfxのVoodooシリーズ、MatroxのG400／G200用のドライバが同梱されている。このほかのビデオカードを使う場合は、XFree86の設定が必要となる（92ページからの解説を参照のこと）。

## 「対戦」を重視

「はるか太古のこと、ミステリアスなアリーナマスター『Vadrigar』が……」云々といったサイドストーリーが一応あるものの、基本的には純粋なガンシューティングゲームである。そして、Quake IIIで最も重要視されているのが「対戦」という要素だ。

プレイヤーはグラディエイター（闘士）となり、究極の闘技場「Arena Eternal」において、シングルプレイヤーモードではCPUを、マルチプレイヤーモードでは対戦プレイヤーを相手に生き残りを賭けて戦う。敵を打ち破って勝つこと以外の要素は、見事なまでに切り捨てられているといっていい。ただし、「撃ちまっくて快感」というだけの単純なゲームではない。

最初のうちは、ただ撃ちまくっているだけでも気持ちいいが、戦いを有利に進めるためには、武器や防具、回復アイテムなどの出現場所（特定の場所で宙に浮かび上がる）を含めた闘技場のマップを記憶し、効果的に戦闘力アップと体力の回復を行っていく必要がある。また、マップ上の地形（扉や段差）を利用した待ち伏せなどの戦略を練ることも重要だ。

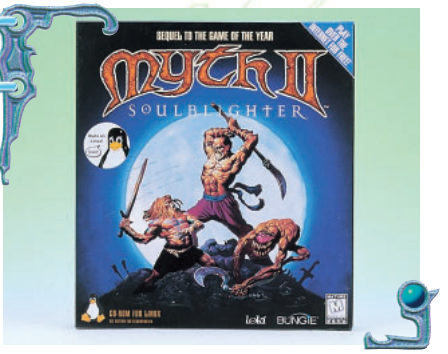
究極のシューティングテクニックと研ぎ澄まされた戦略を兼ね備えた最強のグラディエイターを目指せ！（ちょっと大げさ）。

画面1　ちょっとアレなキャラクター達
右端の目玉が強烈（ORBB君といいます）。このほかにも、とても個性的なキャラがてんこ盛りです。


画面2　これが問題の衝撃シーン
とても「ブラッディー」です。こんなゲームで遊んでいるなんて、お母さんはとても心配です。

●3D対応：OpenGL互換の3Dアクセラレーションがをサポート　●OSS互換：OSS互換のサウンドカードをサポート
●xxxMB HDD：インストールに必要な最小ハードディスク容量　●xxxMB RAM：動作に必要な最低メモリ容量
●ネット対戦：インターネット／LANでのマルチプレイヤーモードに対応

# Myth II Soulblighter




**ターボリナックス直販価格：4300円**

SPEC
3D対応
OSS互換
100MB HDD
32MB RAM
ネット対戦

Myth II Soulblighter（以下Myth II）は、神話の世界を舞台としたリアルタイムシミュレーションだ。優れたインターフェイスと高度な戦略性が高く評価された名作である。オリジナル開発元はBungie Software Products。

## 実績の3Dレンダリングを継承

メインとなる戦闘マップは3Dグラフィックで表示され、プレイヤーは3Dのシーンを捉えている「カメラ」を操作することで、自由に視点を切り替えられる。これは、刻々と状況が変化するリアルタイムシミュレーションにおいて非常に重要な機能で、カメラ操作インターフェイスとレンダリングエンジンの描画能力はゲームそのものの出来を左右する。Mythシリーズが好評価を受けたのも、それらが高いレベルの完成度を持っていたからだ。

Linux版でも、この点はしっかり引き継がれている。3Dハードウェアアクセラレーションにも対応しており、3dfxのVoodooシリーズがサポートされている。サポート外のビデオカードの場合はソフトウェア描画となるが、それでも、カメラのパン、ズーム、回転などは非常にスムーズだ。

## ゲームシステムとプレイモード

プレイヤーは3Dグラフィックで表示される戦闘マップ上で、兵士、魔法使い、ドワーフなどのキャラクターを操り勝利を目指す。都市や城砦の運営などは行わない、戦闘に特化した戦術級のシミュレーションである。

プレイヤーが置かれる状況はさまざまに変化する。敵部隊の出現・展開、天候や地形などを常に把握し、的確な指示をすばやく出さなくてはならないのだ。そのため、自分なりの必勝戦術を編み出すことと同時に、カメラ操作などのテクニックを極めることも重要な攻略要素となる。

ゲームモードは、シナリオに沿って各マップをクリアしていくシングルプレイヤーモードと、インターネットやLANを介したネットワーク対戦のためのマルチプレイヤーモードが用意されている。

シングルモードは、マップごとに決められた勝利条件を満たしてミッションをクリアして進んでいき、最終的にSoulblighterを倒すことが目的。

マルチモードでは、相手側の特定ユニットを暗殺する「Assassin」、相手ボールを奪い合う「Ball on Parade」といったプレイタイプを決めて対戦を行う。

インターネット上に開設されたフリーの対戦サーバ（http://www.bungie.net/）に登録して対戦者を探してもいいし、友達どうしでインターネットやLAN上で対戦することもできる。Windows版との対戦も可能だ。

一見単純そうに思えるゲームシステムなのだが、やってみると独特の奥深さがある。

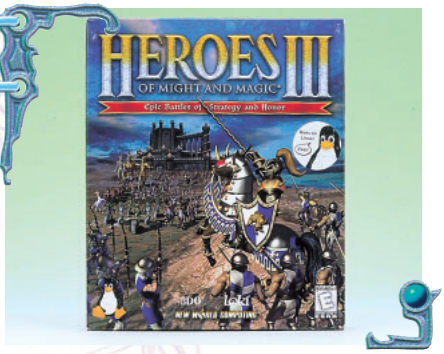
画面3　川面に写り込んだ風車
ソフトウェアレンダリングでも、かなりいい感じ。マニュアルによると現実世界の物理法則を忠実に再現しているのだとか。

画面4　戦場に立ち尽くす兵士達
戦闘後の一場面。このゲームもQuake IIIに負けないほど出血量が多い。アメリカ人っていったい……。

# Heros of Might and Magic III

**ターボリナックス直販価格：4600円**

SPEC
OSS互換
150MB HDD
32MB RAM
ネット対戦




RPGの古典的名作『Might and Magic』シリーズの世界観と登場人物に、戦略シミュレーション的な要素を合体させたシミュレーションRPG（SRPG）。剣と魔法の世界を鋭利な戦略で制覇することを目指す。

## シミュレーションとRPGの融合

RPGの戦闘システムの多くは、攻撃の方法と目標とする敵をコマンドで選択する方式をとっている。この方式では、戦闘シーンに変化が乏しく、繰り返しているうちに飽きてしまうことがある。この欠点を補うために、RPGの魅力を残しながら、戦闘システムに戦略シミュレーションの要素を盛り込んだのがSRPGである。Heros of Might and Magic Ⅲ（以下、Heros Ⅲ）はその代表的な作品だ。

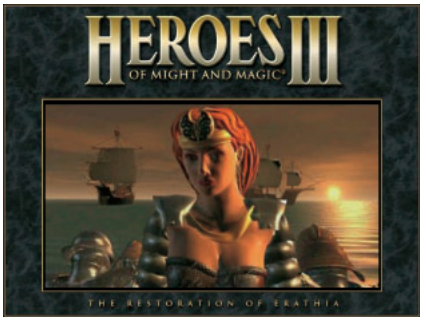
Heros Ⅲのシミュレーション的な要素は大きく2つに分けられる。「戦闘」と「都市の運営」だ。プレイヤーは戦場のヒーローとして軍隊を率いるだけでなく、占領した都市を開発し発展させなければならない。都市が発展することで収入が増え、軍隊の増強、ほかの都市の開発などが可能となる。

戦闘部隊は基本的に、隊長であるヒーローと隊員となるクリーチャー（モンスター）で編成される。ただし、ヒーローは必ずしも人間ではなく、クリーチャーの種族に属するヒーローもいる。また、敵方も人間とクリーチャーの混成部隊だ。このあたりに「人間対モンスター」といった単純な図式ではない、Might and Magicシリーズの世界観が表われている。

一方、RPG的な要素にあたるのが「キャラクターの成長」と「アイテム集め」。登場するヒーロー達は経験を積むことで成長し、魔力などの能力がアップしたり、特殊な技能を習得したりする。RPGではおなじみのシステムである。

魔法の剣、伝説の鎧といったアイテムもファンタジックなRPGの定番。Heros Ⅲにも、さまざまなアイテムが登場する。入手したアイテムは、両手、頭、首、胴など合計9カ所に装備できる（フルに装備すると、相当きらびやかになる。ちょっと派手すぎかも）。

どちらかというとRPG要素が強く、キャラクターに感情移入することでファンタジックな世界を体験できる。家庭用ゲーム機で『ドラゴンクエスト』シリーズなどをプレイして、この手の世界に親しんだことのあるRPGファンにお勧めだ。



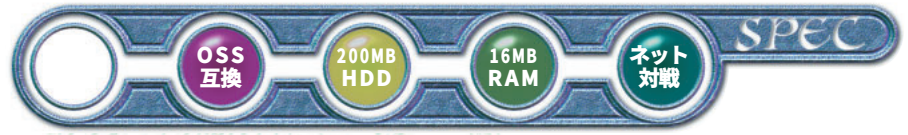

画面5　物語上の主人公キャサリン姫
舞台となるエラシア王国の王女にして、『Might and Magic 6』の主人公エンロス王ローランドの王妃。彼女には過酷な運命が待ち受けているのであった。

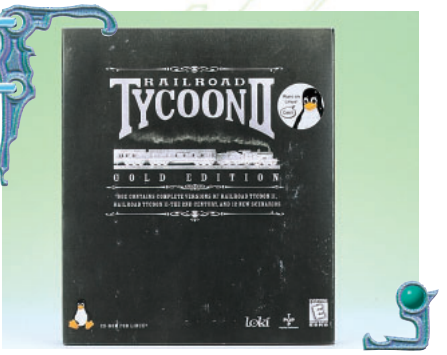

画面6　都市の運営を行う「Town Screen」
都市にはCastle、Dungeon、Fortress、Inferno、Necropolice、Rampart、Stronghold、Towerの8種類があり、それぞれに雇用できるクリーチャーの種類などの特徴がある（画面はCastle）。

画面7　戦闘シーンのひとコマ
部下が戦っているところを左上隅でこっそり見守っているのがヒーロー（やや卑怯）。部下を見捨てて退却することも可（とても卑怯）。

# Railroad Tycoon II Gold Edition

**ターボリナックス直販価格：4900円**




Railroad Tycoon IIは、鉄道会社を舞台にした企業経営シミュレーション。今回リリースされたGold Editionは、Railroad Tycoon IIの完全バージョンで、アドオンパック『The 2nd Century』に収録されていたシナリオのほかに、新しく12のシナリオが追加されている。オリジナルの開発元はPopTop Software。

## 戦略的な経営を重視

鉄道会社を経営するゲームといえば、日本ではアートディンクの『A列車でいこう』シリーズが有名だ。鉄道路線を開設し、会社の利益を追求するというゲーム内での目的は、両者に共通する点である。

しかし、2つのソフトにはこだわりの部分に違いがある。A列車が架空の箱庭世界で都市を発展させ、その中を運行する「マイトレイン」の姿を「愛でる」ことに強いこだわりを感じさせるのに対し、Railroad Tycoon IIは、効率的な人や物資の輸送を行うための経営戦略に重きを置いている。

それが最も顕著に表われているのがグラフィックスだ。A列車では都市の発展や、建物と列車の縮尺などが3Dグラフィックでリアルに再現される。Railroad Tycoon IIは、マップこそ実際の地形のとおり（すべて衛星写真を元に忠実に再現されている）だが、都市や列車はそれほどリアルにグラフィック表現されるわけではない（それなりに味わいのあるグラフィックではある）。ただ、地形のデータを基に傾斜による列車の加速度の違いなどは忠実に再現されている。このあたりの列車に対する「思い」では負けていない。

繰り返しになるが、Railroad Tycoon IIの力点はあくまで経営そのものにある。都市は居住区や工場、港、農場などの拠点で表現され、路線で結んだ各都市ごとの特性に合わせて物資（乗客も含む）を効率よく輸送しなければならない。そのために、機関車で牽引する客車や貨車の種類と数を決定していく。この決定がうまくいかないと、列車は走っているのに利益があがらないことになる。

この点は非常にシビアで、常に最善の手を打っていかないと成功はおぼつかない。正直、付属の英語マニュアルだけでは攻略は難しいだろう。試行錯誤を繰り返しながら、じっくり腰を落ち着けて取り組む覚悟のある方にお勧めしたい。


画面8　日本のマップもある！
東京湾に流れ込む隅田川（だと思う）の河口付近。このほかにも、アメリカ、イギリスをはじめ世界中のマップが用意されている。


画面9　列車の選択画面
国別、年代別に選べる車両の種類が変化する。日本でプレイすると「Shinkansen」も登場する。

Column

## Linuxのゲームって普通に売ってる？

今回のテストプレイ用に、紹介した4タイトルを編集部で購入した。秋葉原なら入手できるだろうと気軽に出かけたのだが、これが意外と見つからないのだ。せっかくの傑作ゲームも、読者のみなさんが入手できなければ意味がない。そこで、Linuxゲームの入手法について簡単に紹介しておこうと思う。

### ■小売店への入荷状況は？

初めにショップでの取り扱い状況から。秋葉原に到着して、まずはゲームソフト専門店に立ち寄ってみた。PCゲームのコーナーには、ずらりと新作が並んでいる（Quake IIIのアドオンパック『Team Arena』も好評発売中でした）。ひと通り探してみたが、Linux版らしきものは見つからない。結局、ソフトウェア専門店ばかり4軒まわって収穫ゼロ。

最初の店で店員さんに聞いたところ「えぇ～！Linux版ですかぁ？」ってな感じであった。まだまだLinux版についての認識度は低いようだ（残念である）。ここは自力で探すしかないと思い、Linuxに力を入れているPCショップへ……。ソフトコーナーに足を運ぶと、棚の真ん中にデーンと『Quake III』と『Myth II』のパッケージが置かれていた。しかも傍に置いてあるPCではMyth IIのデモをやっている。しかしパッケージは1つずつしかなく、売っているというより飾ってあるという感じだ。在庫の状況を聞いてみると、展示品のみとのことであった。

その後、PCショップのいくつかと、輸入ゲームの専門店なども探索してみたが、販売していたのは家電量販店系のPCショップのみ。ここには4タイトルともすべて置かれいていた（Quake IIIは平積み！）。エラいので名前を出して宣伝しておこう。LAOXのTHE COMPUTER館、通称『Theコン』だ！

以上のように、小売りの状況はあまり芳しくないようだ（今後に期待しよう）。現状では、ゲームソフト専門店よりもLinux関連のソフトウェアを取り扱っているPCショップのほうが置いてある確率が高い。その中でも特に、幅広い品揃えを心がけているであろう量販店がねらい目だ。

### ■オンラインショッピングがグッド

というわけで、日本のPCソフト販売のメッカである秋葉原でもこういった状況である。では、どうするか？　国を挙げてIT革命に取り組むご時世、ここはひとつオンラインショッピングでいこう。

#### ●ターボリナックス ジャパン

http://www.turbolinux.co.jp/

FTP版Turbolinuxのバンドルパッケージを販売しているターボリナックス ジャパンでは、Webサイトでの直販も行っている。送料を含めると今回購入した小売価格より若干高くなるが、確実に入手することができる。

#### ●Loki Entertainment Software

http://www.lokigames.com/

本家であるLokiのWebサイトでも直販を行っている。紹介した4タイトル以外に『Civilization： Call to Power』、『Unreal Tournament』など、同社が移植したLinux版の全タイトルを購入できる。価格は、Quake IIIが49.25ドル、Myth IIが29.95ドルと日本よりかなり安い。

海外のサイトでのオンラインショップに慣れている人や、どうしてもほしいタイトルがあるなら、ここがお勧め。

#### ●Amazon.com

http://www.amazon.com/

世界最大級のオンラインショップであるAmazon.comでも購入可能だ。Lokiのゲームタイトルのほぼすべてを扱っている。サイト内のサーチ機能を使って、「loki game」などのキーワードで検索すれば見つかるはずだ。価格はLokiの直販価格から1割引き程度。特集の初めに紹介した『Terminus』も販売も入手できる。掲載されている情報によると、ほとんどのゲームタイトルは24時間以内に出荷可能となっている。

オンラインショップとしては老舗で実績のあるサイトなので、一番のお勧めはここかもしれない。

#### ●シンアイ商会

http://www.shin-ai.com/

Loki製品の日本国内での販売代理店。実は、ターボリナックス ジャパンのバンドルパッケージ以外にも輸入・販売されているのだ。同社のサイトでは、国内で販売されているタイトルを確認できる。また、小売店などでの取り扱い状況の問い合わせも受け付けているので、「オンラインショップはどうも…」という人はメールで問い合わせてみよう。

現在は直販はしていないが、「今後、ぜひオンラインショップでの直販を行っていきたい」とのことだ。

Lokiのオンライン販売ページ

Amazon.comのゲームコーナー

シンアイ商会のホームページ

# ゲームを正しく楽しむための Linuxセットアップ講座

Linuxに限らず、Windowsでも、Macでも、PCでゲームを楽しむためには、それなりの出費とセットアップの手間が必要になる。しかし、ゲームをするためだけに、いきなりカリカリにチューンしたマニアックな構成のマシンを用意することはない。PCゲームの世界に魅力を感じたら、徐々に環境を整えていけばよいのだ。

とりあえずは、今使っているPCをちょっとアップグレードするだけで立派なゲームマシンになる。このコーナーでは、お勧めハードウェアを紹介しながら、そのセットアップ方法を解説し、ゲームをプレイするためのLinux環境について検討してみたい。

## ハードウェア構成を考える

パッケージやマニュアルの記述では、Pentium 133MHzに32Mバイト以上のメモリといったスペックが推奨されていることが多い。これはあくまで必要最小限の構成である。動かなくはないと思うが、ちょっときついだろう。

ゲームごとにハードウェア要件は異なるし、感じ方に個人差はあるだろうから、確実に「これ」という言い方はできないが、ストレスを感じることなくゲームを楽しむためには、CPUがPentium IIまたはK6-2の300MHzクラス、メモリは最低でも64Mバイトはほしい。モニタにも触れておくと、1024×768の解像度が表示可能なSVGAモニタであればまったく問題ない。

つまり、今どきのPCであれば能力的に何も問題はないということだ。実際に編集部でも、今回の特集に際して特別にハードウェアを用意することはしなかった（テストを行ったパーツは除く）。このコーナーのテスト、96ページ以降のレポートに使用したマシンのスペックを**表2**にまとめておく。

| | |
|---|---|
| CPU | AMD Duron 700MHz |
| マザーボード | MSI MS-6340 |
| チップセット | VIA KT133（VT8363／VT82C686A） |
| メモリ | 128MB SDRAM（PC133 CL3） |
| HDD | Maxtor DiamondMax Plus 45（20.4GB） |
| CD-ROM | KENWOOD UCR004010（40倍速） |
| モニタ | EIZO E57T |

**表2**　テストに使用したPCの主なスペック

## 3Dアクセラレーション

**表2**には、ビデオカードとサウンドカードが載っていない。この2つはゲームをするうえで重要なパーツなので個別に詳しく見ていくことにしよう。

PCのパーツの中で、ゲームと最もかかわりが深いのがビデオカードだ。特にQuake IIIのような3Dゲームを楽しむためには、水準以上の描画処理能力を持つビデオカードが必須となる。そして、そのカードに搭載されているグラフィックスチップが持つ「3Dグラフィックアクセラレーション」と呼ばれる機能を有効にしなければならない。

これは、3Dグラフィックのレンダリングに伴う複雑な演算を高速に処理するための機能で、この部分の性能の違いがグラフィックスチップの性能差となって表われる。グラフィックスチップというハードウェアに、直接グラフィックデータを渡して処理させることから、「ハードウェアアクセラレーション」または「ダイレクトレンダリング」と呼ばれることもある。

Linuxで3Dアクセラレーションを実現する場合には、以下のような組み合わせが選択肢となる。

- **XFree86 3.3.6＋Utah-GLX**
- **XFree86 4.0.x＋NVIDIAドライバ**
- **XFree86 4.0.x／DRIオプション**

それぞれサポートしているグラフィックスチップが異なるため、ハードウェアによって組み合わせが決まってく

写真1　3D BLASTER RIVA TNT2 Ultra
名前からもわかるとおり、RIVA TNT2を搭載。今回は主にQuak3 IIIのテストプレイに使用。Quake側の設定でテクスチャモードやライティング方式を調整することで、解像度800×600で30fpsを超える十分なパフォーマンスを発揮した。

る。コストパフォーマンス、入手のしやすさなどを考慮すると、お勧めなのがNVIDIAのRiva TNT2である。

TNT2は今や2世代以上前の製品だが、3Dゲームを楽しむのに十分な性能を持っていて、ローエンドモデルのTNT2 VANTAを搭載したカードであれば7000円程度の値段で購入できる。手軽にゲームを楽しむためには最適の選択なのだ。

もう少し絶対性能がほしいという方は、同じくNVIDIAのGeForce2 MXがお勧め。GeForce2シリーズのラインナップではローエンドモデルにあたるチップだが、非常にコストパフォーマンスに優れた製品だ。GeForce2 MX搭載カードは、ビデオRAMの種類や容量によって価格にバラツキがあるが、だいたい1万4000円前後で販売されている。

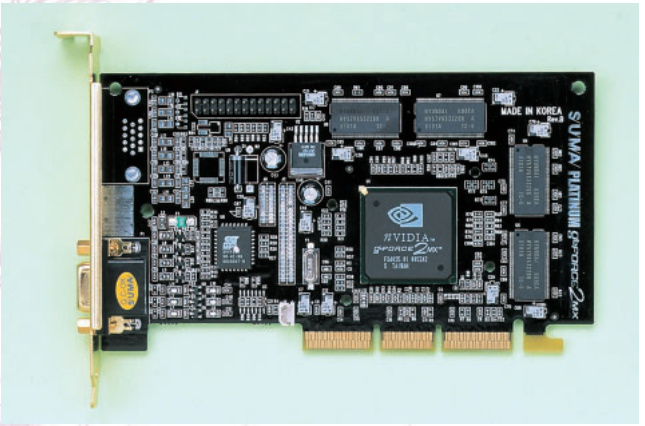



写真2　SUMA PLATINUM（GSV 32 SDR）
噂のカノ○プス似ではなく32MバイトのSDRAMを搭載したGeForce2 MX搭載の廉価モデル（購入価格は1万1980円）。それでも、Quake IIIの簡易ベンチマークで高い値をたたき出した。解像度1280×1024のフルスクリーンモードで45fpsオーバーという結果は価格を考えると驚異的。

TNT2、GeForce2 MXとも、XFree86＋NVIDIAが提供しているドライバで3Dアクセラレーションを有効化できる。設定は同じなので、どちらのチップでも以下で説明する手順でセットアップできるはずだ。

## NVIDIAチップ向けのセットアップ

NVIDIAのドライバは、カーネル2.2.12以上およびXFree86 4.0.1以上に対応している。そのため、XFree86を

## Column

### ATI RADEONのDRIサポート

RADEONは、ATIのハイエンド向けグラフィックスチップである。NVIDIAによる市場独占状態の中、GeForceシリーズに挑戦し得る唯一の対抗馬として登場した製品だ。

RADEONはXFree86 4.0.2でサポートされたが、XFree86の3Dアクセラレーション機能であるDRIについては未対応であった。RADEONは、Linuxにおいてその性能をフルに発揮することができないでいたのだ。

しかし今年になって、DRIの開発プロジェクトチームから、ドライバの開発はすでに完了しており、次期リリースのXFree86には、それが採用されるとのアナウンスがあった。そこでさっそく、プロジェクトのCVSリポジトリからソースファイルを入手、コンパイル＆テストを試みた（このとき入手したソースファイルを付録CD-ROMに収録してある）。

掲載スペースの都合上おおまかな流れだけを説明する。不明な点があれば、ソースアーカイブのxc/programs/Xserver/hw/xfree86/doc/README.DRIファイルを参照してほしい。なお、RADEONのDRIサポートには、カーネル2.4.xのAGPサポートが必要となる。

まず、適当な作業ディレクトリ（ここではDRI-CVSとする）を作って、そこにCVSを使ってソースをダウンロードする（CD-ROMに収録したアーカイブを使う場合は作業ディレクトリにファイルを展開する）。作業ディレクトリに移動して、さらにコンパイル用の作業ディレクトリを作成する。

```
# cd DRI-CVS
# ln -s xc XFree40
# mkdir build
# cd build
# lndir -silent -ignorelinks ../XFree40
```

続いて/usr/X11R6-DRI（ディレクトリ）を作成、DRI-CVS/build/xc/config/cf/Host.defに「#define ProjectRoot /usr/X11R6-DRI」を追加して、コンパイルを実行する。

```
# cd DRI-CVS/build/xc
# Make World >& World.LOG
# make install
```

エラーなく完了したら、/etc/ld.so.confの先頭に「/usr/X11R6-DRI/lib」を追加してldconfigコマンドを実行。コンパイルしたXFree86を起動するように、Xwrapperをリネームしてシンボリックリンクxを再作成する。

```
# cd /usr/X11R6/bin
# mv Xwrapper Xwrapper.OLD
# ln -sf /usr/X11R6-DRI/XFree86 X
```

XF86Configを作成し、Moduleセクションに「Load "glx"」と「Load "dri"」を追加、Deviceセクションのドライバの指定を「Driver "ati"」に変更する。

以上でセットアップは完了だ。ただし、Xを起動する前にカーネルのDRMモジュールradeon.oをロードしておく必要がある。

DDR32Mバイト搭載機種を使った今回テストでは、GeForce2 MX（本文中で紹介したSUMA PLATINUM）とほぼ同程度の結果にとどまった。しかし、XFree86 4.0.3（か？）での正式サポートまでにドライバのチューンアップが進めば、雑誌などで公開されているWindowsでのベンチマーク結果から見ても、GeForce2 GTSさえも凌ぐ可能性を秘めているはずだ。期待したい。

バージョンアップする必要のあるケースが多いだろう。

XFree86 Projectが配布しているx86プラットフォームのglibc 2.1.x環境用にコンパイルされたバイナリ版を付録CD-ROMに収録してあるので、これを利用してほしい。あわせて、XFree86 Projectが配布しているInstallファイルも収録してある。このファイルには、新規インストールとバージョンアップの手順が詳しく記載されている。作業を行う前によく読んで、内容を理解しておこう。慎重に作業すれば、失敗することはないはずだ。

XFree86の準備が整ったら、次にNVIDIAのドライバをインストールする。ドライバパッケージは、カーネルモジュール（NVdriver.o）、XFree86のドライバモジュール（nvidia_drv.o）、そして3Dアクセラレーション用のOpenGL／GLXドライバ（LibGLCore、Libglxなど）から構成される。配布は同社のWebサイト（http://www.nvidia.com/）で行われており、カーネルモジュールのソースRPMパッケージ「NVIDIA_kernel-0.9-6.src.rpm」、XFree86とOpenGLのドライバのバイナリRPMパッケージ「NVIDIA_GLX-0.9-6.i386.rpm」の2つのファイルを入手する必要がある。RPMパッケージなのでインストールはとても簡単。

▼リスト1　XF86Configの設定例

```
Section "Module"
    Load   "dbe"
         :
    Load   "type1"
    Load   "xtt"
    Load   "glx"
EndSection

Section "Device"
    Identifier  "Riva TNT2"
    Driver      "nvidia"
EndSection
```

```
# rpm --rebuild NVIDIA_kernel-0.9-6.src.rpm
# rpm -Uvh NVIDIA_GLX-0.9-6.i386.rpm
```

このコマンドで、モジュールとライブラリのコピー、シンボリックリンクの作成や/etc/modules.confの変更など、ドライバまわりの設定がすべて完了する。OpenGL関連の共有ファイル（拡張子.soのファイル）が置き換えられるので、念のためldconfigコマンドを実行しておこう。

XFree86の設定は、パッケージに含まれている「xf86config」で行う。テキストベースのツールだが、XFree86の設定を理解していれば、それほど迷うことはないはずだ。カードの選択画面では、TNT2の場合はTNTを搭載しているカードのいずれか、GeForce2 MXの場合は「NVIDIA GeForce」をとりあえず選択する。

xf86configが生成する/etc/XF86Configのままではドライバが有効にならない。**リスト1**を参考にして修正しよう。必要なのは、Moduleセクションの「Load "glx"」とDeviceセクションの「Driver "nvidia"」の2行。これが、新しくインストールしたドライバを有効にするための設定だ。

▼リスト2　glxinfoの実行結果の例

```
display: :0.0  screen:0
direct rendering: Yes
server glx vendor string: SGI
server glx version string: 1.2
server glx extensions:
    GLX_EXT_visual_info, GLX_EXT_visual_rating, GLX_EXT_import_context
client glx vendor string: SGI
client glx version string: 1.2
client glx extensions:
    GLX_EXT_visual_info, GLX_EXT_visual_rating, GLX_EXT_import_context
GLX extensions:
    GLX_EXT_visual_info, GLX_EXT_visual_rating, GLX_EXT_import_context
OpenGL vendor string: VA Linux Systems, Inc.
OpenGL renderer string: Mesa DRI Radeon 20010105 AGP 2x x86/3DNow!
OpenGL version string: 1.2 Mesa 3.4.1
```

Xが起動するようになったら、3Dアクセラレーションが機能しているかどうかを確認しよう。XFree86 4.0.2には、glxの動作状況を報告するユーティリティglxinfoが付属している。Xが起動した状態で、X端末からパラメータなしで実行すると、glxおよびOpenGLのドライバ情報を表示してくれる。**リスト2**はglxinfoの出力をリダイレクトしたファイルの一部だ。2行めに「Direct Rendering：Yes」とあることから、3Dアクセラレーションが有効であることがわかる。

## サウンド環境の整備

Linuxでは現在のところ、WindowsにおけるA3DやDirectSound3Dのような3Dオーディオ環境を実現するための規格は確立されていない（Lokiなどが中心となって「OpenAL」とよばれる標準規格を策定中）。PCゲームでは、サウンドも重要な要素のひとつであるだけに残念なところだ。

3Dサウンドなどの特殊な音響は期待できないにしても、テーマ曲や効果音といったゲームが持つ音の世界は再生して楽しむようにしたい。現在使っているPCにサウンド機能がない場合は、

とりあえず低価格のサウンドカードでかまわないので、Linuxで動作実績のあるものを取り付けよう。

サウンドカードのLinux用ドライバは、OSS／FreeとALSA（Advanced Linux Sound Architecture）で提供されるものがメジャーだ。OSS（Open Sound System）は、4Front Tecnologiesが推進するUNIX系OS向けのサウンド規格。Lokiのゲームタイトルでは OSS互換のサウンドシステムが必要となるが、ALSAもOSSとの互換性があるので、OSS/FreeまたはALSAでサポートされているサウンドカードを用意すれば、ゲームサウンドを再生できるようになる。

たとえば、定番であるCreative TechnologyのSound BLASTERシリーズは、OSS/Free、ALSAの両方でサポートされている。Sound BLASTER Live! VALUEのOEM版なら、7000円前後だ。

より安いものでは、YAMAHAのYMF744／754、C-Media ElectronicsのCMI8338／8738、ESS TechnologyのSolo-1／Maestro-1などを搭載した製品がある。これらは価格も3000円前後で入手しやすくお勧めである。これらのサウンドチップはALSAでサポートされている。

## ALSAのセットアップ

ALSAのセットアップ方法を説明しておこう。付録CD-ROMには、ALSAの最新リリース0.5.10bのソースアーカイブを収録してある。まずは、この3つのファイル（alsa-driver-0.5.10b.tar.bz2、alsa-lib-0.5.10.tar.bz2、alsa-util-0.5.10b.tar.bz2）をコンパイル用の作業ディレクトリに展開する。

```
# bzip2 -dc <ファイル名> | tar xvf -
```

ソースファイルがそれぞれのサブディレクトリに展開されるので、そのサブディレクトリに移動してコンパイルではおなじみのアノ呪文を唱えれば、とりあえずインストールは完了だ。

```
# cd <サブディレクトリ名>
# ./configure
# make
# make install
```

このあとのセットアップは、サウンドカードごとに異なる。ALSAのドライバモジュールがブート時にロードされるよう、/etc/modules.conf（またはconf.modules）に**リスト3**の内容を追加しよう。**リスト3**は、テストプレイで使用したLABWAYのXwave-6000の例である。このカードに搭載されているサウンドチップはYMF754だ。

YMF754以外のチップを搭載したサウンドカードの場合、「alias snd-card-0 snd-card-ymfpci」の行をそれぞれのドライバ用に変更すればよい（YMF744はYMF754と共通）。CMI8338／8738は「snd-card-ymfpci」を「snd-card-cmipci」に、Solo-1は「snd-card-es1938」、Maestro-1は「snd-card-es1968」とする。

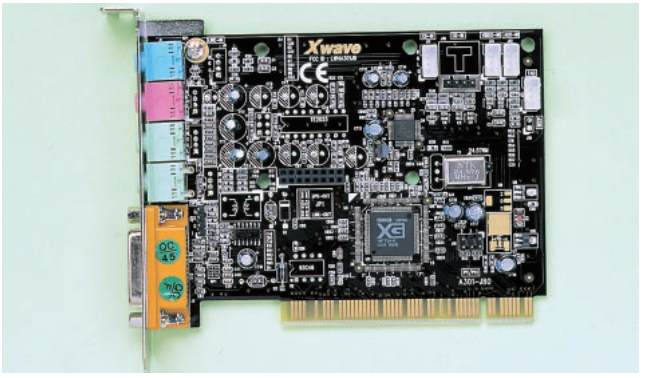


写真2　LABWAY Xwave-6000
安価なサウンドカードの代表格。同じ製品名で2つの異なるモデルがあり、YMF744／YMF754のいずれかが搭載されている。しかもパッケージからは、どちらかわからない。ただし、2つのチップはほとんど同じものという噂があることも確か。

## セットアップの効果をテストしよう

さて、ゲーム用のセットアップが完了したところで、その効果を測ってみよう。といっても、サウンドカードの効果は音が出るだけなので、xplaycdやXMMSを使ってファイルとCD-ROMドライブからきちんと再生できればOKだ。

ビデオカードのほうは、先ほど紹介したglxinfoの情報から3Dアクセラレーションが有効なことはわかっても、効果までは測れない。Linuxには専用の3Dベンチマークツールがないので、glclockやQuake Ⅲのベンチマーク機能を使って描画能力を計測しよう。

glclockのベンチマーク結果に関しては、配布元のWebサイト（http://www.daionet.gr.jp/~masa/glclock/）にいろいろなカードで計測した結果があるので、自分のマシンの実力を確認してみよう。Quake Ⅲのベンチマークモードについては、99ページのコラムで取り上げている。

▼リスト3　conf.modulesの設定例

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

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

# Play Q3A or Die!!

## ～Quake III Arena激闘録～

「Quake III Arena」、それは初代ファミスタで目覚め、ドラクエVIで燃え尽きた私のゲーム魂に再び火をつけた作品の名だ。

### とにかく熱いぜ！

冒頭から、ややクサいくらいに熱くなって申し訳ないが、それくらい熱いゲームなんである。

まずキャラが熱く濃い、それはもうアツ苦しいくらいだ。さらに舞台が熱い。グラディエーター達の闘いの場となる「Arena Eternal」は、高度な3Dグラフィックで生み出された完全なる闘技場なのだ。数々のギミックにあふれ、その中で闘士たちが、走り回り、高く跳ね、吼え、闘う。そして、もちろん闘いが熱い。特に攻撃力を飛躍的にアップさせるアイテム「Quad Damage」を取ってプラズマガンをぶっぱなすときの気分は格別である。

熱い、熱いと言ってばかりではしょうがないのでレポートに移ろう。ちなみに、ヘタッぴではかっこ悪いのでネットワーク対戦は行っていない。これは、そんなふうに見栄っ張りで、原稿を忘れてQuake IIIにハマッた男の記録である。

### INTRODUCTION

始まりのステージは「Q3DM0」。登場する敵キャラは「Crash」だ。マニュアルによると、彼女（女性です）はArena Eternalに初めてやってくる戦士の腕を試す試験官であるそうだ。ごくろう様です。

このステージはチュートリアル的なアリーナなので、開始地点の部屋には敵の影はない。試しに、その部屋から出ずにマシンガンを撃ったりしてみたが、Crash教官が侵入してくることはなかった。ジャンプや武器を切り替えるキー操作を確認してから、意を決してゲートをくぐる……。

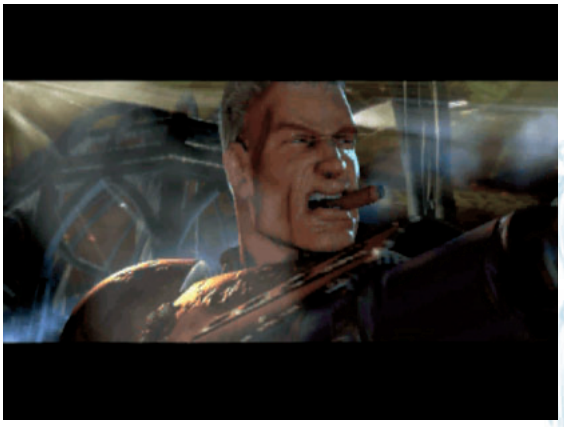
死んだ、すぐ死んだぜ。デフォルトのマイキャラである「Sarge」は、葉巻をくわえた、いかにもアメリカンなマッチョのくせに「ううっ」とかいってテンで弱いでやんの！（自分がヘタなのはわかってます。責めないでください）。でも、Arena Eternalでは死んでも、すぐに生き返るので安心。心置きなく闘えるのだ。

でもでも、やっぱりすぐヤられた（「死んだ」は物騒なので変更）。そのあと何度挑んでもCrash教官にどうしても勝てない。自分にはアリーナで闘う資格さえないように思えて落ち込んできたため、気分転換にSargeをクビにしてマイキャラを変更することに。イカれた造形が素敵なKleskを選択する。

なおもCrash教官にしごかれているうちにメキメキ腕を上げた私は、ついに教官を打ち倒した。ちなみにシングルプレイヤーモードでは、決められたフラッグ数を先に獲得したキャラの勝ちとなる。フラッグは相手を倒すことで獲得できる。Q3DM0のフラッグ数は5、つまりは師匠であるCrash教官を5回も倒した（ストレートに言うと○した）ことになる。身をもって闘いのなんたるかを教えてくれたCrash（すでに呼び捨て）、君のことは忘れないよ！

### TIER 1－試練

Arena Eternalは全部で7つの層



画面10　デフォルトのマイキャラ「Sarge」
Sargeはオープニングムービーにも華々しく登場する。戦闘のプロらしいが、私は嫌いだ。

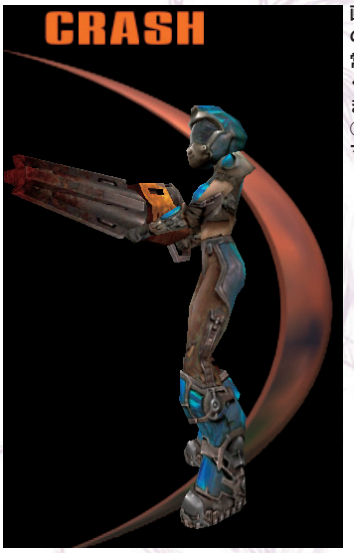

画面11　キャラクター選択画面
各キャラクターごとに基本キャラとその色違いバージョンが数種類ずつ用意されている。

(TIER)に分かれていて、各層にいくつかのアリーナ(マップ)がある。その層のすべてのアリーナで勝利すれば次の層に進むことができ、最終のTIER 7をクリアするとエンディングとなる(シングルプレイヤーモードの場合)。

TIER 1からが本当の闘いで、敵キャラの攻撃も本格的になってくる。アリーナも少し広くなり、立体的な構造が登場する。それほど複雑ではないものの、マップを記憶して効率よく闘うことが必要になる。ということに気づいたのは、ヤられ続けて悲しい気分に浸っているときだった。

とにかくヤられまくったのが、2つめのアリーナ「Q3DM2」の敵キャラ「Phobos」。とにかく動きがはやくて(これは「Speed」というアイテムを取った結果であることがのちほど判明)、なにやら得体の知れない奇妙な闘い方をしてくる。そのうえ勝利後の表彰台の画面でのパフォーマンスがすごく変なのだ。負けたあとだけに「マジでむかつく」感じなのである。

3つめの「Q3DM3」では、初めて敵チャラが「2体」登場する。2人ではなく2体としたのは、敵キャラの一方が87ページで紹介した目玉野郎だからだ。この「Orbb」もかなりの強敵で、甲高い悲鳴のような声を発しながら(どっから出してんだ!)、びょ〜ん、びょ〜んと跳ねまわって攻撃してくる。動きが変則的なのと的がやや小さいのが厄介だ。




画面12　わが心の師「Crash」常に厳しく鍛えてくれたご恩は忘れません。ホントに、○しちゃってスイマセン。


画面13　不倶戴天の敵「Phobos」勝ってうれしいからといって浮かれるのは武士道にもとる行為だよ!キミ(武士ではありません)。

新米グラディエーターの身でありながら生意気にも攻略めいた話をさせていただくと、まずは「Heavy Armor」などで防備を固めてから攻撃に転じたほうがよい。それから、やたらと走りまわらないこと。ふいに現われた敵に対処できないからだ。このゲームでは、敵の動きを予測して先手を打たないと圧倒的に不利になる(圧倒的にヤられた私が言うのだから間違いはない……はず)。

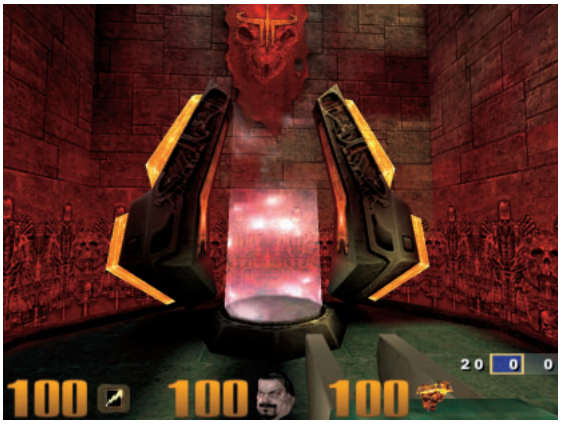
## TIER 2ー激闘

仕事も忘れて打ち込んだ結果、何とかTIER 1を攻略し、やっとの思いでTIER 2に到達することできた。やはり、何ごとも精進が肝要である。

TIER 2ではアリーナのマップがさらに複雑化し、異常に高くジャンプできる「Bounus Pad」や別の地点にワープできる「Teleporter」などの特殊なマップ要素が加わってくる。Bounus Padは、利用するとグラフィック的に楽しめる。つい無意味に何度も使ってしまった。もちろん本来は、敵の攻撃をかわしたり、これらを使わないと行きつけない場所に移動するという目的のためにあるのだ。

また、冒頭で紹介した禁断のアイテム「Quad Damage」もこのTIERから登場する。このアイテムはとにかく強力。自分が取れたときは快感が背筋


画面14　闘いは常に戦略的に
画面のHeavy Armorなどのアイテムを確実に取ってから、攻撃に転じるのが吉。防御力を上げておけば、大胆な攻撃も可能になるのだ。




画面15　TIER 2から登場する「Teleporter」光の中に飛び込むと、決められた場所にワープできる。まわりでウロウロしていると敵が出現してくることもある。


画面16　驚異の跳躍を可能にする「Bounus Pad」
マニュアルによると重力をどうにかすることで機能しているらしいが、詳細は不明。


画面17　跳躍後に見下ろした下界
うまく着地しないと、落っこちてもう一度ジャンプする羽目になることも。

画面18　これが魅惑の「Quad Damage」
出現したら、すぐさま獲得すること。ただし防御力には影響しないので、強気になりすぎないように。

を走るほどに敵を倒しまくれるが、逆に敵に取られるとこれほど恐ろしいものはない。だからといって、こだわりすぎると自分のリズムを壊してしまうので注意が必要。また、敵に取られた場合は、すぐに身を隠してロケットランチャーやレールガンで遠くからコッソリ狙い撃ちしよう。

TIER 2で最も激戦となるアリーナが「Q3DM6」だ。敵キャラ5体（またもやOrbbが登場する）とマイキャラの計6体で壮絶なバトルが繰り広げられる。激闘の結果、闘いのあとに「Excellent」や「Impressive」といったメダルがもらえたときは（歳を考えると恥ずかしいのだが）素直にうれしかった。なお、Exellentは2秒以内に2つのフラッグを取る（つまり2秒以内に2回刑法199条に違反する）、Impressiveはレールガンで2回連続して敵に命中させることで獲得できる。

TIER 2の初登場キャラは合計5人（一応人間に見える）。この中で強烈な個性を放っているのが「Hossman」である。外見はやけに顔色の悪いデブな奴というだけで、どうということもないのだが、ちょっと言葉では表現できない奇妙な声で「アーハ、ヤー、アーハ」などと言いながら、のたのたと近づいてくる。実際に聞いてみないと、おそらくわからいないと思うが、顔色の悪さもあって、かなり不気味である。ゲームのキャラにこんなことを言うのもアレだが、きっとホ○に違いない。

## そしてTIER 3へ

Q3DM6での激闘を制し、TIER 2の最後のアリーナ「Q3TOURNEY2」（ここでの敵キャラ「Hunter」もかなりの難敵だった）を切り抜ける頃になると、自分の戦闘テクニックが格段にアップしたことを実感できる。

Quake IIIの良さのひとつは、「やり


画面19　TIER 2のアリーナ選択画面
Q3DM6は決戦の地であると同時に、勲章（メダル）を手にするチャンスでもある。画面下段に並んでいるのが自慢のマイメダル。


画面20　変態と推測される「Hossman」
担当にあらぬ疑いをかけられた悲しき男。しかしなんせ「ア〜ハ、ヤ〜、ア〜ハ」だしな。


画面21　鏡に映ったマイキャラ
手前の光の具合、手元の銃や床の質感もなかなかいい感じだ。もっとも、戦闘中に見る余裕はないのであった。


画面22　水底から見上げた外界
水の中では、動きが重くなるのもリアルに再現されている（潜りすぎると、もちろん酸欠になります）。

込むごとに自然にうまくなっていく」ことにある。あたりまえのようだが、バランスの悪いゲームではそうはならない。各ステージの難易度や戦闘シーンでのCPUキャラのスマートな闘い（人間では不可能な攻撃とか動き方はしないし、間抜けでもない。でも、たまに強すぎて腹が立つこともある）がプレイヤーを自然に育てるのだ。

実際にプレイして、すごいと感じさせたのがグラフィックス。正直なところ、Linuxでここまで表現できるとは思わなかった。このグラフィックスを見るだけでも、絶対に価値がある（断言）。さらには、そのグラフィック空間の中を自由に動きまわれるのだ。

Quake IIIは面白い。さっきは冗談めかして書いたが、本当に一瞬（だけです＞編集長）仕事を忘れてしまったほどに。というわけで強力にプッシュするしだいである。さて、原稿も書いたし、さっそく『Team Arena』を買いに行ってこよう〜〜っと。

## Column

### Quake IIIによる3Dベンチマーク

本誌ではビデオカードの3Dグラフィックス処理性能を計測するために、これまでに何度かQuake IIIのベンチマーク機能を使ってきた。あまり詳しく取り上げたことがなかったので、これを機会に少し詳しく解説しておこう。

■timedemoコマンド

Quake IIIに限らず、ゲームプログラムでは実行時の処理スピードが一定の速さ以上にならないよう調整を行っている。ハードウェア性能の限界までゲームのスピードを上げてしまうと、とても人間では反応できない速さになってしまう可能性があるからだ。

Quake IIIのtimedemoコマンドは、このゲームスピードの調整機能をオン／オフするためのものだ。チルダ（‘~’）キーを押してコンソールを呼び出し、「\timedemo 1」と入力して実行すると、調整機能がオフになる。この状態でデモを実行すると、まさに「タガのはずれた」状態でデモが再生される。デモの終了後にもう一度コンソールに切り替えると、デモの再生でレンダリングしたフレーム数、再生にかかった秒数、そしてフレームレートが表示される。フレームレートは、1秒間に何フレーム描画したかで表される（フレーム／秒＝fps）。フレームレートの数字が大きいほど、そのPCの3Dレンダリング能力が高いということになる。

コンソールはネットワーク対戦のチャットにも利用されている。シングルプレイヤーモードでは関係ないが、マルチプレイヤーモードでコマンドの前のバックスラッシュを付け忘れると少し恥ずかしい思いをする。入力した文字列がチャットの会話と認識され、対戦に参加しているほかのプレイヤーに「timedemo 0」などと語りかけてしまうことになるのだ。

■そのほかのコンソールコマンド

実はフレームレートを表示する方法がもうひとつある。コンソールで「\cg_drawFPS 1」を実行すると、画面の右上隅にフレームレートが表示されるようになるのだ。

もうひとつ紹介しておこう。ベンチマークとは関係なくて、あまり役にもたたないが、知っていると少し楽しいコンソールコマンドである。

通常の状態では一人称視点なので、自分のキャラクターは倒されたときくらいしか画面に登場しない。「\cg_thirdPerson 1」を実行すると、第三者の視点に切り替わり、マイキャラが画面に現われる。ただし照準の感覚がまったくわからなくなるので、すぐにヤられてしまうのがすごく悲しい。


このようにフレームレートが表示されるようになる

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

## 第1回　キャプチャカードを設定する

パソコンショップをのぞいてみると、デスクトップ型のWindows搭載PCの多くが「テレビが観られて、録画もできる」というのを売り文句にしている。一方で、パーツ売り場を見ると、テレビチューナー内蔵キャプチャカードもそこそこの価格で入手できるようになっている。ほとんどの製品は「Windows対応」としか記載されていないのだが、これらの製品の中にはLinuxでも利用できるものがあるのだ。これを使って、Linuxマシンでテレビを視聴できるようにしてみよう。さらに次回以降では、Windowsマシンでは難しい、バッチ処理による自動録画や、電子メールを使ったリモート録画予約にも挑戦してみたい。

文：竹田善太郎
*Text : Zentaro Takeda*


### テレビ録画もLinuxで

本誌2000年10月号～11月号に掲載された記事「ラジオ自動録音システム」への反響として、何人かの読者の方々から「テレビ録画にも挑戦してほしい」という要望をいただいた。筆者はもともと、あまりテレビが好きではないのだが、たとえば「語学をマスターしたい」という目的なら、テレビの語学番組をLinuxで自動録画できるようになれば、かなり便利なことは確かだ。

パソコンとテレビの「融合」は、メーカー製の機能てんこ盛りデスクトップパソコンを例にあげるまでもなく、一般ユーザーにとってもあたりまえのことになりつつあるようだ。現実には、テレビ放送の形式がアナログからデジタルに移行する時期にもさしかかっているわけで、今あわててパソコンでテレビを観られるようにする必要があるかどうかは疑問だが、現状のパソコンでもテレビのリアルタイム視聴や、そこそこの画質での録画も可能なのだから、Linux上でどこまでできるのか試してみるのも、悪いことではなさそうだ。

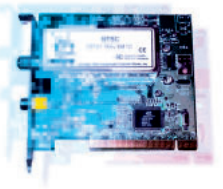
### TVキャプチャカードの現状

PCIバスに接続するTVチューナー内蔵ビデオキャプチャカード（以下、TVキャプチャカードと略す）は、安いものなら3000円前後から入手できるようになった。後述するように、あまりにも安いカードの中には、品質上の問題があるものや、Linuxでの使用が難しいものもあるのでお勧めできないが、おおむね、7000～1万円程度も出せば、日本国内の代理店が販売する、品質の安定したTVキャプチャカードを購入できる。このような製品のいくつかを実際に購入して、Linuxで利用できるかどうかを試してみた。その結果については後述するとして、まず、TVキャプチャカードとはどのようなものなのか、その基礎的な知識についてまとめてみよう。

#### ●キャプチャカードの種類

現在入手できるTVキャプチャカードには、大きく分けて2種類ある。

ハードウェアによるビデオデータの圧縮機能（MPEG-1や



MPEG-2形式）を持つもの、そして圧縮機能を持たないものの2種類である。

ハードウェア圧縮機能を持つカードは、リアルタイムで映像を圧縮しながらキャプチャできるので、CPUへの負担をかけずに長時間の番組を録画できるのだが、残念ながら、Linuxでのサポートは限られている。ごく一部の製品については、Linux用のドライバが開発され、シェアウェアの形式で公開されているようだが、有料のドライバというのは、利用に抵抗があるし、対応するTVキャプチャカードの機種がきわめて限定的なので、今回は取り上げない。対応するカードの数が増え、ドライバが利用しやすくなった時点で、改めて取り上げたいと思う。

ハードウェア圧縮機能を持たないTVキャプチャカードでは、圧縮されていないフォーマット（RGB、YUV形式など）、あるいは簡単な圧縮フォーマット（モーションJPEGなど）による動画キャプチャしかできない。しかしその分、圧縮機能つきのカードに比べて安価で、国内代理店が販売するものであっても、1万円以下で購入できる。また、キャプチャ済みのデータを別のソフトウェアを使って圧縮することは可能なので、Linuxの得意とする「バッチ処理」で圧縮保存できる。MPEG-1形式で圧縮すれば、画質にもよるが、たとえば1時間のテレビ番組なら、数十Mバイトから数百Mバイトのファイルに収めることができるはずだ。

以上のような事情を考慮して、本記事では圧縮機能のないTVキャプチャカードを利用することにした。

### ●キャプチャカードの中身

TVキャプチャカードの中身は、製品によって細部は異なるものの、テレビ放送の電波を受信してアナログ形式の映像信号と音声信号を得る「チューナー」部分と、チューナーから受け取ったアナログの映像信号をデジタルデータに変換し、PCIバスを経由してグラフィックスカードなどに送る「フレームグラバー」（frame grabber）と呼ばれるチップの2つの主要部品から構成されている（**図1**）。

このうち、もっとも重要な働きをしているフレームグラバーチップに関しては、現在市場に出回っているほぼすべてのTVキャプチャカードで、「Bt8x8」シリーズのチップが使われている（**写真1**）。このチップは、Brooktree社が開発したフレームグラバーチップで、「848」、「878」、「879」などの種類があるが、いずれもソフトウェア的には互換性がある。なお、Brooktree社はその後Coexant社に買収されたため、Bt8x8シリーズはCoexant社から発売されるようになり、製品名も「Fusion 8x8」シリーズと変わっているが、中身は以前のBt8x8シリーズそのものである。

もう一方の主要コンポーネントであるTVチューナーは、Philips、ALPS、Temicの3社の製品が使われていることがほとんどである。海外製のTVキャプチャカードでは、現在はほとんどがPhilips製チューナーユニットを使用しているようだ。

このチューナーユニットは、同じメーカー製でも複数の種類があり、テレビだけでなくFMラジオのチューナーを内蔵

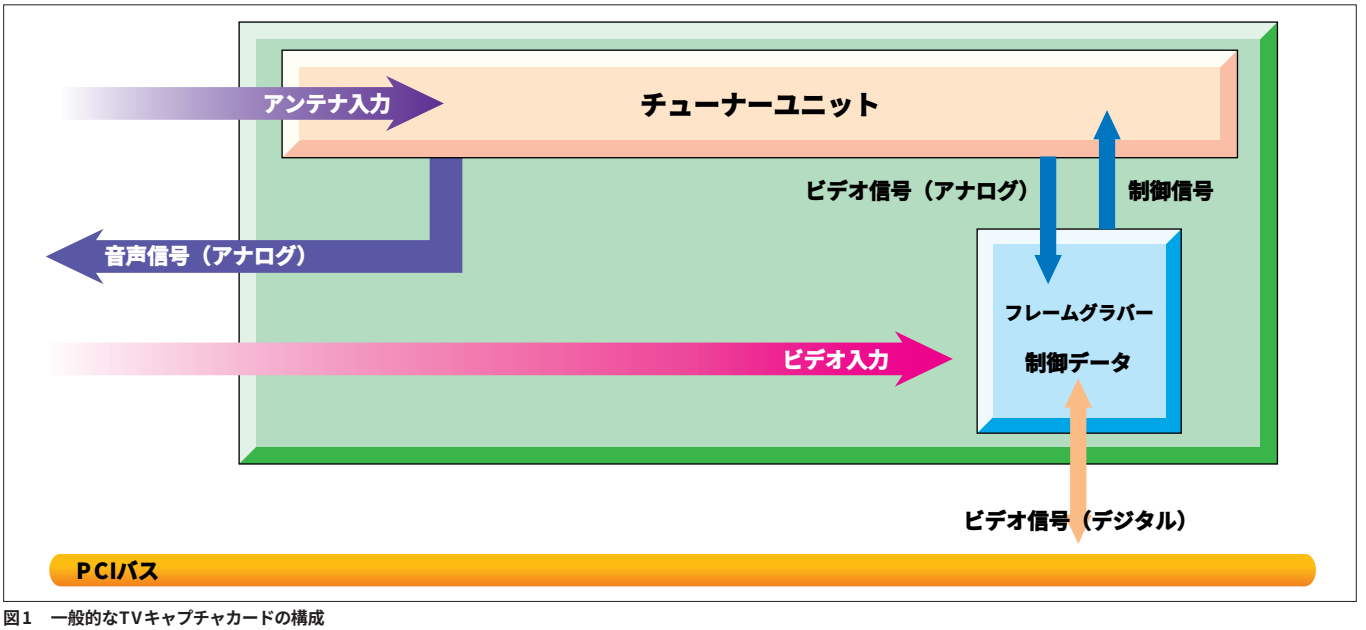



図1　一般的なTVキャプチャカードの構成
TVキャプチャカードの主要部品は、チューナーユニットとフレームグラバーチップの2つである。チューナーユニットで受信したアナログ画像信号はフレームグラバーでデジタル変換され、PCIバスに送られる。

したものや、ケーブルテレビ、デジタル衛星放送（いわゆる「デジタルCS放送」）などに対応したものなどさまざまである。また、テレビ放送の信号形式についても、日本や米国などのNTSC方式、主に西欧で使われているPAL形式、東欧で使われているSECAM方式などが存在するため、それぞれの形式に対応するモデルが存在する。

TVキャプチャカード上にあるTVチューナーユニットは、アルミ製の箱の形をしていて、かなり目立つ存在だが、モデル名などがはっきりと記載されていることは少ない。また、実際には、ユニット表面にカードベンダーのロゴシールなどが貼られていることが多いため、ショップの店頭において外見からチューナーユニットの種類を見分けることは、きわめて難しい（**写真2**）。

### ●グラフィックスカードとの関係

グラバーチップからの映像信号は、前述のようにPCIバスを経由して、グラフィックスカードに送られる。この映像信号をどのような手段で画面上に表示するかは、グラフィックチップの種類によって異なるが、通常は「オーバーレイ」と呼ばれる方法を使って、通常のディスプレイ画像の上にビデオ画像をはめ込んでいる（**図2**）。

このようなオーバーレイ表示は、グラフィックスカード上のグラフィックチップなどが、オーバーレイに対応していないと不可能である。どのグラフィックチップがオーバーレイ表示に対応しているかどうかは、ここでは詳しく述べる余地はないが、現在多く使われているグラフィックスカードなら、ほとんどの製品が対応しているだろう。

オーバーレイ表示については、画面のモード（解像度やリフレッシュレートの設定）によっては、同じグラフィックスカードでも使えたり使えなかったりすることがある。どの画面モードでオーバーレイ表示が可能かについては、通常、グラフィックスカードのマニュアルに記載されているはずだが、バルクの状態で売られているグラフィックスカードを購入した場合には、詳細はカードメーカーのWebページなどで調べるしかないだろう。

Linux（X Window System）の画面上でテレビ画像をオーバーレイ表示させるには、X Window Systemの画面表示モードを、オーバーレイ表示可能な値に設定しておく必要がある。

オーバーレイ表示が不可能なグラフィックスカードや、オーバーレイ対応範囲外の表示モードで利用している場合でも、テレビ画像の表示は可能である。ただし、この場合、画像データをメインメモリに取り込んでから、改めてグラフィックスカードのフレームバッファに書き込むことになるため、フレーム落ちが頻発して不自然な表示になったり、CPUに負荷がかかったり、グラフィックスカードの描画性能が低下することになるので注意が必要だ。

### ●サウンドカードとの関係

先ほど説明したTVキャプチャカードの構成の中で、音声関係の要素にまったく触れていなかったことに気付いた読者もいるかもしれない。実は、TVキャプチャカードでは、音声に関する処理はほとんど行っていないのだ。チューナー部が受信した音声信号は、直接「ライン出力」端子に出力されるだけで、TVキャプチャカード内部での処理は一切行われない。

音声多重放送などに対応したチューナーに関しては、音声信号の切り替えなどの最低限の操作はできるようになっているが、その場合でも、最終的な音声信号はアナログ信号の形式でカード外に出力されるだけである。

TVキャプチャカードからの音声信号は、カード外部の端子を経由して、PCのサウンドカードの「ライン入力」端子に

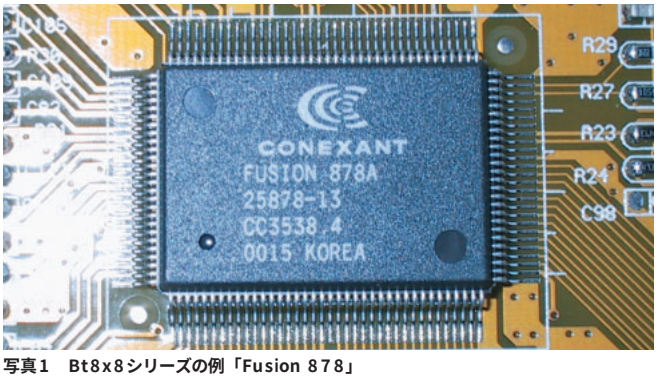



写真1　Bt8x8シリーズの例「Fusion 878」
Brooktree社がCoexant社に買収された結果、「Bt8x8」シリーズの名称は「Fusion 8x8」に変更されている。写真は、もっともよく使われている「Fusion 878A」。

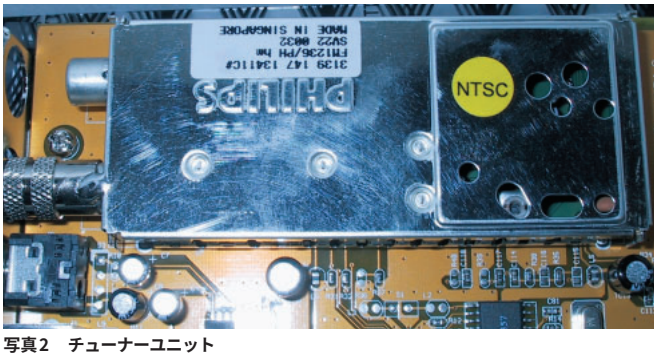
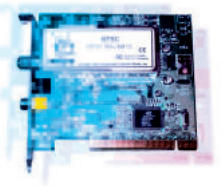



写真2　チューナーユニット
チューナーユニットは銀色の箱（電波をシールドするため）に入っているので、すぐにわかる。写真はPhilips製のチューナーの例だが、実際にはステッカーなどが貼られていてメーカー名やモデル名が分からないようになっている場合が多い。



入力される。そして、PCのスピーカーへの出力や、音声信号のキャプチャ処理は、サウンドカード側で行われるのである（**図3**）。つまり、サウンドカードがなければ、テレビの画像を見ることはできても、音声を聞くことはまったくできないことになる（実際は、TVキャプチャカードのライン出力端子にPC用のスピーカーをつなげば、受信中の音声は聞くことができるが、録音はできない）。

このことをLinuxの立場で考えてみると、テレビの視聴や録画を行うには、サウンドカードも適切に動いていなければならない。すなわち、TVキャプチャカード以外に、Linuxで動作するサウンドカードとドライバが必要になるわけだ。Linuxで使えるサウンドカードの種類やドライバの設定について、本記事中では詳しくは触れない。本誌のバックナンバーなどを参考にしてもらいたい。

## Linuxで使えるTVキャプチャカード

ここまで読んできた皆さんにとって、最大の関心事は「いったい、どのTVキャプチャカードならLinuxで使えるのか」ということだろう。のっけから、期待に背くようで申し訳ないのだが、この問いに対して完璧に回答することは不可能である。その理由は主に2つある。1つは、TVキャプチャカー

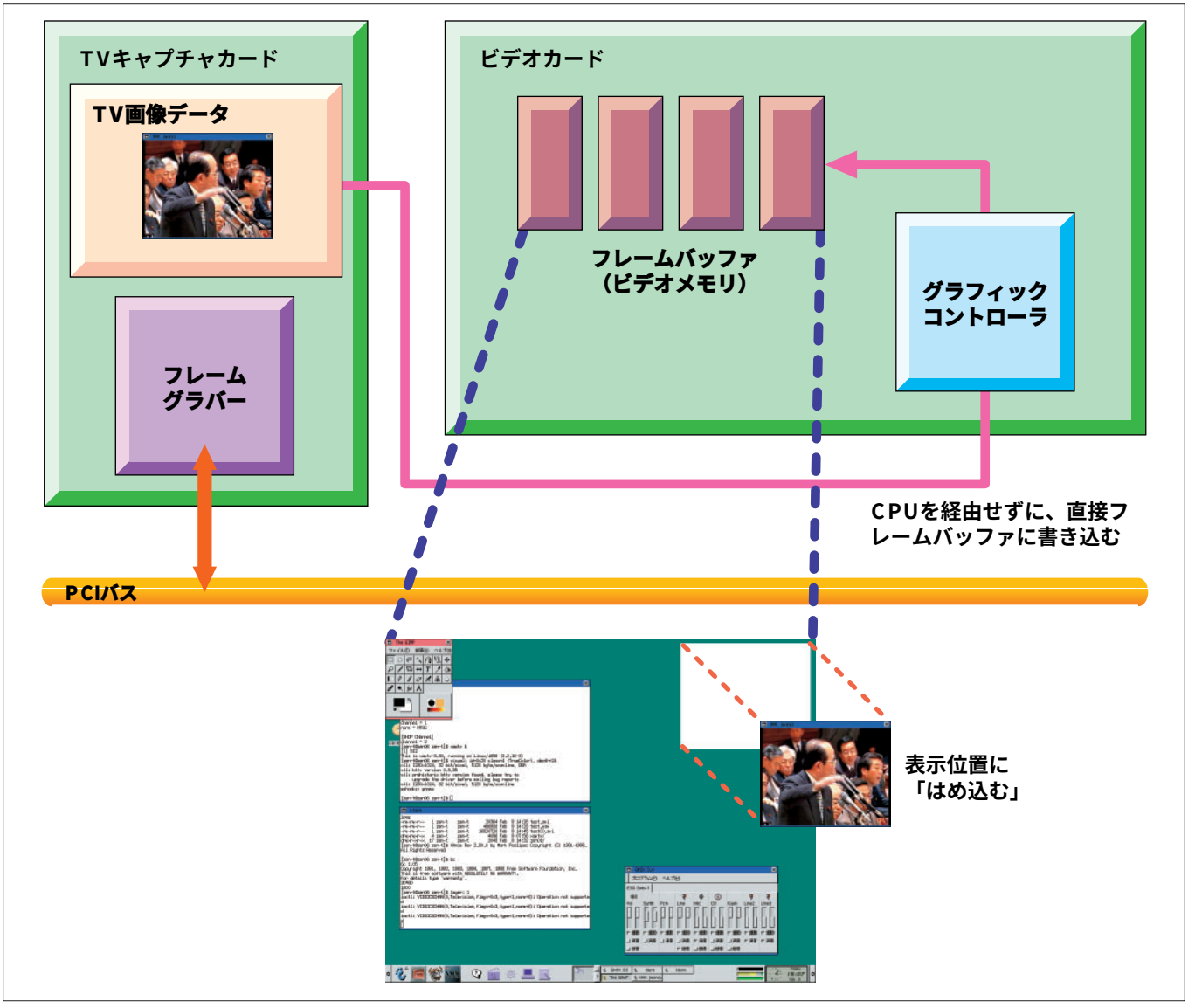



図2　TV画像のオーバーレイ表示
オーバーレイ表示は、CPUを介さずに、キャプチャカードからの画像データをグラフィックスカードのフレームバッファ（ビデオメモリ、VRAM）に直接書き込むことで、PCの画像とTV画像の合成を行う方法だ。CPUに負荷が掛からないのが特徴だが、表示解像度やリフレッシュレートの値によっては、書き込みのタイミングが合わないため、この方法が使えない場合もある。

ドの種類がかなり多く、モデルチェンジもかなり頻繁に行われていること、そして、TVキャプチャカード自体が、PCのほかのコンポーネントに比べるとまだマイナーな存在で、カードのベンダーも中小の企業が多く、製品寿命が短いことにある。

とくに、パーツショップで、半ば「バルク品」のような状態で売られている安価なTVキャプチャカードの中には、ベンダーの住所すら記載されていない製品もある。カードの外見からフレームグラバーチップの種類やチューナーのメーカーがかろうじてわかるだけで、Linuxで使ってみようにも、手も足も出ないような製品がほとんどだ。前述したように、フレームグラバーチップやチューナーに関しては、業界の標準的な製品が使われていることがほとんどなのだが、コラム「Linuxで使えたり使えなかったりするわけ」で説明しているような事情から、Linuxでの利用は簡単ではない。

さらに、安価な製品の中には、カード自身の品質が悪いものも見受けられる。筆者が購入したTVキャプチャカードの中には、Linuxではなんとか動くものの、音声の品質が極端に悪い（ノイズがひどいほか、低音部がまったく聞こえない）など、ほとんど実用にならない製品もあった。このような製品は、付属のドライバを利用してWindows上で使っても、状況は変わらない。ここで特定の製品名をあげることは、いろいろと差し障りがあるので控えたいが、極端に安い値段で売られているカードについては、「とりあえず試してみて、だめならきっぱりあきらめるか、別の用途を探す」といった覚悟で購入すべきだろう。

### ●ベンダー事情

このような不安定なTVキャプチャカード業界の中で、比較的長期にわたって安定した品質のTVキャプチャカード製品を出しているのが、米国のHauppauge Computer Works、ドイツのPinnacle Systems、韓国のSIGMACOM、日本のアイ・オー・データ機器の4社である。そして、これらのベンダーのカードなら、Linuxで利用できる可能性が高い。このうち、筆者が確認した範囲内では、Hauppauge社の製品についてはエスケイネット株式会社（http://www.

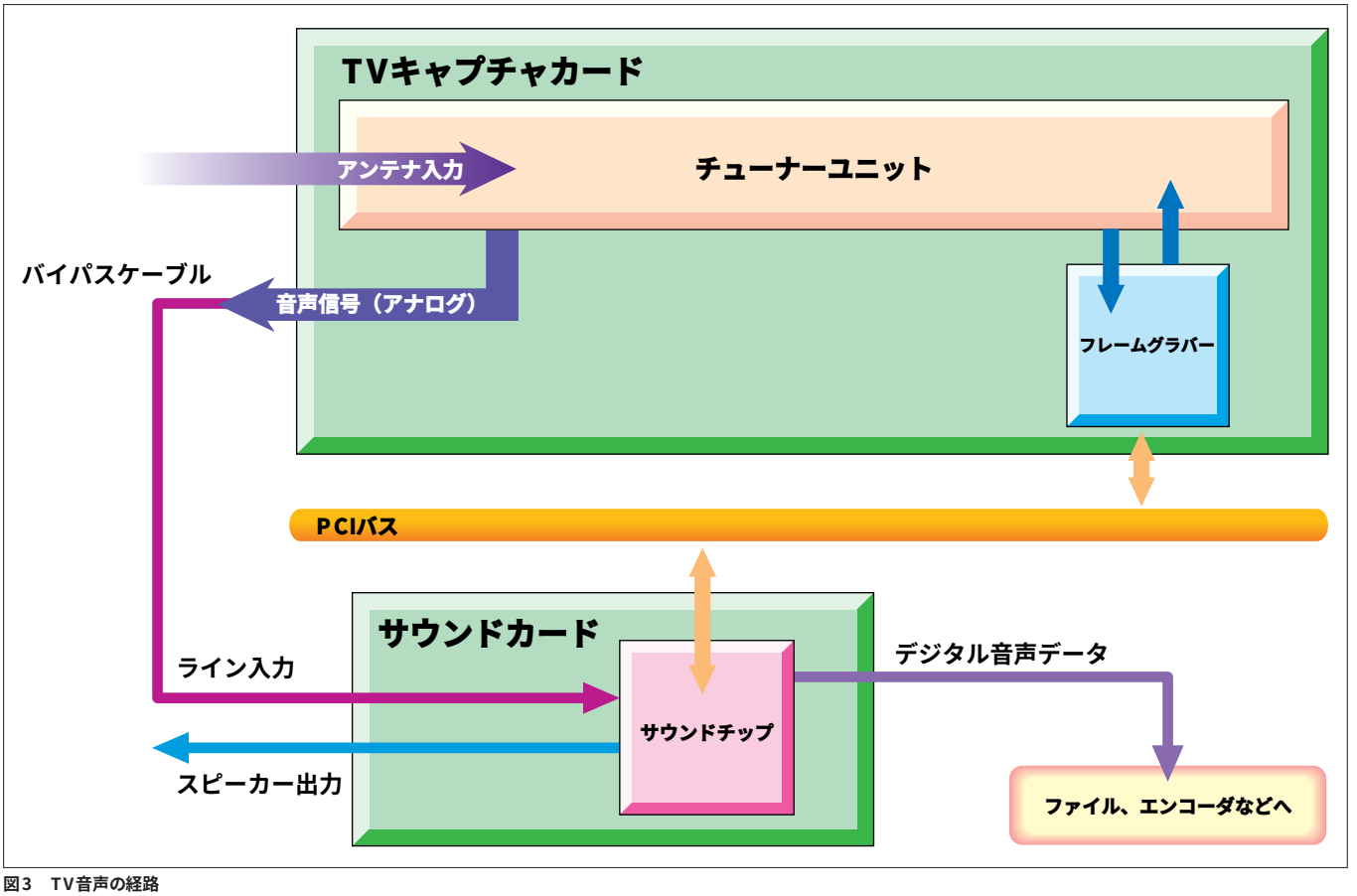
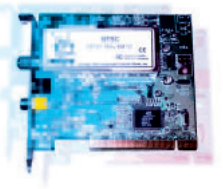



**図3　TV音声の経路**
画像データとは異なり、音声のデータはTVキャプチャカード側での処理はまったく行われない。カード背面の音声出力端子から、サウンドカードの音声入力端子をバイパス用ケーブルで直接に接続し、サウンドカード側でPCのスピーカーへ出力したり、音声のデジタル変換を行ったりする。



sknet-web.co.jp/)、SIGMACOM社の製品についてはアイル株式会社（http://www.ail-corp.co.jp/）がそれぞれ代理店として、日本国内での販売を担当しているようだ（Pinnacle Systemsについては、日本国内に正規の代理店があるかどうか不明）。

国内の代理店が販売する製品を購入すれば、少なくともハードウェアの不良などについては対応してもらえるだろう。ただし、いずれの代理店も、Linuxでの使用については一切保証していないようだ。また、併用するグラフィックスカード、サウンドカード、マザーボードとの相性が悪くてうまく動かなかったり、キャプチャカード自体が仕様変更されたりする場合もあるため、本記事中で試用したカードとまったく同じ製品であっても、実際には使えない可能性もある。TVキャプチャカードの購入にあたっては、個々人で負うべきリスクがあることをご承知おきいただきたい。

「記事を読んでTVキャプチャカードを買ってみたけど、自分の環境では使えなかった」というような苦情には、筆者も編集部も一切対応できないことを、あらかじめお断りしておく。また、このような場合の製品の返品や交換については、ショップや代理店では対応できない場合がほとんどなので、無理な問い合わせは控えてもらいたい。

### ●Linuxで使えるカードの実例

前述したベンダーのTVキャプチャカード製品のうち、筆者が実際に購入してLinux（Turbolinux Workstation 6.0）での動作を確認したのは、以下の2製品である。

- **「WinTV GO for PCI」（Hauppauge Computer Works製、販売　元：エスケイネット）**
- **「SIGMA Cyber TV」（SIGMACOM製、販売元：アイル）**

上記のうち、「SIGMA Cyber TV」は、SIGMACOM製の「SIGMA TV II」というカードに、Windows用のMPEG録画ソフトウェア（CyberLink製PowerVCR）をバンドルしたパッケージのようだが、「SIGMA TV II」カード単体での販売がされているかどうかは不明である。

TVキャプチャカード製品の多くは、FMラジオ機能やハードウェア圧縮機能を持つ上位機種や、USB接続タイプの姉妹機種もあるため、購入時にはパッケージの記述をよく確認して、間違えないようにしてもらいたい（FMラジオ機能についてはLinuxでも利用できるのだが、本記事では言及しない。

## Column

### Linuxで使えたり使えなかったりするわけ

本文中で述べたように、現在市販されているPCIバス接続のTVキャプチャカードのほとんどすべては、ビデオグラバーチップとしてBt8x8シリーズを、チューナーユニットとしてPhilips、Temic、ALPSのいずれかの製品を利用している。このため、多くの製品に互換性があって、Linuxでも簡単に対応できるのではと考えるのは自然だが、実際にはそれほど単純な話ではない。

もっとも大きな問題は、グラバーチップとチューナーユニットの間を結んでいる、制御用信号に関する規格がないことだ。Bt8x8チップには、チューナーなどの外部ユニットのコントロール（選局、音声切り替えなど）を行うために用意された、多目的のレジスタがあり、このレジスタに設定した値に応じて、チップの端子に信号を出力するようになっている。ところが、このレジスタや信号線に関しては、どの信号線にどの機能を割り当てるかという基準が定められていないのだ。

キャプチャカードのメーカーは、それぞれが独自に信号線とチューナーの接続を行っているため、レジスタ中のどのビットがチューナーのどの機能に対応するかは、メーカーやモデルごとにまちまちなのだ。bttvが対応していないTVキャプチャカードを無理に使ってみると「画面は映るのにチャンネル切り替えができなかったり、意図しない動作をする」という状態になることが多い。これはまさに、レジスタの値と制御信号のミスマッチが原因だ。

レジスタの設定情報が公開されていたり、リバースエンジニアリングで割り当てが判明していれば、bttvなどのドライバに反映され、Linuxでも簡単に利用できるようになるのだが、情報公開がされていなかったり、不意の仕様変更などでレジスタの割り当てが変わったりすると、Linuxでは使えないことになる。また、あぶくのように登場しては消えてゆく多数のキャプチャカード製品すべてに対応することは、事実上不可能だ。

TVキャプチャカードの扱いが面倒なのは、Linuxに限った話ではない。Windowsの場合も、OS側でのTVキャプチャカードのサポートは相変わらず不十分だ。特に、Windows Meに変わってからは状況が厄介になっていて、Windows側はTVキャプチャカードの存在を認識して、標準のドライバを勝手にインストールするのだが、それを使えるアプリケーションがないため、カードに付属するドライバにユーザー自身が入れなおさなければならない（Windows Meに正式対応した製品ならよいのだろうが、まだ市場では見かけない）。このような作業を一般ユーザーに強制するのは、かなり酷な話だろう。

ただし、カーネル2.4がUSBに正式対応したことで、このような状況は改善されるかもしれない。USBデバイスのTVキャプチャユニットがLinuxでも使えるようになれば、もう少し簡単に使えるようになるかもしれない。

USBポート経由の接続については、Linuxでの利用は現時点では事実上不可能である)。

どちらの製品も、後述するようにLinuxのカーネルに標準で含まれているキャプチャカード用ドライバ（bttv.o）がデフォルトで対応しているため、比較的簡単な手順でLinux上での利用が可能である。本記事では、このうちの「WinTV GO for PCI」を例に、詳しい使い方を説明することにする。

このほか、Linux側で対応している唯一の国内メーカーであるアイ・オー・データ機器の製品については、「GV-BCTV3」という機種が利用できるはずだが、現在市販されているのはこの後継機種の「GV-BCTV4」のみで、これがLinuxで利用できるかどうかについては、製品が入手できなかったため確認していない。中古品などで「GV-BCTV3」が入手できれば、Linux上で試してみるのも悪くない。なお、「GV-BCTV」シリーズは「テレビ地上波データ放送」に対応しているのが特徴の製品なのだが、Linuxでは残念ながら日本国内のテレビ地上波データ放送には対応していない。また、他のTVキャプチャカード製品に比べるとかなり高価なので、Linuxで利用するためだけに新たに購入するのはちょっともったいない。

## Linuxでのテレビ視聴に必要な環境

では次に、TVキャプチャカードを使ってLinux上でテレビを視聴したり、録画したりするのに必要な環境を見てみよう。ハードウェアについては、前述したようにTVキャプチャカードのほかに、サウンドカードが必要になる。サウンドカードは、マザーボード内蔵のものでも、Linux側のドライバ（OSSやAlsaなど）が対応していれば、まず問題なく使えるはずだ。

ソフトウェアについては、カーネル組み込みのキャプチャカードドライバと、表示用ソフトが必要になる。

### ●bttvとVideo for Linux

ソフトウェア部分でまず必要になるのが、キャプチャカード上のフレームグラバーチップやチューナーを制御し、画像データの流れを制御するためのドライバだ。Linux上でこの役割を負っているのが、「bttv」と呼ばれるカーネル組み込みのドライバ群である。「bttv」は、その名が示すようにBt8x8チップを直接制御するためのドライバである。

bttvは、Linuxのカーネルセットと一緒に配布されているため、通常は改めてインストールする必要はない。ただし、ディストリビューションによっては、bttvのドライバ類を組み込んでいないこともあるので、そのような場合は、カーネルを再構築して、ドライバを「モジュール」として組み込むように設定する必要がある。カーネルの再構築の方法と、組み込むべきモジュールの種類については後述する。

また、ドライバモジュールがすでに組み込まれている場合でも、「デバイスファイル」の作成を行わないと、ドライバは利用できない。デバイスファイルの作成については、「xawtv」のインストール手順の中で説明する。

bttvの最新版は、作者であるGerd Knorr氏のWebサイト（http://www.strusel007.de/linux/bttv/）から入手できるが、通常は、使用中のカーネルに付属するドライバをそのまま利用したほうが賢明だろう。

bttvドライバは、実は「Video for Linux」と総称されるドライバ群の一部である。Video for Linuxは、後述するxawtvなどのアプリケーションに対して、ビデオキャプチャカードやチューナーユニットを制御するためのAPIを提供するものだ。したがって、Linuxのカーネルの設定を行う場合は、「bttv」ではなく「Video for Linux」という項目の設定を変更することになる。

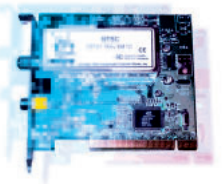
### ●テレビ表示ソフト「xawtv」

ドライバ類が完備していても、それだけではテレビの視聴はできない。受信したテレビ画像をPCの画面に表示するには、それ専用のソフトウェアが必要になるのだ。Linuxの場合、「xawtv」というX Window System用のソフトウェアを利用する。

xawtvは、GPL準拠のフリーソフトウェアで、ディストリビューションによってはバイナリがバンドルされている場合もあるが、TVキャプチャカードがあまり一般的でないこともあって、バンドルされていないディストリビューションが多いようだ。その場合でも、ソースファイルを入手すれば、比較的簡単にコンパイルとインストールができる。

xawtvの最新版のソースファイルは、bttvと同様に、Gerd Knorr氏のWebサイト（http://www.strusel007.de/linux/xawtv/）から入手できる。

xawtvには、非GUI環境（コンソール画面）でテレビを視聴できるようにするアプリケーション（fbtv）、コマンドラインからチューナーをコントロールするツール（v4lctl）、バッチ処理でビデオキャプチャ処理を行うためのツール（streamer）なども付属するので、次回以降に説明する「自動予約録画」も可能になるのだ。



### ●サウンド環境

すでに述べたように、TVキャプチャカードを使ったテレビの視聴や録画には、サウンドカードが必須である。このため、Alsa、OSS、あるいはカーネルドライバに対応したサウンドカードと、ドライバのインストール作業が必須となる。

本記事では、サウンド環境の設定については詳しくは述べないが、Alsaドライバは設定も簡単で対応するサウンドカードの種類も多いのでお勧めである。Alsaドライバは主要なディストリビューションに含まれているほか、http://www.alsa-project.org/からも入手できる。

### ●汎用動画ファイル表示ツール「xanim」

xawtvなどを使って録画した映像ファイルの再生は、「xanim」というツールを使って行う。xanimは、AVI、MPEG、モーションJPEG、GIFアニメーションを含むさまざまな種類の動画ファイルを再生できる汎用ツールで、主要なディストリビューションに付属するほか、http://xanim.va.pubnix.com/home.htmlから入手できる。

録画した直後のデータについては、現在のところxanimなど、モーションJPEGに対応したプレーヤでしか再生しかできないが、MPEG形式などに変換したあとは、たとえばWindowsマシンの「メディアプレイヤー」などでも再生できるようになる。これについては、次回以降に説明したい。

## ハードウェアのセットアップ

一般的な構成のLinuxマシンに対して、追加する必要のあるハードウェアはTVキャプチャカードだけである。もちろん、サウンド機能を持たないマシンの場合は、サウンドカードの追加も必要になる。

本記事でサンプルに使っているマシンは、かなり古いものではあるが、Asus製P2B-N（Intel 440BX使用のNLXマザーボード）をベースにしている。このマザーボードは、グラフィックチップとしてATiのRAGE 3D Pro、サウンドチップとしてESS Solo-1を内蔵しており、これらのデバイスをそのまま利用している。

なお、TVキャプチャの処理には、かなりのCPUパワーが必要になる。最初、筆者はPentium II 350MHzの環境で実験してみたのだが、キャプチャのフレーム落ちがかなり頻繁に発生し、実用にはならなかった。現在は、CPUをPentium III 550MHzに取り替えて、かなり実用的なキャプチャができるようになったが、画面サイズ320×240ピクセル、リフレッシュレート10フレーム／秒程度のキャプチャが限界のようで、それ以上の画面サイズやリフレッシュレートでは、フレーム落ちが多発して、音声と画像との同期が取れなくなる。

これ以上の性能を持った環境（たとえば、現在主流の800MHz以上のCoppermine Pentium IIIやCeleron、AMD Athlon、Pentium 4など）は、使ったことがないのでわからないが、ある程度以上の画面サイズやリフレッシュレートになると、CPUの処理能力よりもPCIバスの転送能力が問題になるかもしれない。いずれにせよ、Pentium III 550MHzというのが、ひとつのボーダーラインであることは覚えておいてほしい。

### ●TVチューナー／キャプチャカードのインストール

すでに述べたように、TVキャプチャカードとしてはHauppauge製の「WinTV GO for PCI」を使用する。このカードは、標準的なサイズのPCIカードなので、ハードウェアのインストールにおいて特別に問題になるようなことはないだろう（**写真3**）。ただし、チューナーユニットの厚みがそれなりにあるので、隣のスロットに装着するカードの種類によっては、うまく収まらない可能性もある。そのような場合には、装着するスロットを変えるなどの対処が必要になることもあるだろう。

### ●サウンドカードとの接続

次に、TVキャプチャカードの「Line OUT」端子と、サウンドカードの「Line IN」端子を、TVキャプチャカードに


写真3　WinTV GO for PCI
本記事でテスト用に使った「WinTV GO for PCI」。秋葉原の大型パソコンショップにて、1万499円（消費税込み）で入手した。写真でもわかるように、カード自体はHauppauge Computer Works社の製品である。国内の代理店が販売しているので、比較的入手しやすいが、Intel製のチップセットを使ったマザーボードでないと、動作しないという情報もあるので注意してほしい。

付属するバイパス用音声ケーブルで接続する（この場合はマザーボード内蔵のサウンド機能を使っているので、筐体背面にあるLine IN端子と接続した）。この接続を行わなかったり、間違った端子に接続してしまうと、テレビの音声が聞こえなかったり、音声のキャプチャができなくなったりするので注意しよう。

### ●アンテナケーブルの接続

テレビ放送を受信するのだから、アンテナを接続しなければならないのは当然のことだ。TVキャプチャカードの背面にあるアンテナ入力端子に、適当なTVアンテナをケーブルで接続するか、部屋などにアンテナ分配端子があるような場合なら、その端子と同軸ケーブルで接続すればよい。

## ソフトウェアのセットアップ

ではいよいよ、もっとも面倒なソフトウェアのセットアップに移ろう。面倒とはいっても、特別な専門知識が必要になることはない。カーネルの再構築が必要になる場合もあるのだが、必要となる設定項目は少ないので、落ち着いて挑戦してもらえれば、難しいことはないだろう。

### ●カーネルの再構築

カーネルを再構築しなければならないのは、次の2つのケースである。

1. **現在のカーネルの設定で、「Video for Linux」関連の設定が「NO」になっている場合。**
2. **カーネルに付属するものよりも新しいbttvドライバを使いたい場合。**

1については、次のコマンドを実行することで、現在Video for Linuxが利用できる状態になっているかどうか確認できる。

```
$ /sbin/modprobe -l bttv.o
```

このコマンドを実行した結果、次のような行が表示されれば、Video for Linuxが利用できるようになっているので、カーネルの再構築は必要ない。

```
/lib/modules/2.2.16-3/bttv.o
```

上のパス中で表示されるカーネルのバージョン番号は、実行中のLinux環境によって変わる。

2の場合に関しては、streamerなどのキャプチャツールを実行する際に、「あなたは古いbttvを使っている。アップデートしろ」というような意味のメッセージが表示されることがある。現在主流のバージョン2.2系のカーネルを使っている場合はとくにそうなのだが、このメッセージはとりあえず無視してもかまわない。bttvのバージョンアップをすれば、対応できるTVキャプチャカードの種類が増えるなどの利点もあるが、ほかのアプリケーションへ悪影響を与える可能性もまったくないわけではないので、問題なく動いているようなら、bttvのバージョンアップはしなくてもよいだろう。

いずれの場合も注意しなければならないのが、カーネルソースツリーの一貫性である。Linuxのディストリビューションの多くは、インストール直後のカーネルソースツリー（通常、/usr/src/linux以下に存在する）の状態が、現在実行中のカーネルの状態と一致していないことがある。このため、いきなり「make config」や「make xconfig」を実行してカーネルを再構築しようとすると、現在のカーネルから大きく環境が変わってしまったり、設定しなければならない項目が膨大な数になったりするなど、大変な作業になってしまう可能性がある。

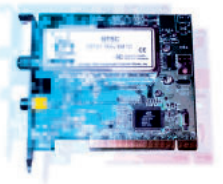
Turbolinuxの場合は、/usr/src/linux/configsディレクトリ中に、標準構成の設定ファイルがいくつか用意されている。

```
$ ls /usr/src/linux/configs
kernel-2.2.16-3-i386.config
kernel-2.2.16-3smp-i386.config
kernel-2.2.16-3BOOT-i386.config
```

たとえば、SMP非対応のカーネルを選択しているのなら、kernel-2.2.16-3-i386.configファイルを、次のようにコピーしておく（suコマンドでrootユーザーになった状態で作業すること）。

```
$ su
# mv /usr/src/linux/.config /usr/src/linux/.config.
backup
# cp /usr/src/linux/configs/kernel-2.2.16-3-i386.config
/usr/src/linux/.config
```

設定ファイルをコピーしたら、次のコマンドを実行する。



```
# cd /usr/src/linux
# make oldconfig
# make dep
```

この状態で「make bzImage」などを実行すれば、ディストリビューションのデフォルトの状態でカーネルを再構築できるが、Video for Linuxを有効にする場合には、次のいずれかのコマンドを実行して、カーネルのオプションを変更する。

```
# make menuconfig
```
（コンソール環境で実行する場合）

あるいは、

```
# make xconfig
```
（X Window System環境で実行する場合）

変更が必要な項目は、「Video for Linux」関連の部分だけで、いくつかある項目をすべて「M」（Module）に設定する（**画面1**）。それ以外の部分については変更の必要はない。

make menuconfig、あるいはmake xconfigが終了したら、次のコマンドを実行してカーネルのコンパイルを行う。

```
# make dep
# make bzImage
```

コンパイルが成功したら生成されたカーネルイメージ（/usr/src/linux/arch/i386/boot/bzImage）を/boot/vmlinuzにコピーし、それが起動されるように/etc/lilo.confを編集してからliloコマンドを実行し、システムを再起動する。

このほかのカーネルの再構築の詳細やliloの使い方などについては、ここでいちいち取り上げるわけにはいかないので、ディストリビューションに付属のドキュメント、解説書などを参考にしてもらいたい（手順は少々違うが本誌3月号の50ページにも解説がある）。

### ●xawtvのソースコードを入手

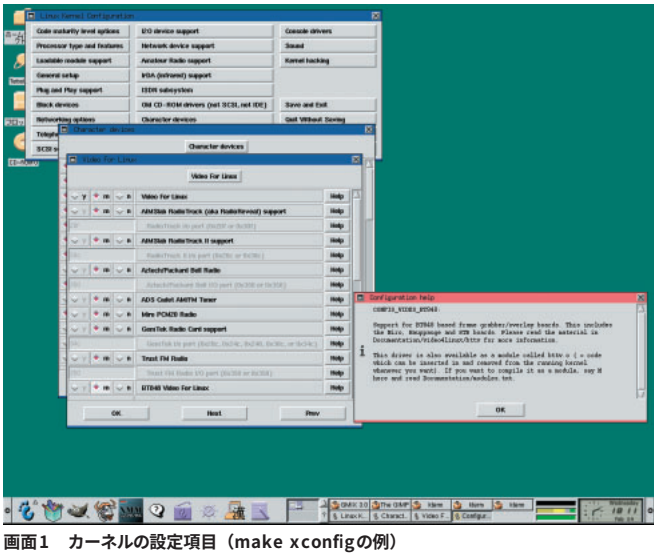
カーネルの準備ができたら、ドライバ（bttv）の設定に移る前に、まずxawtvのソースコードを入手して展開しておこう。xawtvのソースコードには、bttvの設定方法も含めたドキュメント類が含まれているので、作業に移る前にこのドキュメントを一読しておいたほうがよい。

xawtvのソースコードは、すでに述べたとおり、作者であるGerd Knorr氏のWebサイト（http://www.strusel007.de/linux/xawtv/）から入手できる。2001年2月9日現在の最新バージョンは3.30である。ソースコードのファイルは、「xawtv_3.30.tar.gz」というような名前のtarボールになっているので、適当な作業用ディレクトリの上で展開する。

```
$ tar xzf xawtv_3.30.tar.gz
```

すると、作業ディレクトリの中にxawtv-3.30というディレクトリが作成され、その中にソースコード一式が展開されるはずだ。展開されたドキュメントファイルの中で、READMEとREADME.bttvの2つのファイルには、必ず目を通しておいてもらいたい。とくに、README.bttvファイルには、bttvドライバの設定方法や対応するキャプチャカードの機種名などの貴重な情報が含まれているので、英語に抵抗がある場合でも一読しておいてほしい。

なお、JFプロジェクトが日本語に翻訳した「The BTTV Mini-HOWTO」という文書もある。内容は、xawtvのREADME.bttvとほとんど同じなので、こちらを参考にしてもよいだろう（http://www.linux.or.jp/JF/JFdocs/BTTV-Mini-HOWTO-0.3.html）。ただし、bttvやxawtvのバージョンアップに伴って、内容が古くなる可能性もあるので注意す



**画面1　カーネルの設定項目（make xconfigの例）**
make xconfigを実行してカーネルの設定を行う場合は、メインメニューの「Character devices」をクリックし、表示されたウィンドウの「Next」ボタンを何回かクリックして「Video For Linux」のウィンドウを表示させる。いろいろなデバイスのドライバが表示されるが、とりあえずすべてについて「m」を選択しておく。特に、「BT848 Video For Linux」の項目は「m」にしておかないと、ビデオキャプチャは一切できないので注意すること。

ること。

### ●/etc/modules.confの設定

ドキュメントを一読して作業の手順をつかんだら、bttvなどのドライバがきちんとロードされるようにするための設定をする。これは/etc/modules.confファイルで行う（ディストリビューションによっては/etc/conf.modulesなど、ファイル名が異なる場合がある）。

既存の/etc/modules.confファイルの末尾に**リスト1**のような内容を追加すればよいのだが、使用するTVキャプチャカードの種類によっては、記述する内容は変わってくる。たとえば、bttvドライバがカードの機種を自動判別できる場合は、「options bttv」の行でカードの種類を指定するオプション（card=××）は必要ないか、あるいは「card=0」（自動判別）を指定すればよい。また、FMラジオ機能を内蔵したキャプチャカードなら、「radio=1」の設定を追加すれば、FMラジオも聴取できるようになる（この件については本記事では触れない）。

「options tuner」の行では、チューナーユニットの機種を指定する。カードを自動で判別できるような場合は、機種の指定は必要ないこともあるが、今回は念のために「type=2」（Philips製チューナー）を指定している。「debug=1」の設定を追加すると、チューナーの周波数を切り替えるなどの操作をするたびに、ターミナル上にメッセージが表示されるようになるので、動作を確認したいときなどにこのオプションを指定する。ここで指定できるオプション類の意味や設定値の一覧は、すでに述べたように、xawtvのソースファイルに含まれるドキュメント「README.bttv」に詳しく解説されている。

/etc/modules.confの設定が終了したら、次のコマンドを実行して、モジュールを初期化する。

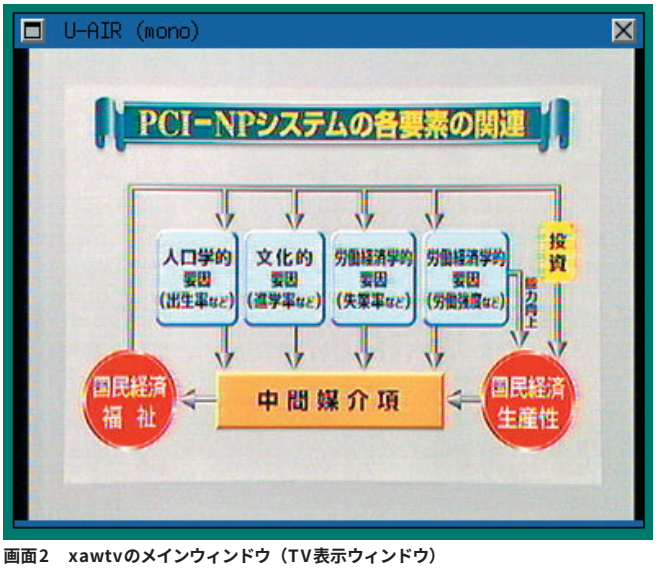
```
# /sbin/depmod -a
```

### ●xawtvのコンパイル

次に、xawtv本体のコンパイルを行う。コンパイル自体はきわめて簡単でmakeコマンド一発で完了する（「make config」などを実行する必要はない）。

```
$ make
```

コンパイルが正常に終了したら、rootユーザーになってからコマンドのインストールを行う。



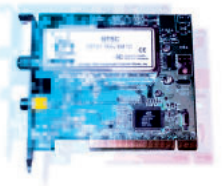

**画面2　xawtvのメインウィンドウ（TV表示ウィンドウ）**
xawtvを起動すると、何の飾りもないウィンドウが1つ表示され、その中に受信中のTV画像が表示される。初めて起動するときは、たいていの場合、ノイズしか表示されないが、気にせずにこのウィンドウ上でマウスで右クリックして、画面3の設定ウィンドウを表示させればよい。

**▼リスト1　/etc/modules.conf（/etc/conf.modules）に追加する内容**

```
# video capture card
# i2c
alias char-major-89     i2c-dev
options i2c-core        i2c_debug=1
options i2c-algo-bit    bit_test=1

# bttv
alias char-major-81     videodev
alias char-major-81-0   bttv
pre-install bttv        modprobe -k msp3400; modprobe -k tuner
options bttv    card=0
options tuner   type=2
```

カードの種類を手動で設定するときはこのオプションで指定する

チューナーの種類を手動で設定するときはこのオプションで指定する



```
$ su
# make install
```

この作業で、xawtv本体だけでなく、コマンドライン用の各種ツール類も一緒に/usr/local/bin/ディレクトリにインストールされる。また、オンラインマニュアルもインストールされる。

### ●デバイスファイルの作成

インストール作業の最後は、デバイスファイルの作成だ。この作業は、xawtvのソースコードのディレクトリにある「MAKEDEV.v4l」というシェルスクリプトを使えば完了する（rootユーザーの状態で実行すること）。

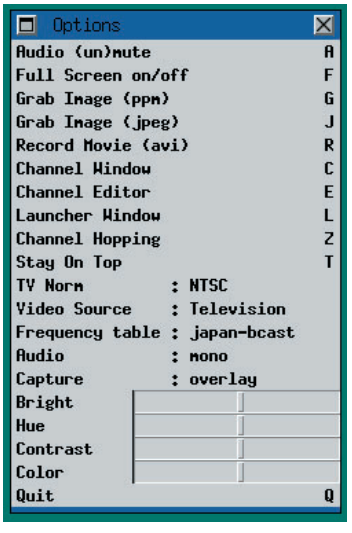
```
# ./MAKEDEV.v4l
```

すると、/devディレクトリ以下に、video、radio、vtx、vbiなど一連のデバイスファイルが作成される。

## とにかく使ってみよう

ハードウェアとソフトウェアのインストールが完了したら、xawtvを起動して、正しくテレビが見られるかどうか確認しよう。xawtvの起動は、X Window Systemを起動した状態で、シェルのコマンドラインから次のコマンドを実行するだけでよい。

```
$ xawtv &
```




**画面3　xawtvの設定ウィンドウ（Optionsウィンドウ）**
xawtvの各機能を設定するウィンドウ。メインウィンドウ上でマウスを右クリックすると表示され、もう一度右クリックすると閉じることができる。各メニュー項目の右側の英文字は、ショートカットキーである（メインウィンドウにフォーカスがある状態でこのキーを押せば、各機能を直接呼び出すことができる）。xawtvを最初に起動したときは、まず「TV NORM」と「Frequency table」の設定を行う。TV放送ではなく外部ビデオ入力の画像を表示させたい場合は、「Video source」を「Composite1」か「S-Video」に設定する。

すると、**画面2**のようなウィンドウが表示されるはずだ。起動直後は、テレビ画面のウィンドウにはノイズしか表示されていないはずである。xawtvはデフォルトでヨーロッパの放送方式で受信するような設定になっているため、日本のテレビ放送に合わせた設定が必要になるからだ。

xawtvのウィンドウ上でマウスを右クリックすると、画面3のようなウィンドウが表示される。ここで、［TV NORM］というドロップダウンリストをクリックして、「NTSC」を選択し、［Frequency table］のリストから「japan-bcast」を選択する（CATV放送を受信したい場合は、「japan-cable」を選択する）。そして、再びテレビ表示用ウィンドウにフォーカスを移した状態で、上下矢印のカーソルキー（↑、↓）を押してみると、キーを押すのに伴ってチャンネルが切り替わり、テレビ放送の画面が表示されるはずだ。

ここで、［TV NORM］のドロップダウンリストには「NTSC-JP」という項目もあって、こちらを選択するほうが正しいように思えるのだが、筆者が使っているTVキャプチャカードでは、なぜか「NTSC」を選択しないと正しく表示できなかった。このあたりは、使っているTVキャプチャカードの種類によって状況が変わると思うので、自分の環境でうまく表示できなかった場合には、設定をいろいろと切り替えて、もっとも綺麗に表示できる設定を探してみてほしい。

### ●音が出ない場合

テレビの画像が表示されるのに、音が出ない場合は、サウンドカードの音量設定が適切になされていないことが多い。GMIXなどのミキサーアプリケーション（**画面4**）を起動して、［Line］の音量が0になっていないか、［Mute］（日本語化バージョンでは［消音］）のスイッチがオンになっていない


**画面4　音量の設定（GMIX）**
受信中のTV放送の音量を調整したい場合は、GMIXなどのミキサーアプリケーションで「Line」のレベルを調整する。ここで「録音」のボタンがオフになっていると、キャプチャを行ったときに音声が録音されなくなるので注意すること。また、サウンドカードによっては「IGain」を調整しないと、録音レベルが高すぎたり低すぎたりすることがあるので注意する（IGainの設定値がまったく意味を持たないサウンドカードもある）。

かなどを確認する。

### ●~/.xawtvファイルの作成

受信方式などの設定が終わったら、受信するチャネル一覧を設定する。xawtvのテレビウィンドウでマウスを右クリックして、表示されたコントロールメニューの［Channel Editor］をクリックする。すると、**画面5**のような「Channel Editor」ウィンドウが表示されるので、テレビ表示画面でチャンネルを選択してから、［Station ID］のテキストボックスに放送局の名前を英数字で簡潔に入力して［Add］ボタンをクリックする。この操作を、受信したいチャンネルの数だけ繰り返す。

最後に［Save］というボタンをクリックすれば、既定の受信方式を含めたxawtvのすべての設定が、自分のホームディレクトリ上に「.xawtv」という名前のテキストファイルとして保存される。参考までに、.xawtvの例をリスト2に挙げる。.xawtvファイルが作成できれば、次回にxawtvを立ち上げたときにはこの内容で自動的に設定が行われるので、受信方式などを改めて設定する必要はなくなる。

なお、今回掲載した2つのリストは、テキストファイル（/TV/list.txt）として付録CD-ROMに収録されている。

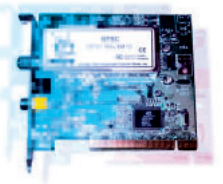
### ●静止画キャプチャ

xawtvでテレビ画像の静止画キャプチャを行うには、コントロールメニューの［Grab Image］を使う。Grab Image機能には、ppm、jpegの2通りがあるが、これはそれぞれPPM形式、JPEG形式の画像ファイルを作成する。たとえば、JPEG形式の画像ファイルを作成したい場合には、「Grab Image (jpeg)］をクリックする。すると、現時点でテレビウィンドウに表示されている内容が、xawtvを起動した際のカレントディレクトリ上に「snap-2-20010209-092357-1.jpeg」というような名前のファイルとして保存される。

保存される画像のサイズは、表示中のテレビウィンドウのサイズとは関係なく、768×480ピクセルの24ビットカラー画像になる。使用するTVキャプチャカードの機種にもよるが、このキャプチャ画像の品質はかなり高い。たとえば、自分で撮影したビデオテープなどから静止画像を作成するような用途にも向いているので、ぜひ試してみてほしい。また、本記事では詳しくは触れないが、xawtvには、静止画キャプチャしたデータをWebサイトに自動的にアップロードするツール「webcam」も付属しているので、Linuxマシンをライブカメラサイトの運用に使うこともできる。

▼リスト2 ~/.xawtvの例

```
[global]
freqtab = japan-bcast
pixsize = 128 x 96
pixcols = 1
jpeg-quality = 75
mjpeg-quality = 75
keypad-ntsc = no
osd = yes

# [Station name]
# capture = overlay | grabdisplay | on | off
# input = Television | Composite1 | S-Video | ...
# norm = PAL | NTSC | SECAM | ...
# channel = #
# fine = # (-128..+127)
# key = keysym | modifier+keysym
# color = #
# bright = #
# hue = #
# contrast = #

[defaults]
norm = NTSC
capture = over
input = Television

[U-AIR]
channel = 11
norm = NTSC

[NHK]
channel = 1
norm = NTSC

[SHOP CHannel]
channel = 2
norm = NTSC

[NHK-Edu]
channel = 3
norm = NTSC

[NTV]
channel = 4
norm = NTSC

[TV Saitama]
channel = 5
norm = NTSC

[Fuji TV]
channel = 8
norm = NTSC

[Wink]
channel = 9
norm = NTSC

[TV Asahi]
channel = 10
norm = NTSC

[TV TOKYO]
channel = 12
norm = NTSC
```



### ●動画キャプチャ

いよいよ本命の「動画キャプチャ」すなわち「録画」に挑戦してみよう。xawtvのGUI画面から動画のキャプチャを行うには、コントロールメニューの［Record Movie］を選択する。すると、画面6のような「Record movies (avi)」ウィンドウが表示されるので、録画するファイルの形式やファイル名などを指定する。使用するTVキャプチャカードによって、ここで選択できる形式は変わってくることもあるが、ここでは、［movie driver］の項目を「Microsoft AVI (RIFF) format」に、［video format］を［JPEG］に設定している。これは、録画ファイルをMPEG形式に再圧縮する際に必須となる設定なのだが、今のところは、いろいろな形式でのキャプチャを試してみてよいだろう。特に、［frames/sec］や［video size］の設定をいろいろと変えてみて、自分のマシン環境でどの程度の精細度でのキャプチャが可能になるのかを試してみてほしい。

### ●キャプチャした動画の再生

キャプチャした動画データは、ほとんどのディストリビューションに標準で付属するアプリケーション「xanim」で再生できる。動画のキャプチャを行ったときに表示される「Record movies (avi)」ウィンドウにある［playback (start xanim)］ボタンをクリックすれば、xanimが起動して、直前にキャプチャした動画を再生してくれる（**画面7**）。xanimのコントロールウィンドウにある、各種ボタンをクリックすれば、一時停止、再開、音量調整、逆再生などの操作を、ビデオデッキのリモコンのような感覚で行うことができる。またマウスやキーボードによる制御もできる。詳細については、xanimのオンラインマニュアルを参照してもらいたい。

コマンドラインからxanimを起動する場合には、次のようにファイル名を指定してコマンドを実行すればよい。

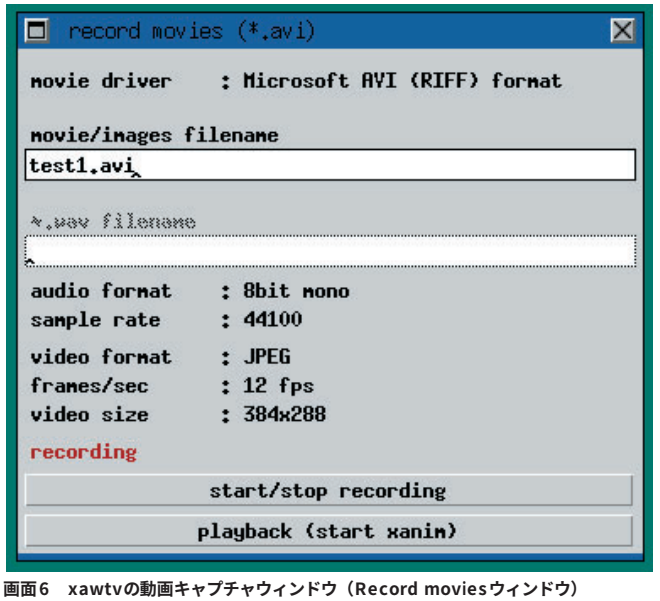
```
$ xanim testmovie.avi
```

キャプチャしたファイルを再生しても、音声がまったく聞こえない場合は、ミキサーの録音設定が正しく設定されていない可能性がある。GMIXなどを起動して、Line入力の［Capture］（日本語版の場合は［録音］）ボタンがオンになっているかどうかを確認して、もう一度キャプチャをやりなおしてみる。また、再生画像が早回しのように進み、音声が少しずつ遅れていくような場合は、キャプチャの処理が過重で、フレーム落ちが発生しているので、フレームレートを落とすか、画像サイズを小さくする必要がある。

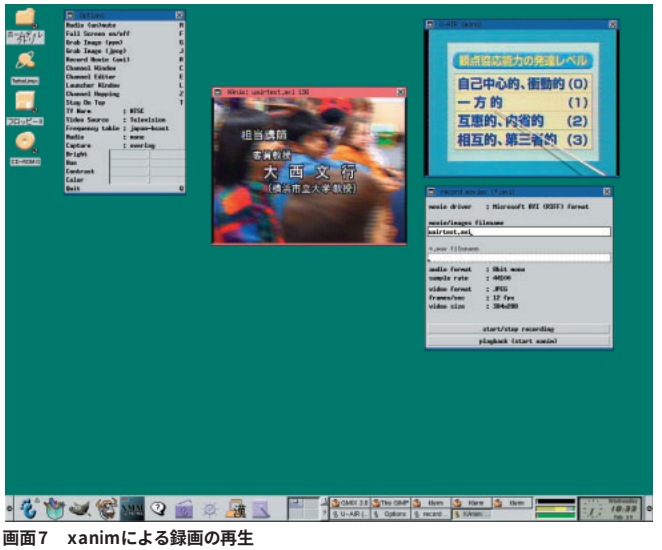
いろいろな設定でのキャプチャを試してみて、自分のマシン環境にもっとも適した設定を見つけてほしい。

●

次回は、以上の作業で見つけた最適なキャプチャ設定を利用して、バッチ処理による自動録画を行ってみる。




画面6　xawtvの動画キャプチャウィンドウ（Record moviesウィンドウ）
設定ウィンドウで「Record Movie」を選択すると、このウィンドウが表示される。ここで、作成するビデオファイルの形式、音声のサンプリング方法、画面の大きさやフレームレートなどを設定する。ちなみに、「movie driver」で「multiple image data」を選択すると、1フレームごとにつき1つのファイルとして保存されるので、カレントディレクトリに膨大な数の画像ファイルが作成されることになるので注意。また、ここで保存した「AVI」形式のファイルは、Windowsのメディアプレイヤーなどでは再生できない。



画面7　xanimによる録画の再生
画面6の「playback(start xanim)」ボタンをクリックすると、直前に録画したファイルが、xanimアプリケーションによって再生される。これもシンプルな画面のアプリケーションだが、ウィンドウ上でマウスの中央ボタンをクリックすると再生が一時停止し、右クリックでコマ送り、左クリックで逆コマ送り、中央ボタンクリックで再生を再開できる。音声が聞こえなかったり小さかったりする場合は、GMIXなどのミキサーアプリケーションで、「Pcm」の音量を調整してみる。

連載特集

申し込みから独自ドメイン運用まで徹底解説！

# フレッツ・ISDNで常時接続インターネットサーバ!

## 独自サーバで自分だけのインターネットサイトを構築しよう

今年になって、ADSLや光ケーブルによるFTTH（Fiver To The Home）など、新しい常時接続環境のサービスが続々と提供されるようになってきた。編集部のサーバはフレッツ・ISDNなので、これらのサービスに比べると速度の面で若干の難はある……。しかし、サーバで設定することはこれらのサービスでも変わらない。
今回は、少しサーバのメンテナンス向けの話となる。まず最初にサーバの時刻を正確に保つためのタイムサーバを構築する。続いて、インターネットからサーバに安全にリモートログインするためのSSHサーバを構築する。

ACT. 4

# タイムサーバを構築しよう

常時接続

PCの時計が正確かどうかは、あまり気にしたことがないユーザーが多いかもしれない。しかし、サーバマシンの時計となるとそうはいかない。メールヘッダが狂ってしまい、過去や未来のタイムスタンプが付いたメールを送受信してしまったり、新しいファイルを古いファイルで上書きしてしまうことになりかねない。

それ以上に大きな問題は、ログの記録時間が「不正確」になってしまうことだ。ログファイルは、システムの問題（イベント）や、クラックされたかどうかを追跡する重要な情報が、時間とともに記録されている。しかし、この時間が正確でないと、本当はいつのログなのかがわからなくなってしまう。

Linuxでは、時計が2つ動いている。1つはBIOSのCMOSクロックで動いているハードウェアクロック（CMOSクロック）で、もう1つはカーネルの内部で動いている内部時計（カーネルクロック）である。しかし、実はこの時計には誤差があって、微妙にずれていく。

カーネルクロックを合わせるには、dateコマンドを使う。時刻と日付の指定は、月日時分年.秒という形式（MMDDhhmmCCYY.ss）で指定する。dateコマンドで設定されるのはカーネルクロックなので、CMOSクロックも同期しなければならない。CMOSクロックを同期させるには、clockコマンドを使う。たとえば、2001年3月1日10時30分00秒に合わせるには、

```
# date 030110302001.00
# clock -w
```

このようにすることで、時計を合わせることはできるが、日々時計を合わせるのではシステム管理者に負担がかかってしまう。そこで、タイムサーバを使って、自動的に時計を合わせるようにしよう。

インターネットには、タイムサーバ（NTPサーバ）と呼ばれる、正確な時刻を提供するサーバが用意されている。このタイムサーバから貰った時刻に自分のサーバの時計を合わせることで、つねに正確な時刻に保つことができる。さらに、内部の時計にどのくらいの誤差が発生するのかを調査し、自動的に誤差を補正する。これによって、より正確な時刻に近づける。

実際には、いきなり時計を合わせるのではなく、徐々に正確な時刻に近づける。遅れていれば時計の進みを速め、進んでいれば時計の進みを遅らせるように微妙に調整するのだ。このため、いきなり時刻が変わるわけではない。これによって、時間が飛んでしまうこと（時刻のロスト）を防いでいる。

タイムサーバは、DNSのように階層構造だと考えればよい。最上位のタイムサーバは、stratum 1と呼ばれている。下位のタイムサーバは、順にstratum 2、3……と数値が大きくなる。

最上位のタイムサーバ（stratum 1）はGPSなどから正確な時刻を取得し、下位のタイムサーバに伝え（stratum 2）、さらにまた下位の……という構造だ。これは、上位のタイムサーバの負担を軽減するためである。もちろん、タイムサーバはネットワークの遅延なども考慮して正確な時刻を転送するようにしている（ただし、ADSLのように上りと下りで速度が異なる通信経路の場合は、誤差が大きくなる）。

また、なるべくネットワーク的に近いタイムサーバを参照するように、通常は、プロバイダが提供するタイムサーバを利用すべきだ。

しかし、編集部が加入したNTTコミュニケーションズのInfoSphereでは、残念ながらタイムサーバが提供されていない。そこで、仕方なくアスキーのタイムサーバを利用することにする。

## タイムサーバの構築

タイムサーバは、NTP（Network Time Protocol）プロトコルを使用する。NTPはポート123番を使用するので、ダイヤルアップルータのIPパケットフィルタリングで、ポート123番が通過できるように設定しておこう。

タイムサーバの構築に必要となるパッケージは、次のものだ。

```
ntp-4.0.99j-7.i386.rpm
```

このパッケージはRed Hat Linux 7J Disc1に収録されている。また、Red HatのFTPサイト（ftp://ftp.redhat.com/pub/redhat/redhat-7.0/i386/ja/RedHat/RPMS）からもダウンロードすることができる。

編集部で構築したサーバにはインストールされていなかったので、まずはインストールを行う。

```
# rpm -ivh ntp-4.0.99j-7.i386.rpm
```

次に、上位のタイムサーバの時計と、自分のサーバの時計を合わせておく。これは、前述したように徐々に時刻を正確に近づけていくため、あまり誤差が大きいと調整できなくなってしまうからだ。上位のタイムサーバの時刻にマシンの時計を合わせるには、ntpdateコマンドに「-b」オプションを指定して実行する（**画面1**）。

これで自分のサーバのカーネルクロックが正しく設定される。もし、よくサーバをリブートするのであれば、ブート時に自動的に実行するようにしたほうがいいだろう。

次にタイムサーバの設定だ。タイムサーバの設定ファイルは、/etc/ntp.confである。ntp.confファイルは上位タイムサーバを指定するだけでいい。たとえば、

```
server 111.122.133.144
```

と指定する（IPアドレスは架空）。IPアドレスを直接記述しているのは、DNSに余計な負担をかけないことと、セキュリティのためだ。

**リスト1**は、ntp.confのサンプルである。最後の「restrict default nomodify」は、外部からの時刻変更要求を拒否するための設定だ。

設定が完了したら、タイムサーバを起動しよう。

```
# /etc/init.d/ntpd start
```

または、

```
# /etc/rc.d/init.d/ntpd start
```

```
# ntpdate -b ntp.hogehoge.ne.jp
23 Feb 18:43:48 ntpdate[29590]: step time server 111.122.133.144 offset -174.838369 sec
```

画面1　ntpdateコマンドによる時刻の設定

```
# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*ntp.hogehoge.ne ntp1.booboo.ne.  4 u   31   64  177   79.443   19.513   0.840
+ntp.foofoo.ne.j ntp1.foobar.ne   4 u   28   64  177  294.284  -22.106   1.489
```

画面2　ntpqコマンドによるタイムサーバの同期情報

| 項目 | 意味 |
|---|---|
| remote | タイムサーバのホスト名 |
| refid | タイムサーバが参照しているホスト名 |
| st | statum |
| t | タイムサーバのタイプ<br>u:unicast<br>m:multicast<br>l:Local<br>b:broadcast |
| when | 最後にパケットを受け取った時間（秒） |
| poll | ポーリング間隔（秒） |
| reach | 到達可能性を表すレジスタデータ（8進数） |
| delay | 通信による遅延（ミリ秒） |
| offset | 参照しているタイムサーバとのずれ（ミリ秒） |
| jitter | 時刻のばらつき（ミリ秒） |

表1　ntpqで出力される内容

タイムサーバが正常に動作しているかどうかを確認するには、ntpqコマンドに「-p」オプションを付けて実行すればいい。

ただし、タイムサーバの同期にしばらく時間がかかるので、すぐには正しい情報は表示されない。おおむね5〜10分程度必要だ。

正常に同期できていれば、**画面2**のように表示される。各項目の内容は**表1**を参照してほしい。簡単にいえば、サーバの名前の前に「＊」が表示されているサーバと同期していることになる（**表2**）。

正常に同期できているのであれば、ntsysvコマンド（Turbolinuxでは Turboserviceコマンド）で、ntpdがシステム起動時に有効になるように設定しておこう。

なお、同一組織の複数のサーバからインターネット上の同一のタイムサーバを参照するようなことはしないようにしよう。1台のサーバを外部のタイムサーバに同期させ、ほかのサーバはそのサーバを参照すればよい。

もし、身近にタイムサーバがないようであれば、今回構築した編集部のタイムサーバ（202.239.88.250）を公開しているので、このタイムサーバを参照するのもいいだろう。

▼リスト1　/etc/ntp.conf

```
server 111.122.133.144
server 144.133.122.111
restrict default nomodify
```

| 記号 | 意味 |
|---|---|
| ＊ | 同期中のサーバ |
| ＋ | 同期可能なサーバ |
| # | 距離が遠いサーバ |
|  | 未接続状態のサーバ |
| X | 利用できないサーバ |
| − | 利用できないサーバ |
| . | 利用できないサーバ |

表2　タイムサーバの同期状態

# リモートログインサーバを構築しよう

Linux（UNIX）の便利な機能の1つにtelnetがある。telnetは、ネットワーク上にあるLinuxマシンに、リモートログインするためのもので、たとえ遠隔地にあるマシンだとしても、あたかも手元にあるマシンであるかのように使うことができる非常に便利なものだ。

しかし、このtelnetにはセキュリティ上、重大な問題がある。それは、パスワードを含め、すべての通信内容が平文（クリアテキスト）で送信されてしまうというものだ。

これは、インターネット上の第三者に通信内容を傍受されるだけでなく、場合によってはマシンを乗っ取られてしまう危険が高い。

たとえば、インターネット上のリモートホストに、telnetコマンドでログインしたとしよう。このとき、あなたのアカウントとパスワードは筒抜けだ。さらに、不注意にもsuコマンドでrootユーザーのパスワードを送信してしまったとしたら……。第三者にrootユーザーのパスワードが知られてしまい、いとも簡単にあなたのマシンを乗っ取られてしまうだろう。

このようなことを避けるために、インターネット上で不用意にtelnetコマンドを使うことは避けるべきだが、会社や学校などからリモートログインしたいことも多い。実際、リモートログインできないとかなり不便だ。このような場合は、暗号化されたSSH（Secure SHell）を使うようにしよう。

## SSHによるリモートログイン

Linux上のSSHの実装はいくつかあるが、ここではOpenSSHを使用する。SSHはリモートコマンド（rシリーズと呼ばれる）を代替するためのものだ。

OpenSSHは、RSAアルゴリズムを使った公開鍵方式による暗号化手法を利用している。公開鍵方式による暗号化手法では、「公開鍵（Public Key）」と、「秘密鍵（Private Key）」という2つの対になった鍵が使われる。それぞれの鍵は、暗号化（符号化）を行う際と、暗号を解読（復号化）する際に利用される。簡単にいえば、公開鍵によってデータが暗号化され、この暗号化されたデータは、対となる秘密鍵がなければ解読することができない。

これは、誰でもこの公開鍵を使って暗号化することはできるが、解読できるのは対となる秘密鍵を持った人だけに限定できるということを意味する。つまり、インターネット上で公開鍵を公開しておいたとしても、対になる秘密鍵を持っていなければ解読できないというわけだ（**図**）。

実際にSSHサーバを運用する際には、サーバ側（リモートログインされる側。リモートホスト）には公開鍵を置いておき、インターネット上から接続するクライアント（リモートログインする側。ローカルホスト）には秘密鍵を置いておくことになる。

このように、OpenSSHではRSAという強力な暗号化アルゴリズムを使用することで（そのほかの暗号化手法も利用可）、高いセキュリティレベルを実現している（**コラム**参照）。

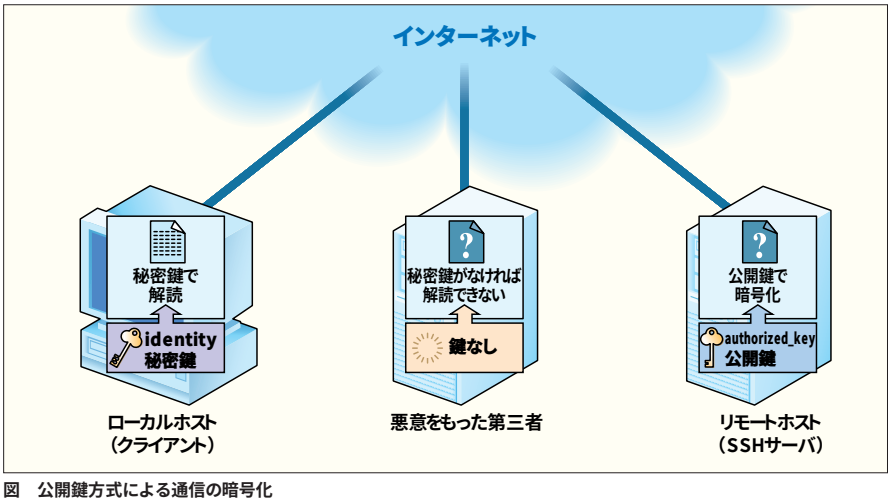



図　公開鍵方式による通信の暗号化

## OepnSSHのインストール

SSHサーバは、sshプロトコルを利用する。sshプロトコルはポート22番（ウェルノウンポートではsshと定義）を使用するので、ダイヤルアップルータのIPパケットフィルタリングで、ポート22番が通過できるように設定して

おこう。

OpenSSHを利用するには、**表3**のパッケージが必要だ。インストールされていない場合は、rpmコマンドを利用してインストールする。

実際には、リモートホストとローカルホストで必要となるパッケージは異なるが、リモートログインするマシンが逆になれば、リモートホストとローカルホストは逆となるので、両方のマシンにすべてのパッケージをインストールしておいたほうがいいだろう。

## 鍵の作成

インストールが完了したら、公開鍵と秘密鍵の対を作成する。鍵を作成するマシンは同一のユーザー名であれば、リモートホスト上でも、ローカルホスト上でもかまわない。ただし、ネットワーク経由で作成したのではセキュリティ上意味がないので、コンソール上で作成する必要がある。なお、今回はリモートホスト上（サーバ側）で作成することにした。

鍵の対を作成するには、作成するユーザー名でログインし、ssh-keygenコマンドを使用する。

ssh-keygenコマンドを実行し、鍵のファイル名とパスフレーズを2回入力すれば鍵の対が作成される（**画面3**）。パスフレーズとは、ログインするための認証パスワードだと考えればよい。コマンドを実行すると、ユーザーのホームディレクトリにある「.ssh」ディレクトリに、「identity.pub」という名前で公開鍵が、「identity」という名前で秘密鍵が作成される。

ここで作成されたidentity.pub（公開鍵）は、authorized_keysという名前に変更しておく必要がある。

```
$ mv identity.pub authorized_keys
```

次に、ローカルホストのホームディレクトリに.sshという名前でディレクトリを作成し、先ほどの秘密鍵（identity）をコピーする。ネットワークを経由してコピーするとセキュリティ上問題があるので、今回はFAT形式のフロッピーディスクを経由して物理的にコピーすることにしよう。

フロッピーディスクをドライブにセットし、**画面4**のようにして秘密鍵をフロッピーディスクにコピーする。

これで秘密鍵（identity）がフロッピーディスクにコピーされた。今度は、このフロッピーディスクをローカルホストのドライブにセットし、**画面5**のようにして秘密鍵をコピーする。

以上で鍵の準備はOKだ。ただし、どんなにSSHが安全でも、秘密鍵が漏れてしまっては意味がない。秘密鍵の扱いには十分注意する必要がある。

## OpenSSHサーバの構築

OpenSSHサーバの設定は、リモートホスト上の/etc/ssh/sshd_configで行

| パッケージ名 | 内容 | ローカルホスト | リモートホスト |
|---|---|---|---|
| openssh-2.1.1p4-1 | OpenSSHツール、ライブラリなど | 必須 | 必須 |
| openssh-server-2.1.1p4-1 | OpenSSHサーバ | | 必須 |
| openssh-clients-2.1.1p4-1 | OpenSSHクライアント | 必須 | |
| openssl-0.9.5a-14 | 暗号化ライブラリ | 必須 | 必須 |
| zlib-1.1.3-12 | 圧縮ライブラリ | 必須 | 必須 |

表3　OpenSSHに必要なRPMパッケージ一覧

```
$ ssh-keygen
Generating RSA keys:  ..............oooooo0.........oooooo0
Enter file in which to save the key (/home/hogehoge/.ssh/identity):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hogehoge/.ssh/identity.
Your public key has been saved in /home/hogehoge/.ssh/identity.pub.
The key fingerprint is:
12:34:56:78:9a:bc:de:f0:12:34:56:78:9a:bc:de:f0 hogehoge@ns1.ascii-linux.com
```

[Enter]キーを押す
パスフレーズを入力
パスフレーズをもう一度入力

画面3　鍵の作成

```
$ su -                                  rootユーザーになる
# mformat a:                            FAT形式でフォーマット
# mount /mnt/floppy                     フロッピーディスクをマウント
# cd /home/<ユーザー名>/.ssh            ディレクトリの移動
# cp identity /mnt/floppy               秘密鍵のコピー
# sync                                  キャッシュの書き込み
# umount /mnt/floppy                    フロッピーディスクのアンマウント
```

画面4　秘密鍵をフロッピーディスクにコピー（リモートホスト）

```
# mount /mnt/floppy                     フロッピーディスクのマウント
# cd /home/<ユーザー名>                 ディレクトリの移動
# mkdir .ssh                            .sshディレクトリの作成
# ch .ssh                               ディレクトリの移動
# cp /mnt/floppy/identity .             秘密鍵のコピー
# chown <ユーザー名>:<ユーザー名> identity   所有権の変更
# chmod 600 identity                    パーミッションの変更
```

画面5　フロッピーディスクから秘密鍵をコピー（ローカルホスト）

う。このファイルはOpenSSHをインストールすると用意されているので、**リスト2**のように、一部を修正する。

**リスト2**で修正しているのは、Linuxのパスワード認証方式を使わないように設定し、公開鍵と秘密鍵による認証のみを行うように設定している。

次に、リモートマシン側のスーパーサーバの設定を確認しておこう。OpenSSHはスーパーサーバから起動されるデーモンではないが、スーパーサーバのアクセス制限を参照しているので、スーパーサーバのアクセス制限が設けられているローカルホストからは接続することはできない。Red Hat Linux 7Jのデフォルトではスーパーサーバにアクセス制限は設けられていないが、これ以外のディストリビューションや、独自に制限を設けた場合は設定が必要になるかもしれない。

スーパーサーバのアクセス制限は、/etc/hosts.allowと、/etc/hosts.denyで行う。hosts.allowファイルに書かれたIPアドレスからのアクセスは許可され、hosts.denyに書かれたIPアドレスからのアクセスを拒否することができる。もし、ローカルホストのIPアドレスが固定されているのであれば設定しておくといいだろう。たとえば、111.122.133.144/24のネットワークからのみSSHを利用したアクセスを許可する場合は、**リスト3・4**のように設定する。

すべての設定が完了したら、SSHサーバを次のようにして起動する。

```
/etc/init.d/sshd start
```

▼リスト2　/etc/ssh/sshd_config（抜粋）

```
# This is ssh server systemwide configuration file.

Port 22
    :
    :
# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes        ← #を付けてコメントアウト
PasswordAuthentication no          ← 追加
PermitEmptyPasswords no
    :
    :
```

▼リスト3　/etc/hosts.allow

```
sshd: 111.122.133.144/255.255.255.0
```

▼リスト4　/etc/hosts.deny

```
sshd: ALL: deny
```

## Column

### RSA暗号化アルゴリズム

RSA暗号は公開鍵と秘密鍵という対の鍵を使って、データの符号化と復号化を行なう暗号化アルゴリズムのひとつだ。

2つの鍵を用いる方式を公開鍵暗号方式と呼び、この方式には、RSA暗号のほかにもElGamal暗号や楕円曲線暗号など、いくつか存在する。

RSA暗号は、巨大な整数の素因数分解が難しいという事実を暗号解読の難しさに結びつけたアルゴリズムである。また、RSA暗号は解読が難しいというだけでなく、公開鍵を使ったデータの符号化と、秘密鍵を使った復号化を高速に行なえるという優れた特徴を持つ。RSA暗号がさまざまな場面で使われるのは、暗号としての強さと高速に計算できるという実用上の簡便さを兼ね備えているためだ。

さて、RSA暗号が基にしている素因数分解はどれほど難しいのだろうか？　たとえば、35は5と7の2つの素数に分けられるし、143は11と13に分けられる。このように、大きくない数を素因数分解するのは人間でもそれほど難しくない。しかし、39772916239307209103を2つの素数に分けろ言われたらどうだろうか。この大きな数字は625749337と6356046119に分けられる。普通の人が計算するのはちょっと無理だが、PCならば瞬時に計算できるレベルの数だ。

それでは高速なコンピュータがあれば、どんな数も素因数分解できるかといえばそんなことはない。素因数分解にかかる時間は、数の大きさに比例して増加するのではなくて、指数関数より少し低い割合で増加していく。

RSA暗号はこのアルゴリズムを考案した3人、Rivest、Shamir、Aldemanの名前の頭文字をとって名付けられている。この3人はRSA暗号を考えついた1977年に、RSA-129と呼ばれる129桁の整数の素因数分解問題を出した。このRSA-129は17年後に解読されたが、解読するのに17年間で約2000倍高速になったコンピュータを1600台必要とした。

このように素因数分解する数を大きくする、つまりRSA暗号で使う公開鍵のサイズを大きくすると解読は非常に難しくなる。コンピュータの計算速度が進化する度合いを考慮しても、150桁を超える数字はどうあがいても分解できないということになるようだ。

さて、暗号としてこれだけ強力で、プログラムとしても実装しやすいRSAだが、アルゴリズムの使用について特許権を持つRSA Securityへライセンス料を支払う必要があった。このためOpenSSHへの実装も見送られていた。しかし、特許行使権が切れる直前の昨年9月にRSA Securityが特許行使権を放棄し、誰でもRSAアルゴリズムを使えるようにパブリックドメインとして公開した。このため、OpenSSHを始め、オープンソースプログラムでもRSA暗号を使えるようになった。

このように強力なRSA暗号を使えるOpenSSHを使っていれば、外部からリモートログインする場合もクラックされる可能性はほとんどゼロだと言えるだろう。もっとも、秘密鍵が安全に保管されていることが前提であるが……。

または、

```
/etc/rc.d/init.d/sshd start
```

以上でSSHサーバの構築は完了だ。

なお、sshdに関するアクセス制限を変更した場合は、スーパーサーバではなく、SSHサーバを再起動する必要があるので注意してほしい。

## リモートログイン

SSHを使ってリモートログインするには、telnetコマンドの代替となるsloginコマンドを使用する。まずローカルホストにログインし、sloginコマンドに、リモートホスト名を指定して実行する。最初の接続時には、ホストキーが保存されていないので、「yes」と入力してホストキーを保存する。ホストキーは、フィンガープリント（指紋）とも呼ばれ、次回以降に接続する際に、接続するリモートホストが前回接続したホストと同じかどうかを検証するために利用される（つまり、偽証ホストではないかどうかが検証される）。このホストキーは.sshディレクトリにあるknown_hostsファイルに保存される。

リモートホストにログインする際には、鍵を作成した際に入力したパスフレーズを入力する。

認証に成功してリモートログインすれば、telnetやコンソールからログインしたときと同じようにマシンを利用することができる（**画面6**）。しかし、telnetコマンドとは異なり通信内容も暗号化されている。

正常に認証が行え、リモートログインできることが確認できたら、ntsysvコマンド（Turbolinuxではturbo serviceコマンド）で、システム起動時にsshdデーモンが起動できるように設定しておこう。

## リモートファイルコピー

SSHにはそのほか、rcpコマンドを代替する、ホスト間でファイルを安全にコピーできるscpコマンドが用意されている。scpコマンドを使えば、コピーするファイルの内容も暗号化されるので、ftpよりも安全にファイルをコピーすることができる。

scpコマンドの書式は**リスト5**のようになっている。

たとえば、fooという名前のファイルをリモートホストの/tmpフォルダにコピーする場合のscpコマンドは、次のように実行する。

```
$ scp foo hogehoge@ns1.ascii-linux.com:/tmp
```

前述したように、scpコマンドは暗号化されたコピーなので、ホスト間での鍵のコピーにも使用することができる。なお、「-r」オプションを指定すれば、ディレクトリごとコピーすることもできる。

## Windowsからのリモートログイン

前述したようにLinux同士であれば、sloginコマンドを使って安全にリモートログインできる。しかし、会社や学校からリモートログインする場合などはローカルホスト（クライアント）がWindowsの場合も多いだろう。

このような場合は、Windows用のSSHクライアントソフトを使用することで、SSHによるリモートログインが可能だ。ここでは、Windows用のフリーのtelnetクライアントとして有名なTera Term Pro（寺西高氏作）と、Tera Term ProをSSHに対応させるフリーのエクステンションであるTTSSH（Robert O'Callahan氏作）を組み合わせて利用する方法を紹介する。

Tera Term Proと、TTSSHは、**表4**のWebサイトで入手可能だ。

### ■インストール

まずは、Tera Term Pro（ttermp23.zip）と、TTSSH（ttssh154.zip）の2つをダウンロードして、ttermp23.zipを展開する。Tera Term Proにはインストーラ（setup.exe）が付属しているのでこれを実行すればインストールは簡単だ。メッセージに従ってインストールしよう。

Tera Term Proのインストールが完了したら、ttssh154.zipを展開し、

```
$ slogin ns1.ascii-linux.com
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host 'ns1.ascii-linux.com' added to the list of known hosts.
Enter passphrase for RSA key 'hogehoge@ns1.ascii-linux.com':
Last login: Sat Feb 24 22:09:02 2001 on tty1

$
```

sloginコマンドの実行
ホストキーが保存される
yesと入力
パスフレーズを入力
ログイン完了

画面6　sloginコマンドによるリモートログイン

▼リスト5　scpコマンドの書式

```
scp <ファイル名> <リモートユーザー名>@<リモートホスト名>：<リモートディレクトリ名>
scp <リモートユーザー名>@<リモートホスト名>：<リモートファイル名> <ディレクトリ名>
```

Tera Term Proをインストールしたフォルダにコピーする。

以上でインストールは完了だ。

■設定

WindowsからSSHでリモートログインする場合も、当然秘密鍵が必要となる。秘密鍵は先ほどFAT形式でフォーマットしたフロッピーディスクにコピーしておいたので、これを利用する。TTSSHで秘密鍵を利用するためにTera Term Proと同じフォルダにコピーしておこう。

Tera Term Proは通常、ttermpro.exeを実行することで起動するが、TTSSHを使う場合はttssh.exeを実行する。

ttssh.exeを実行すると**画面7**のように表示されるので、［キャンセル］をクリックして一度ダイアログボックスを閉じる。次に、メニューから［Setup］－［SSH Autentication］を選んで、「TTSSH: Authentication Setup」ダイアログボックスを開く。このダイアログボックスの、「User name」にリモートホストのアカウントを入力し、「Use RSA key to login」のラジオボタンをチェックする。さらに、［Private key file］をクリックして、先ほどコピーした秘密鍵を指定する（**画面8**）。秘密鍵を指定したら、［OK］ボタンをクリックしてダイアログボックスを閉じる。

次に、メニューから［Setup］－［SSH］を選んで、「TTSSH: Setup」ダイアログボックスを開く。このダイアログボックスで、［Read/Write file:］ボタンをクリックし、**画面9**のようにホストキーを保存するフォルダを指定する。設定が完了したら［OK］ボタンをクリックしてダイアログボックスを閉じる。

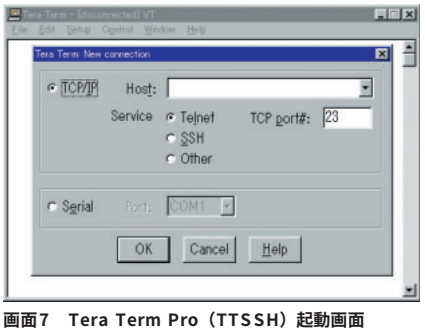
以上でSSHの設定は完了だ。それでは、実際に接続してみることにしよう。

メニューから［File］－［New connection］を選んで、「Tera Term New connection」ダイアログボックスを開く。このダイアログボックスの、「Host:」にリモートホストの名前を入力し、「Service」のSSHをチェックする（**画面10**）。設定が完了したら、［OK］ボタンをクリックする。最初の接続では、**画面11**のように、「SECURITY WARNING」ダイアログボックスが表示される。これはホストキーを保存するかどうかの確認だ。「Add this machine and its key to the known host list」をチェックして［Continue］ボタンをクリックする。これで、ホストキーが保存される。続いて、「SSH Authentication」ダイアログボックスの「Passphrase:」にパスフェーズを入力し（**画面12**）、［OK］ボタンをクリックする。以上で接続は完了だ。次回以降の接続時には、ホストキーを保存するかどうかを確認するダイアログボックスは表示されない。

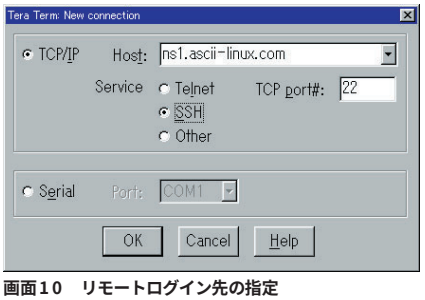
なお、会社や学校など、ファイアウォール内からSSHを使って接続する場合は、ポート22番が通過できるようになっていなければならないので注意してほしい。

| | |
|---|---|
| Tera Term Pro | http://hp.vector.co.jp/authors/VA002416/ |
| TTSSH | http://www.zip.com.au/~roca/ttssh.html |

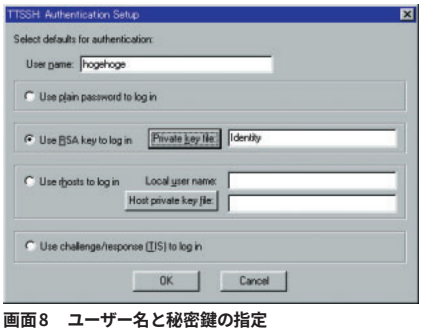
表4　ダウンロードWebサイト一覧



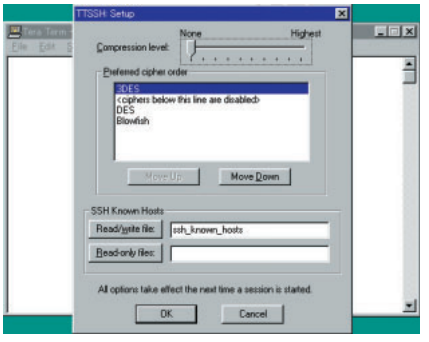

画面7　Tera Term Pro（TTSSH）起動画面



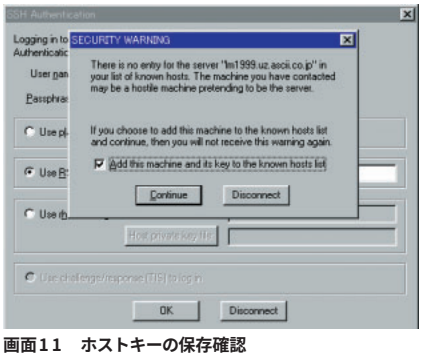

画面10　リモートログイン先の指定



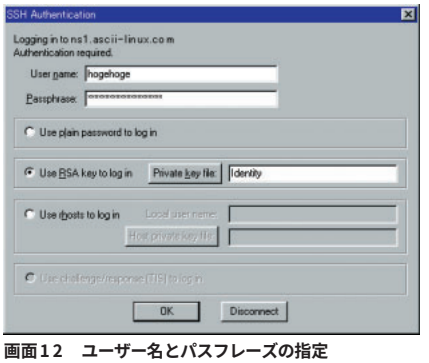

画面8　ユーザー名と秘密鍵の指定




画面9　ホストキーを保存するファイルの指定




画面11　ホストキーの保存確認

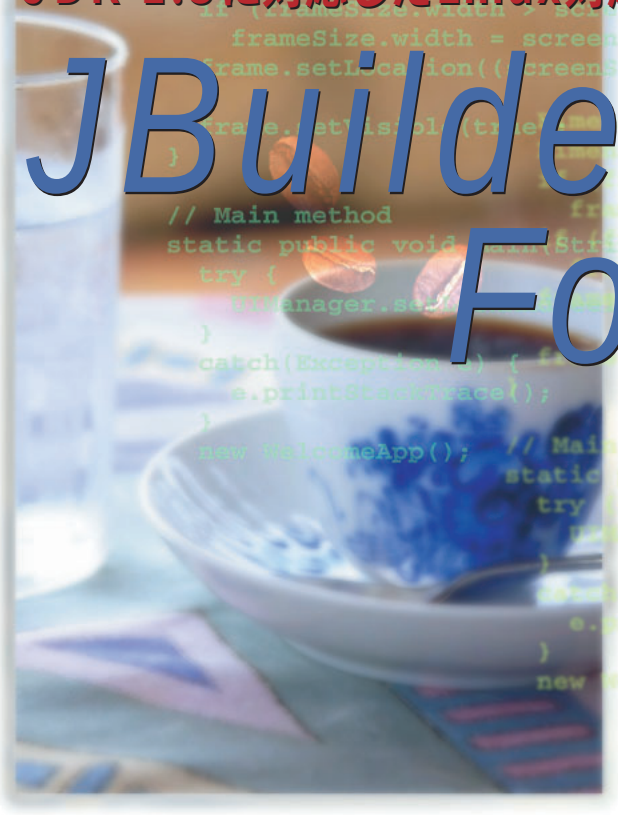



画面12　ユーザー名とパスフレーズの指定

短期集中連載

JDK 1.3に対応したLinux対応のJavaビジュアル開発ツール

# JBuilder 4 Foundation



## 第4回　データベースとの連携1

文：加藤大受
*Text：Daiju Kato　dkato@jcom.home.ne.jp*

**Java言語は非常に拡張性の高い言語で、携帯電話で稼働するような小さいサイズのプログラムから、メインフレーム上の基幹システムとの連携や、Web上のバーチャルショッピングモールなどのシステムまで幅広くサポートしています。今回はJava言語の持つデータベースとの連携について説明していきながら、Java言語の持つデータベースとの連携機能について解説します。**

データベースはさまざまな用途で活用されています。また、一部のアプリケーションではデータベースの存在は隠されています。たとえば、みなさんが年末に使用する年賀状印刷ソフトなどでは、住所録の部分を何らかのデータベースで管理しています。テキストファイルなどを使用することでもデータは管理できますが、格納されているデータを検索・置換するには、いったんすべてのデータを読み込む必要があるだけでなく、データ数の増加につれてパフォーマンスが悪化していきます。ある程度のデータを管理していくのであれば、やはりデータベースの利用が必須になります。

それではデータベースとは何なのでしょうか？　細かい説明はここでは書きませんが、データベースにはスタンドアローンでの利用を想定したデスクトップ型のデータベースと、大規模な利用を想定したリレーショナルデータベース（RDB）の大きく2種類に分かれます。もともとデスクトップデータベースはリレーショナルデータベースと比べ、軽く、操作性がいいものが多かったのですが、最近は軽いリレーショナルデータベースも登場しており、徐々にこの垣根がなくなってきています。これからデータベースについて学ぼうと思われる方は、リレーショナルデータベースを利用されることをお勧めします。

データベースは単なるデータを格納する器となります。どのようにデータを格納するのか、また、それらのデータをどうやって活用していくのかはプログラマー側が決定していくことになります。データベースは単に与えられた条件に従い、データの追加・変更・検索などを行います。

どのようにデータを格納していくかは、スキーマ構造といわれるテーブルの構造、インデックスの構造によって変わってきます。このスキーマを決定する作業をスキーマ設計と呼び、データベースを使用するシステムを設計するうえで一番重要な作業となります。

スキーマ設計については詳細に解説されている書籍が多数存在しますので、そちらを参照してみるとよいでしょう。スキーマの構造によってシステムの拡張性、パフォーマンスなどが変わってきますので、本格的にシステムの構築を検討されている方はスキーマ設計に関する知識を身につけておくといいでしょう。

### プログラムからデータベースへのアクセス

プログラムからデータベースにアクセスする方法はさまざまです。たとえば、Visual Basicなどを利用されたことがある方は、ODBCドライバを使用

してデータにアクセスする方法などを思い浮かべるでしょう。また、C言語などを利用している方は、プリプロセッサを利用する埋め込みSQLを使用しているかもしれません。それではJava言語の場合はどうでしょうか？　Java言語からデータベースにアクセスするには次の2通りがあります。

- **JDBCドライバを使用する**
- **SQLJを使用する**

JavaにはJDBC APIというものが用意されており、このAPIによってさまざまなJDBCドライバが使用可能になります。JDBC APIは次のようなことを行います。

- **Javaプログラムからデータベースサーバへ接続する**
- **SQL文を組み立てて、データベースサーバで実行させる**
- **データベースサーバが処理した結果を取り出す**
- **データベースの情報、処理結果に関する情報などを取り出す**

つまり、JDBC APIはJDBCドライバを使用するためのインターフェイスを提供しており、Javaプログラムと JDBCドライバを結びつけているのです。

JDBCドライバにはさまざまな種類があり、それぞれタイプ別に分類されます。

- **タイプ1 ODBCブリッジドライバ**
- **タイプ2 ネイティブAPI Partly-Javaドライバ**
- **タイプ3 ネットプロトコルAll-Javaドライバ**
- **タイプ4 ネイティブプロトコルAll-Javaドライバ**

各ドライバのタイプは**図1**のようにデータベースにアクセスします。

それぞれのタイプを簡単に説明すると次のようになります。

### ■タイプ1 ODBCブリッジドライバ

既存のODBCドライバをラップして使用することのできるドライバです。タイプ1のドライバは特殊なJDBCドライバを使用せずに利用できますが、クライアントに存在するODBCドライバにアクセスしますので、使用できるプラットフォームが限られてしまいます。ほとんどの場合、Windowsプラットフォームに限られます。また、JavaプログラムからODBCドライバを呼び出すので、クライアント側の負荷が高いだけでなく、パフォーマンスの問題も発生しやすいのであまり実用的ではありません。 JDBC-ODBCブリッジドライバなどはJDKの中に提供されているので、ODBCが使用できる環境があれば簡単に使用することができます。

### ■タイプ2 ネイティブAPI Partly-Javaドライバ

Microsoft SQL ServerのDB-Libraryや、OracleのOCIなど、クライアント側のAPIを提供するライブラリの一部を呼び出すように実装されたJDBCドライバです。JDBCドライバからこれらのネイティブインターフェイスを呼び出しますので、パフォーマンスは非常に優れているのですが、クライアント側へのセットアップが必要になるなど、Javaの持つ本来のマルチプ

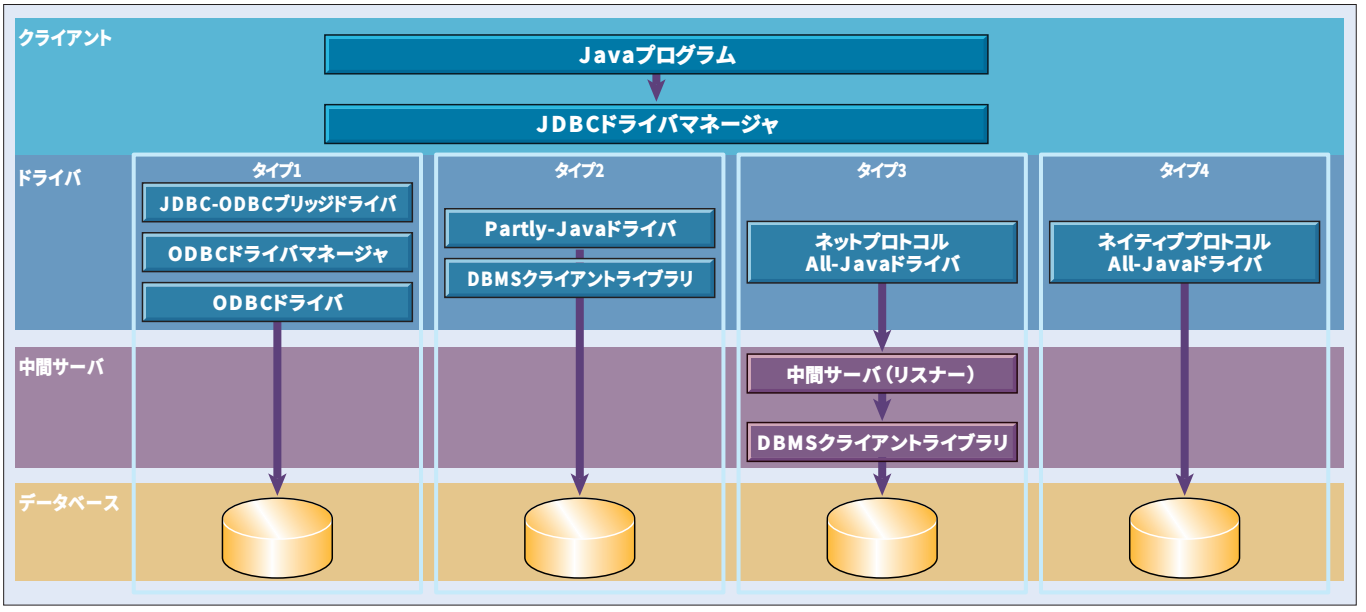



図1　JDBCドライバのタイプ

ラットフォーム性を活かすことができません。また、クライアント側のモジュールを呼び出すことになるので、アプレットで使用することもできません。このタイプは、Javaアプリケーションで大量のデータ処理を行う場合に向いています。

### ■タイプ3 ネットプロトコル All-Javaドライバ

InterBase用のInterClientなど、ネイティブのJDBCドライバから中継サーバを経由し、データベースサーバに接続します。中継サーバを使用することで、JDBCドライバをコンパクトに作成することができ、アプレットでも使用可能です。中継サーバとデータベースサーバとは、ネイティブなAPIを使用してデータ処理を行うため、パフォーマンスも非常に優れています。

中継サーバはデータベースサーバと同じマシンに置かれることが多く、アプレットで使用する場合、データベースサーバ、中継サーバ、Webサーバが同一のマシンに置かれるため、サーバの負荷が高くなる傾向があります。また、データベースサーバが持っている機能であっても、中継サーバがそれらの機能をサポートしておらず、機能が制限されることもあります。

### ■タイプ4 ネイティブプロトコル All-Javaドライバ

クライアントAPIライブラリと、ネットワークライブラリすべてをJDBCで実装して、DBMSと直接データ通信を行う形態がタイプ4のJDBCドライバになります。

このJDBCドライバは、かなり重い実装になってしまいますが、データベースサーバに用意されているすべての機能が使用できる利点があります。

使用するデータベースサーバによって、用意されているJDBCドライバのタイプが異なりますので、JDBCドライバを使用するときはタイプを確認してから使用するようにしてください。

Java言語は、JDBCドライバとJDBC APIという仕組みによって、データベースサーバの違いを吸収しやすくしているだけでなく、Javaの持つマルチプラットフォーム性を活かせるように、データベースとの接続も考慮されています。このため、大規模な基幹系システムでも使用できるのです。ただし、JDBC APIが用意されているのはデータベースとのやりとりの部分です。受け取ったデータを表示するGUIなどは、標準では用意されておりません。JBuilderのProfessional版以上や、VisualAge、VisualCafeなどのJava開発ツールでは、受け取ったデータを簡単に表示できるJFC/Swingを拡張したGUIコンポーネントが用意されています。興味がある方はこれらのツールを評価してみるといいでしょう。

Javaプログラムとデータベースとの接続を行うもうひとつの方法はSQLJです。SQLJはJavaプログラムから静的埋め込みSQLを使用するための仕様で、データベースの業界標準であるSQL99で正式に採用されているものです。SQLJは静的埋め込みSQLの実行を提供するので、サーバでの負荷が軽減し、Javaプログラムによる大量のクライアント処理やトランザクション処理の実現を可能にしますので、エンタープライズレベルで非常に重要性が高いものです。

JDBCとSQLJの使い分けを簡単に説明すると、動的なSQLを使用したい場合はJDBCで、静的なSQLでパフォーマンスを重視する場合はSQLJということになります。SQLJはまだまだ新しい規格ですので、対応しているデータベースサーバは限られていますが、信頼性に優れたJavaプログラムを作成できる技術として注目されています。

## JDBCを利用したデータ検索

JDBCの説明はここまでにして、実際にJDBCを使用してデータ処理を行ってみたいと思います。ここではもともと商用のデータベースで、新しいバージョンからオープンソース化されたInterBaseを使用します。InterBaseについては本誌に連載されている筆者の「InterBase 6.0」で説明しておりますので、InterBaseの概要は割愛します。

## InterClientとは

InterClientはInterBase用のJDBCドライバです。タイプ3のドライバで、InterServerと呼ばれる中間サーバを経由して、InterBaseサーバと通信を行います。InterClientは100% Pure Javaで書かれたJDBCドライバで、非常に軽いドライバのため、アプレットで使用することも可能です。InterClientは中間サーバであるInterServerと、ポート番号3060を使用して通信を行います。通常、InterBaseの通信は3050を使用していますので、まったく別の通信であることがここからもわかります（**リスト1**）。

それではInterClientをインストールしてみましょう。InterClientはBorland SoftwareのWebサイトから入手することができます。なおInterClientには、JDBC 1.0規格に準拠したInterClient 1.6と、JDBC 2.0規格に準拠したInterClient 2.0（ベータ版）の2種類が存在します。また、ソースコードも同様に同社のWebサイトからダウンロードできます。

今回は、InterClient 2.0を使用してみたいと思います。インストールの方法については、**画面1**を参照してください。シェルスクリプトで書かれたインストールプログラムが用意されており、このシェルプログラムを起動することで、services、inetd.confなどの環境設定も自動で行ってくれます。

インストールが終了したら、InterClientに含まれているサンプルファイルを使用して、InterClientが正しく動作しているか確認してみましょう。

コマンドラインでJavaを使用しますので、パスとクラスパスの設定を行い、X Window Systemを起動します。X Window System起動後、ターミナルを起動してコマンドラインからサンプルファイルを起動します。クラスパスが正しく設定されていないとエラーになりますので、起動できない場合はクラスパスを再度確認してみてください。

サンプルプログラムが起動したら、ホスト名、データベースファイルのパス、ユーザー、パスワードを入力して［TEST］ボタンをクリックします。InterClientが正しくインストールされている場合は、**画面2**のように、「No Installation problems detected!」と表示され、接続先のデータベースの情報が表示されます。

## InterClientを使用したプログラムの作成

InterClientのサンプルプログラムが正しく動作したので、それでは実際にJBuilder 4 Foundationを利用してInterBaseのデータをアクセスするプログラムを作成してみましょう。JBuilder 4 FoundationでJDBC 2.0のAPIを使用するには追加のオプショナルパッケージが必要となります。プロジェクトを作成する前に、オプショナルパッケージをSunのWebサイトからダウンロードします。ダウンロードしたファイルはJBuilder 4 Foundationをインストールしたディレクトリのjbuilder4/lib/ext/にコピーしてください。また、このパッケージを使用するには、ライブラリの設定が必要となります。

ライブラリの設定は、［ツール］－［ライブラリ設定］で表示される［ライブラリ設定］ダイアログボックスで、［新規］のボタンをクリックして、「新規ライブラリウィザード」を起動します。ここで、**画面3**のように名前を入力し、場所のリストボックスから「JBuilder」を選択します。続いて、［追加］ボタンをクリックして、先ほど

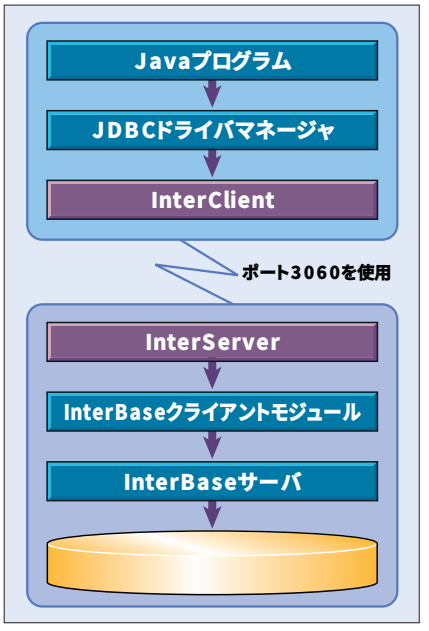


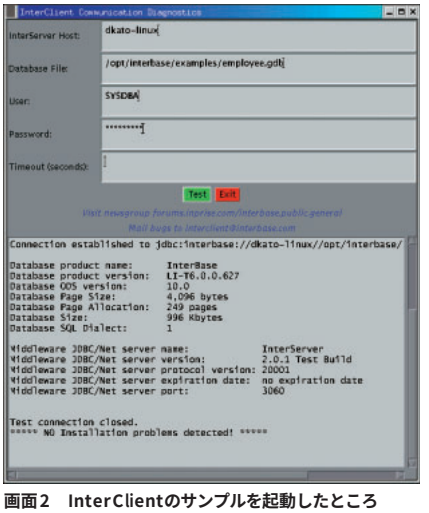

図2　InterClientの構造

▼リスト1　servicesファイルの内容

```
interserver     3060/tcp        # InterBase InterServer
gds_db          3050/tcp        # InterBase Server
```

```
# tar zxvf IC20001LinuxJRE12.tar.gz          ← InterClient 2.0の解凍
# cd interclient_install_temp_dir
# sh install.sh                              ← インストールプログラムを実行
# export PATH=/home/jbuilder4/jdk1.3/bin:$PATH   ← コマンドラインでJavaを起動するためにパスを設定
# export
CLASSPATH=/home/jbuilder4/jdk1.3/jre/lib/rt.jar:/usr/interclient/interclient.
jar:$CLASSPATH                               ← コマンドラインでJavaを起動するためにクラスパスを設定
# startx                                     ← Xが起動していなければ起動
# cd /usr/interclient                        ← InterClientのディレクトリに移動
# java interbase.interclient.utils.CommDiag  ← サンプルファイルを起動
```

画面1　InterClient 2.0のインストール




画面2　InterClientのサンプルを起動したところ

| | |
|---|---|
| InterClientのダウンロードサイト | http://www.borland.com/interbase/downloads/ |
| JDBC 2.0オプショナルパッケージのダウンロードサイト | http://java.sun.com/products/jdbc/download.html#spec |

表1　URL一覧

extディレクトリにコピーしたjdbc2_0-stdext.jarを選択します。ウィザードを終了すると**画面4**のようにライブラリに設定されます。

同様に、InterClient 2.0を登録します。再度、「新規ライブラリウィザード」を起動し、今度は場所のリストボックスで「ユーザーホーム」を選択し、追加ボタンをクリックして、/usr/interclient/interclient.jarを登録します（**画面5**）。これで、InterClient 2.0がJBuilder 4 Foundationで使用できるようになりました（**画面6**）。

それでは新規のプロジェクトを作成して、InterBaseを利用したプログラムを作成してみましょう。［ファイル］－［新規プロジェクト］を選択し、プロジェクトウィザードを起動します。プロジェクト名に「DBSample」と入力し、ステップ2/2に移ります。必須ライブラリで［追加］ボタンをクリックし、先ほど登録した「JDBC 2.0 Optional Package」と「InterClient 2.0」を選択します。これで、今回作成するプロジェクトでこれらのライブラリが使用できるようになります。

アプリケーションクラス名は「DBSampleApp」、フレームクラス名は「DBSampleFrame」とします。

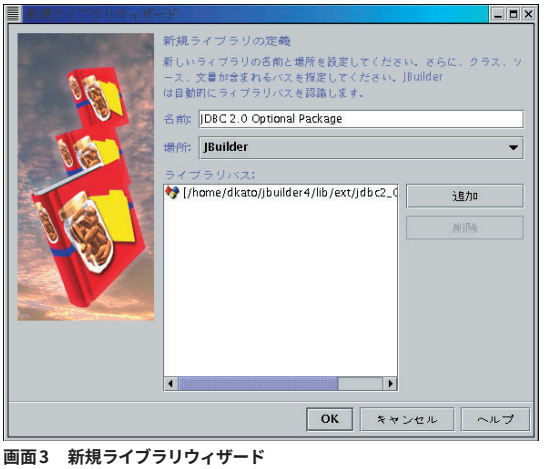
設定が終了したらプロジェクトウィザードを終了し、GUIの設計に移ります。ビジュアルデザイナを起動し、レイアウトを「GirdLayout」に変更し、gridlayout1のプロパティをインスペクタで、「Column」プロパティを「2」に、「Row」プロパティを「4」に変更します。

**画面7**のように、Labelオブジェクトを3つ、TextFieldオブジェクトを3つ、JButtonを2つ配置します。JButtonオブジェクトは、次へ移動するボタンの名前を「nextButton」に、先頭に移動するボタンの名前を「firstButton」に変更します。

エディタ画面に戻り、InterBaseへの接続部分を記述します。まず、必要なパッケージを追加します。必要になるパッケージは**リスト2**の通りです。

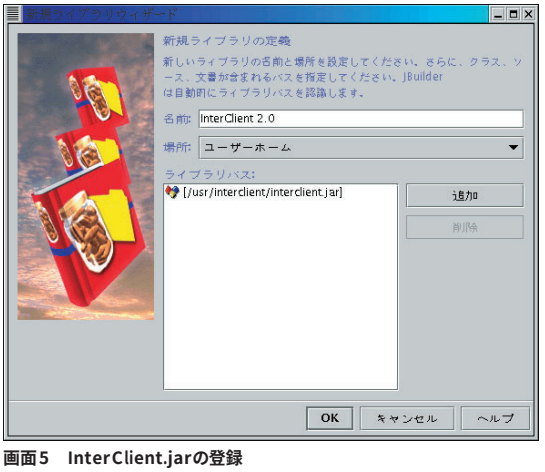
パッケージを追加したら、変数を定義します。jbInit()メソッドの末尾に**リスト3**の内容を追加します。

接続時に使用するオブジェクト、結果セットを受け取るオブジェクト、そして接続するデータベースの情報などを格納するオブジェクトを定義します。

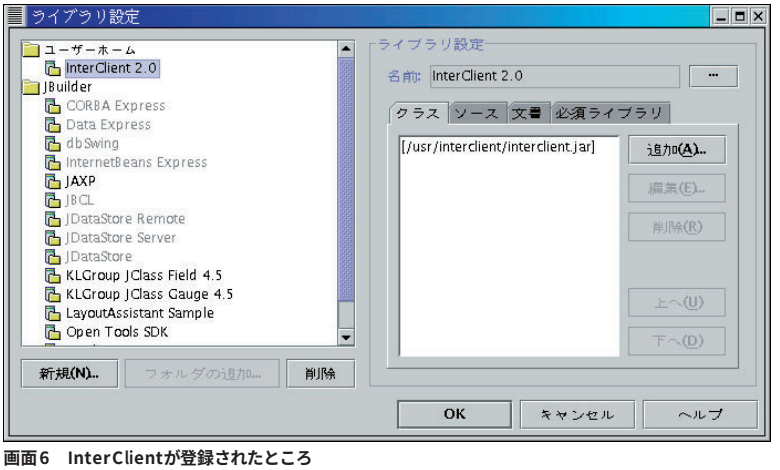



**画面3　新規ライブラリウィザード**


**画面4　ライブラリに登録**




**画面5　InterClient.jarの登録**




**画面6　InterClientが登録されたところ**

続いて、InterBaseへの接続を行うICInit()メソッドを作成します。メソッド化することで、汎用性を高めるようにします。

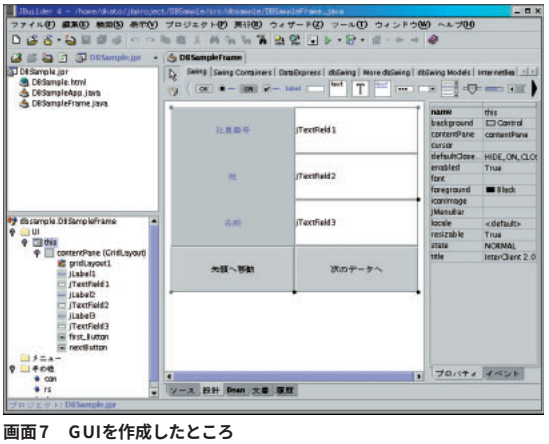
まず、サーバ名、接続するデータベースのパスを指定し、続いてユーザ名とパスワードをパラメータにしてInterBaseサーバへ接続します。続いて、SQL文を発行する準備を行い、SQL文の実行を行います。結果セットを受け取る場合は、executeQuery()メソッドを使用します。

結果セットを受け取ったら、nextButtonを呼び出し、受け取ったデータをGUIに表示するようにします。ICInit()メソッドのソースは**リスト4**のようになります。メソッドになっていますので、**リスト5**のように呼び出すことを忘れないようにしてください。

最後に、ボタンを押したときの処理を記述します。nextButtonボタンを押したときは、単純に次のレコードを呼び出すnext()メソッドを呼び出します（**リスト6**）。ただし、末尾の場合がありますので、if文を使用して必ず確認を行います。また、first_buttonボタンを押したときは、再度クエリを実行し、先頭データを表示します（**リスト7**）。

残念ながら、InterClient 2.0にはprevious()メソッド、first()メソッドがありませんので、前の行に戻る処理は自分で記述する必要があります。JDBC 2.0の規格にはこれらの逆方向へのカーサー操作のメソッドが用意されています。InterClient 2.0にも今後これらのメソッドが追加されていくでしょう。

ボタンの処理が終わったら、実行して動作を確認してみましょう。正常にデータが受け取れていると思います。



画面7　GUIを作成したところ

## JDBC 1.0とJDBC 2.0

JDBC 1.0とJDBC 2.0では、データベースサーバとの接続方法が異なります。もちろん、JDBC 1.0の接続方法で、JDBC 2.0に対応しているJDBCドライバを使用して接続することは可能です。ここで、少し違いを説明しながら、先ほど作成したプログラム内で使用したJDBC APIについて解説していきましょう。

JDBCドライバを使用するには、ドライバをロードする必要があります。JDBCドライバによって若干ロードの

▼リスト4　ICInit()メソッド

```
//InterBaseへの接続を行うICInit()メソッド
private void ICInit() throws Exception {
  //InterClientドライバの定義
  try {
      //サーバー名の設定
      ds.setServerName("dkato-linux");
      //パスの設定
      ds.setDatabaseName("/opt/interbase/examples/employee.gdb");
      //InterBaseに接続を実施(ユーザー名、パスワード)
      con = ds.getConnection("SYSDBA","masterkey");
      //SQL文の発行の準備
      java.sql.Statement stmt = con.createStatement();
      //SQL文の実行
      rs=stmt.executeQuery("SELECT * FROM EMPLOYEE");
      //GUIに先頭のデータを表示
      nextButton.doClick();
      }
  catch (SQLException ex){
     System.out.println("\n**************接続エラー発生\n**************");
      while (ex !=null) {
        System.out.println("エラーメッセージ: "+ex.getMessage());
        ex = ex.getNextException();
      }
  }
}
```

▼リスト2　パッケージを追加

```
import interbase.interclient.*;
import javax.swing.*;
import java.sql.*;
import java.io.*;
import javax.sql.*;
```

▼リスト3　変数の定義

```
jbInit()の末尾に追加
  //InterBaseへの接続に必要な変数
  java.sql.Connection con = null; //データベースとの接続用
  java.sql.ResultSet rs = null;   //結果セットの格納
  interbase.interclient.DataSource ds =
       new interbase.interclient.DataSource();  //データベース情報を格納
```

仕方が違いますが、Driverクラスのインスタンスを作成し、使用するドライバを宣言します。

InterClientを使用する場合は、InterClientのドライバクラスであるinterbase.interclient.Driverをパラメータに渡します（**リスト8**）。JDBC 2.0の場合では、DataSourceクラスという新しいクラスが用意されており、こちらのクラスを使用してJDBCドライバをロードすることができます（**リスト9**）。

JDBC 2.0ではJNDIという、ディレクトリサービスをJavaから使用する機能が用意されています。JNDIでは、接続するデータベースサーバの情報を、ディレクトリサービスに登録することができます。

また、Enterprise JavaBeans（EJB）や、JavaBeansでJDBCドライバを通じてデータベースサーバにアクセスすることを想定した使用方法がサポートされています。

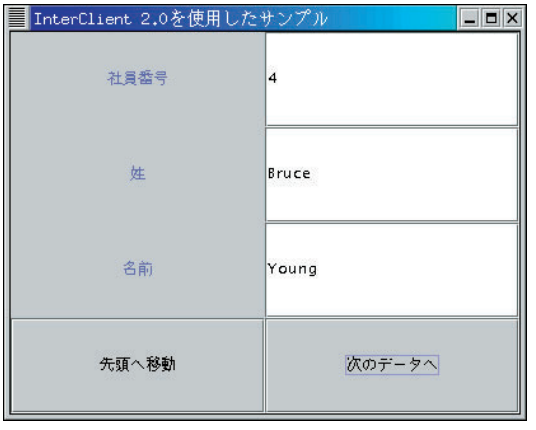
JDBCドライバのロードが終了したら、データベースサーバへの接続を行います。接続先のデータベースの情報は「jdbc://＜ドライバ名＞/＜パス＞」といった形の、URL形式で指定します。接続時にユーザー名、パスワードなどの情報が必要になります。ユーザー名などの情報は、Propertiesクラスを利用します（**リスト10**）。JDBC 2.0では、DataSourceクラスのメソッドにサーバ名、データベース名などを管理するプロパティが追加されていますので、メソッドを通じてこれらの内容を設定します。また、日本語を使用する場合は、キャラ

**▼リスト5　ICInit()メソッドを呼び出す**

```
/**フレームの構築*/
public DBSampleFrame() {
  enableEvents(AWTEvent.WINDOW_EVENT_MASK);
  try {
    jbInit();
    //InterBaseへの接続
    ICInit();
  }
  catch(Exception e) {
    e.printStackTrace();
  }
}
```

ICInit()メソッドを呼び出す

画面8　実行したところ




**▼リスト6　nextButtonを押したときの処理**

```
void nextButton_actionPerformed(ActionEvent e) {
//次のデータを取得する
   try {
       if (rs.next()) {
           jTextField1.setText(
                rs.getString("EMP_NO"));
           jTextField2.setText(
                rs.getString("FIRST_NAME"));
           jTextField3.setText(
                rs.getString("LAST_NAME"));
        }
   } catch (java.sql.SQLException ex) {
       System.out.println(
                "\n**************SQL Error\n*************");
       while (ex !=null) {
        System.out.println("SQLState:
            "+ex.getSQLState());
        System.out.println("Message:
            "+ex.getMessage());
        ex = ex.getNextException();
       }
       System.out.println(" ");
   }
}
```

**▼リスト7　first_Buttonを押したときの処理**

```
void first_Button_actionPerformed(ActionEvent e) {
 //先頭に戻る
    try {
        //SQL文の発行の準備
        java.sql.Statement stmt =
                con.createStatement();
        //SQL文を再実行
        rs=stmt.executeQuery("SELECT * FROM
                EMPLOYEE");
        nextButton.doClick();
    } catch (java.sql.SQLException ex) {
        System.out.println(
                "\n**************SQL Error\n*************");
        while (ex !=null) {
         System.out.println("SQLState:
                "+ex.getSQLState());
         System.out.println("Message:
                "+ex.getMessage());
         ex = ex.getNextException();
        }
        System.out.println(" ");
    }
 }
```

クタセットの指定を行い、データベースと同じキャラクタセットを指定して接続する必要があります。InterClientではキャラクタセットの指定をcharSetというプロパティで指定します。データベースへの接続はConnectionクラスを利用します（**リスト11**）。

接続が完了したら、SQLを格納するための入れ物（コンテナ）を準備します。JDBCにはSQLのコンテナとして次の3種類のオブジェクトが用意されています。

- **Statementオブジェクト**
  パラメータを持たない、単純なSQLで使用
- **PreparedStatementオブジェクト**
  入力パラメータを持つSQLで使用。繰り返し使用するSQLなどに向いている
- **CallableStatementオブジェクト**
  ストアドプロシージャの実行に使用

SQL文の実行ですが、SELECT文を実行して結果セットを受け取る場合と、INSERT、UPDATE文を使用する場合とでは、使用するメソッドが異なります。SELECT文の場合はexecuteQueryメソッドを使用して結果セットを受け取り（**リスト12**）、INSERT、UPDATE文ではexecuteUpdateメソッドを使用してエラーステータスを受け取ります（**リスト13**）。

結果セットの取り出しですが、行単位に取り出します。行の移動には、next()メソッドを使用します。受け取るデータが文字列ならgetString()メソッド、数値データならgetInt()メソッドを使用してデータを取得します（**リスト14**）。列名をパラメータにすることもできますし、列番号を指定してもデータを取得することができます。

最後にデータベースへのアクセスが終了したら、生成したオブジェクトを解放します（**リスト15**）。すぐにデータベースに再接続するかどうかによって、どのオブジェクトを解放するかを考慮してください。データベースとの接続を解放してしまうと、次回アクセスする場合も接続から行う必要があり、接続に時間がかかりますので注意してください。

**▼リスト8　JDBC 1.0の場合**

```
//Driverクラスを使用してInterClientをロード
java.sql.Driver d=(java.sql.Driver) Class.forName(
        "interbase.interclient.Driver").newInstance();
```

**▼リスト9　JDBC 2.0の場合**

```
//DataSourceクラスを使用してInterClientをロード
interbase.interclient.DataSource ds =
          new interbase.interclient.DataSource();
```

**▼リスト10　JDBC 1.0の場合**

```
//ユーザー名などの情報を管理するオブジェクトを宣言
java.util.Properties prop = new java.util.Properties();
//接続するURLの宣言
String url =
  "jdbc:interbase://dkato-linux/opt/interbase/examples/employee.gdb";
prop.put("user","SYSDBA");            //ユーザー名の設定
prop.put("password","masterkey");     //パスワードの設定
prop.put("charSet","SJIS");       //キャラクターセットの指定

//InterBaseサーバに接続
java.sql.Connection con =
         DriverManager.getConnection(url,prop);
```

**▼リスト11　JDBC 2.0の場合**

```
ds.setServerName("dkato-linux");        //サーバ名の設定
//パスの設定
ds.setDatabaseName(
        "/opt/interbase/examples/employee.gdb");
//キャラクタセットの指定（SJISの場合）
ds.setCharSet(
        interbase.interclient.CharacterEncodings.SJIS);
//InterBaseに接続を実施(ユーザー名、パスワード)
java.sql.Connection con =
        ds.getConnection("SYSDBA","masterkey");
```

**▼リスト12　SELECT文の場合**

```
java.sql.Statement stmt = con.createStatement();
//SELECT文の実行
java.sql.ResultSet rs =
     stmt.executeQuery("SELECT * FROM EMPLOYEE");
```

**▼リスト13　INSERT文の場合**

```
java.sql.Statement stmt = con.createStatement();
//INSERT文の実行 エラーステータスが戻ってくる
int error=stmt.executeUpdate(
  "INSERT INTO EMPLOYEE (EMP_NO) VALUES (10)");
```

**▼リスト14　結果セットの取り出し**

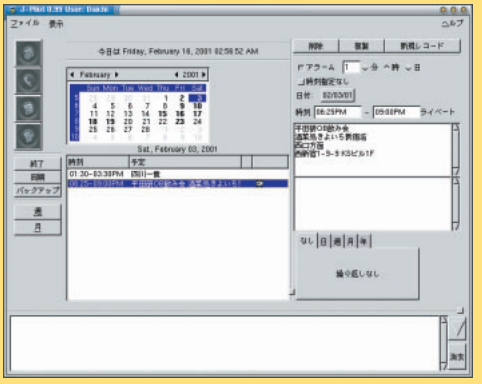
```
int iEmployee_No = rs.getInt("EMP_NO"));        //数値型の場合
String sFirst_Name =rs.getString("FIRST_NAME");//文字型の場合
```

**▼リスト15　データベースオブジェクトの解放**

```
rs.close();       //結果セットのオブジェクトをクローズ
stmt.close();     //SQLコンテナのクローズ
con.close();      //データベースとの接続をクローズ
```

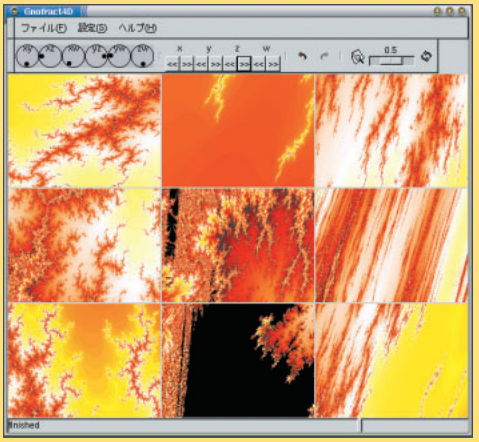
# Free Application Showcase

文：出井 一
*Text : Hajime Dei*



J-Pilot　P.134


Rocks'n'Diamonds　P.142



Gnofract 4D　P.144



※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パネルにメモやニュースなどを表示

# MemoPanel／HtmlHeadLine.sh

バージョン：7.6／29.2　　ライセンス：GPL

http://kano.technolust.cx/memopanel/memopanel2.php3
http://kano.technolust.cx/hhl/（HtmlHeadLine.sh）

MemoPanelは、GNOMEパネルにメモやヘッドラインニュースを表示するアプレットだ。ヘッドラインニュースの取得を行う「HtmlHeadLine.sh」との組み合わせで威力を発揮する。日本語のメモも問題なく入力可能だ。一定時間ごとに表示を更新する機能を利用してカレンダーや時計代わりに使えるほか、メールチェックや画面キャプチャなどの機能も備えている。動作にはGNOMEが必要だ。

### ●ビルドから起動まで

MemoPanelは、ファイル一式をtar＋gzip（またはbzip2）したtarボールのみ配布されている。configureスクリプトは用意されていないが、通常はそのまま「make」とすればビルドできるはずだ。

インストールは、「su -」としてrootになってから「make install」とすればいい。

ktermなどのコマンドラインで「memopanel&」と入力するか、GNOMEパネル上で右クリックし、ポップアップメニューの[アプレット]－[ユーティリティ]－[MemoPanel]を選択すると、MemoPanelが起動して、起動した日時がメモとして表示される（**画面1**）。表示内容の変更方法は次節で説明する。

上記の手順でMemoPanelを複数起動して、パネル上に複数のメモを置くことも可能だ。メモの内容などはGNOME終了時に自動的に保存され、次回起動時に再現される。メモを取り除くには、メモ上で右クリックし、ポップアップメニューの[パネルから取り除く]を選択すればいい。


画面1　GNOMEのパネルに日付と時間を示すメモが表示される。

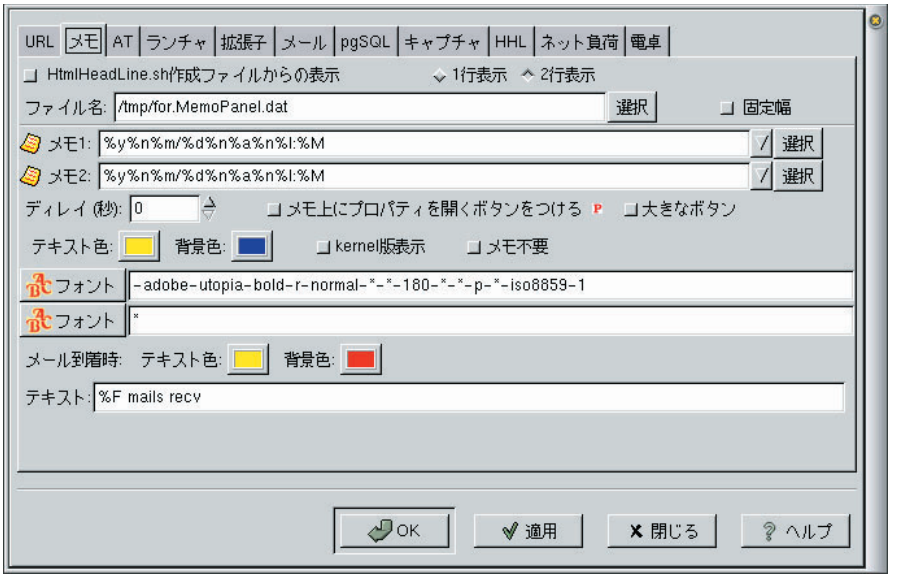
### ●メモの内容を設定する

最初に、メモの内容を設定する方法を説明しよう。メモ上で右クリックし、ポップアップメニューから[プロパティ]を選択すると、プロパティのダイアログが開く。MemoPanelの多機能さを反映して設定項目は多岐にわたっており、おおまかなジャンルによってタブ付きページで分類されている。まずは、基本となる[メモ]ページの内容を変更しよう（**画面2**）。

メモとして表示したいテキストは、[メモ1]に設定する。もちろん、日本語をそのまま入力可能だ。ただし、複数行にわたるメモを書くには、改行したい位置でEnterキーを押すのではなく、「%n」と書く必要がある。たとえば「13:30%n四川一貫」と書くと、1行目が「13:30」、2行目が「四川一貫」というメモが表示される。

メモの行数を増やすと、パネルの幅が大きくなってしまうため、長いメモは2つに分割して、一定時間ごとに切り替えて表示するといい。具体的には、[メモ1]と[メモ2]にそれぞれテキストを書き、[ディレイ（秒）]に表示時間を設定する。なお、ディレイが0（初期値）のまま、[メモ2]にテキストを設定すると、[OK]ボタンを押すたびにメモの表示が切り替わってしまうので気をつけよう。

このほか、メモに使われる文字色や背景色、フォント（英語・日本語）などを変更できる。メモの内容と同様、こうした設定もそれぞれのメモごとに独立しているので、重要なメモを目立つ色で表示するといった使い分けを行うとよいだろう。




画面2　プロパティの[メモ]ページでメモの内容などを設定。

### ●現在の日付・時刻を表示する

MemoPanelの起動時には、**画面1**のように作成（あるいは更新）日時が表示される。このとき、プロパティでメモのテキストを確認すると、実際の日時ではなく「%y%n%m/%d%n%a%n%l：%M」という文字列が設定されている。

これらは、dateコマンドで用いられるstrftime形式の文字列で、改行を表わす「%n」もその一部だ。MemoPanelで有用と思われるstrftimeをリスト1に抜粋した。

たとえば、メモを書く際にいちいち現在時刻を調べないでも、「%r 笹北さんから電話…」などと書いておけば、「10：31：48 AM 笹北さんから電話…」のように、メモを作成した時刻（12時間制）に自動的に置き換えてくれる。ただし、メモの内容を更新すると、自動的に時刻も更新されてしまうので気をつけよう。

| (1) 時刻関連 | |
|---|---|
| %H | 24時間制の時間（00-23） |
| %k | 同上（0-23） |
| %I | 12時間制の時間（01-12） |
| %l | 同上（1-12） |
| %M | 分（00-59） |
| %S | 秒（00-61） |
| %p | ロケール対応のAM／PM表記 |
| %r | 12時間制の時刻 |
| %T | 24時間制の時刻 |
| %X | ロケール対応の時刻表記 |
| %Z | タイムゾーン |
| **(2) 日付関連** | |
| %y | 下2桁の年（00-99） |
| %Y | 4桁表記の年（1970-） |
| %m | 月（01-12） |
| %b | ロケール対応の月名（省略形） |
| %B | ロケール対応の月名（完全形） |
| %d | 日（01-31） |
| %a | ロケール対応の曜日名（省略形） |
| %A | ロケール対応の曜日名（完全形） |
| %J | 日本語の曜日名（※独自拡張） |
| %D | 日付 |
| %x | ロケール対応の日付表記 |
| %c | ロケール対応の日付・時刻表記 |
| **(3) その他** | |
| %n | 改行 |
| %% | %自身 |

リスト1　strftime形式の文字列（一部抜粋）

この機能を、一定時間ごとにメモの表示を更新するディレイ機能と組み合わせると、MemoPanelをカレンダーや時計アプレットの代わりとして使えるので便利だ。

たとえば、[メモ1]に「%x%n%r」、[メモ2]は空欄、[ディレイ（秒）]を1に設定すると、1秒ごとに表示が更新され、そのたびにメモの内容である日付や時刻も更新される（**画面3**）。

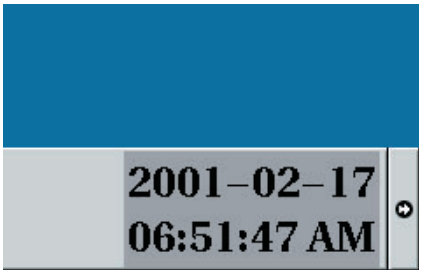
strftimeの組み合わせを変更して自分好みの表示にしてもいい。また、[メモ1]に日付、[メモ2]に時刻のstrftimeをそれぞれ設定すれば、日付と時刻が1秒ごとに切り替わる。

### ●メールチェッカーとして使う

MemoPanelをメールチェッカーとして利用すると、指定したメールサーバを一定時間ごとにチェックして、到着したメールの数をメモ上に表示する。さらに、指定したメールクライアントを起動して、新着メールを受信させることも可能だ。

設定は、プロパティの[メール]ページで行う（**画面4**）。メールサーバ（POP3／IMAPに対応）とユーザーID、パスワード、チェック間隔（秒単位）を設定し、[メールチェック]のチェックボックスをオンにすればいい。

ディレイで設定した秒数（初期値180秒）が経過すると、メールボックスがチェックされ、メモの内容が「6 mails recv」などメール数の表示に変化する（**画面5**）。この変化は一時的なもので、しばらくするとメモの表示は元の内容に戻る。




画面3　strftime文字列とディレイ機能の組み合わせで時計に変身。

メールクライアントと連動させるには、プロパティの[新メール着信時起動コマンド]に、メールクライアントを起動するコマンドラインを設定する。また、メールクライアントの側では、起動時にメールの受信を行う設定を行っておこう。

### ●HtmlHeadLine.shをインストール

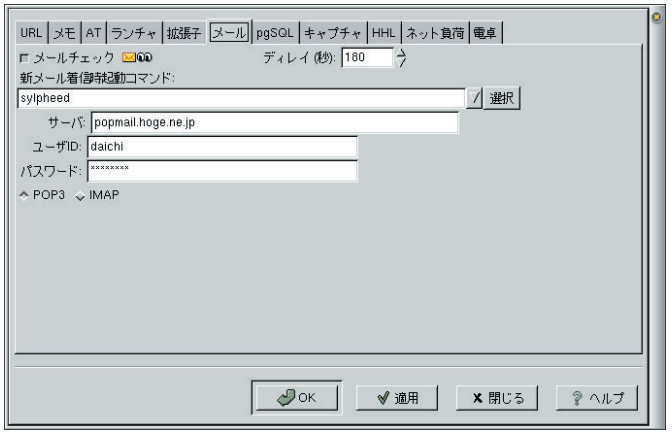
MemoPanelの最大の特長は、同じ作者が作った「HtmlHeadLine.sh」と組み合わせて、ヘッドラインニュースを次々に表示するニュースティッカーとして利用できることだ。

HtmlHeadLine.shは、freshmeatなどのニュースサイトから取得した情報から自動的にヘッドラインニュースを生成するシェルスクリプトで、対応サイトの数は現在280に達する。MemoPanelと同様にtarボールのみ配布されている。

まずは、展開先ディレクトリにあるSITES.tableをテキストエディタで編集して、ニュースサイトの選択を行う。初期設定では、280のサイトのうち、「END」の前に書かれた12個だけが実際の取得対象だ。

12のサイトのうち、対象から除外したいサイトの行頭には「#」を挿入してコメントにする。また、ほかに興味あるサイトがあれば、そのサイトの記述を「END」より前にコピー&ペーストしておこう。

ニュースサイト選択後、展開先ディレクトリで「./Make.source.sh」とすると、HtmlHeadLine.shが作成される。




画面4　一定間隔でメールの到着を調べるメールチェッカーとしての設定。

インストーラは付属しないので、「su -」としてrootになってから、「cp HtmlHeadLine.sh SITES.table /usr/local/bin」として/usr/local/binにコピーしよう。

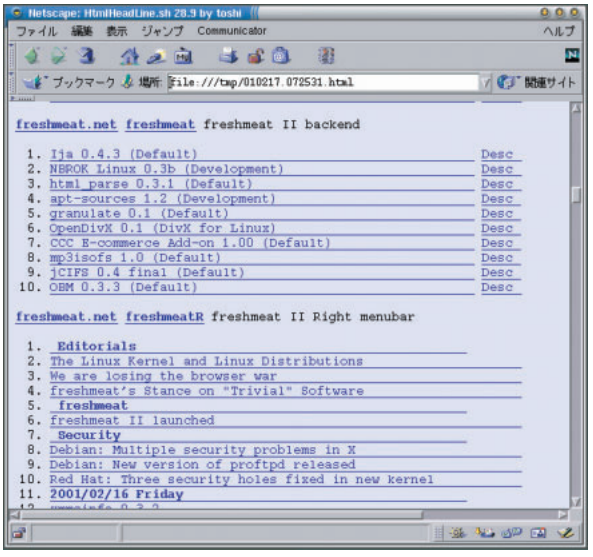
### ●MemoPanelと組み合わせて使う

HtmlHeadLine.sh単独で使う場合は、ヘッドラインニュースが列挙されたHTMLファイルを生成して、Netscapeなどのブラウザで閲覧する（**画面6**）。

一方、MemoPanelと組み合わせる場合は、HTMLファイルではなく、プレーンなテキストファイルを生成する必要がある。それには、環境変数FILE_ONLYに「YES」を設定してHtmlHeadLine.shを起動すればいい。

実際の起動は、MemoPanelのランチャ機能を利用して行う。プロパティの[ランチャー]ページで、[コマンド]のリストから「export FILE_ONLY=YES;HtmlHeadLine.sh」を選択し、[開始]ボタンを押すだけだ。取得した情報は、/tmp/for.MemoPanel.datに保存される。

続いて、この情報をMemoPanelで表示するための設定を行う。プロパティの[メモ]ページで、[HtmlHeadLine.sh作成ファイルからの表示]をチェックし、[ディレイ（秒）]に短めの値（5など）を設定しよう。これで、取得した情報が5秒ごとにヘッドラインニュースとして2行ずつパネルに表示されるようになる（**画面7**）。



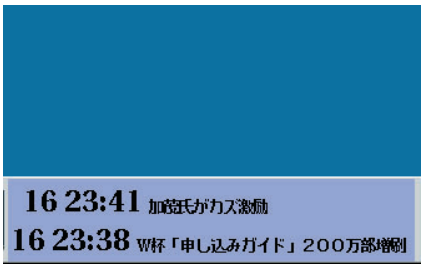

画面6　HtmlHeadLine.shで取得した情報をWebブラウザで表示。


画面5　到着したメールの数が一時的にメモ上に表示される。

なお、HtmlHeadLine.shを一定時間（たとえば1時間や30分）ごとに繰り返し実行できれば、MemoPanelで表示するヘッドラインニュースの内容を常に最新の状態に保てる。

それには、HtmlHeadLine.shに同梱のシェルスクリプトSTART_HHL_30.shを利用するのが簡単だ。まず、スクリプトの49行目の設定を「export FILE_ONLY=YES」に変更し、「cp START_HHL_30.sh /usr/local/bin」としてコピーする。あとは、「SHART_HHL_30.sh&」としてバックグラウンドで実行すると、30分ごとにHtmlHeadLine.shが繰り返し実行されるようになる。

取得間隔を変更する場合は、/usr/local/binにコピーしたSTART_HHL_30.shの61行目「sleep 30m」の「30m」を適宜変更すればいい。たとえばこれを「2h」とすると、2時間ごとに情報を取得するようになる。詳細はsleepコマンドのマニュアル（man sleepで表示）を参照されたい。




画面7　HtmlHeadLine.shで取得した情報をヘッドラインとして表示。

LinuxマシンをPalmOS機と連携

# J-Pilot

バージョン：0.99　　ライセンス：GPL

http://jpilot.org/
http://www.gnu-designs.com/pilot-link/（pilot-link）

J-Pilotは、PalmOS機と連携動作する個人情報管理（PDA）ソフトだ。予定表やアドレス帳などのデータの入力やホットシンク（同期）、Palm用ソフトのインストールなどをX上で実現する。また、メニューなどが日本語で表示されるだけでなく、日本語EUCとシフトJISの相互変換機能により、PalmOS機とLinuxマシン間で問題なく日本語データをやりとりできる。動作にはpilot-linkが必要だ。

### ●pilot-linkを先にインストール

J-Pilotをインストールする前に、PalmOS機とのホットシンクなどに必要なpilot-linkライブラリをインストールする必要がある。pilot-linkはtarボールのみ配布されている。展開先ディレクトリで「./configure」→「make」でビルドした後、「su -」でrootになってから「make install」としてインストールする一般的な手順だ。

なお、ライブラリは/usr/local/libにインストールされるので、J-Pilotがこれを参照できるようにする必要がある。rootになった状態で、/etc/ld.so.confの末尾に「/usr/local/lib」を（もしまだなければ）追加し、ldconfigコマンドを実行しておこう。

### ●J-Pilotのビルドとインストール

J-Pilotは、tarボールとRPMパッケージが配布されている。ただし、最新版（0.99）にはマルチバイト文字の処理に不具合があり、日本語データを登録すると異常終了してしまう。

そこで、日本語を正しく扱えるようにするパッチ（jpilot-0.99-mb.patch）を作成し、パッチ済みのソース・バイナリパッケージも用意した。

なお、pilot-linkをtarボールからインストールしている場合は、バイナリパッケージのインストール時に--nodepsオプションを付けて、「rpm -Uvh --nodeps jpilot-0.99-1mb.i386.rpm」とすること。

tarボールの場合、展開先で「patch -p1 < jpilot-0.99-mb.patch」としてパッチを当てた後は、pilot-linkと同じ手順だ。ただし、一部機能がプラグインとして独立しているので、サブディレクトリのExpenseとSyncTimeでも同様の作業を行う必要がある。

### ●利用するポートの設定を行う

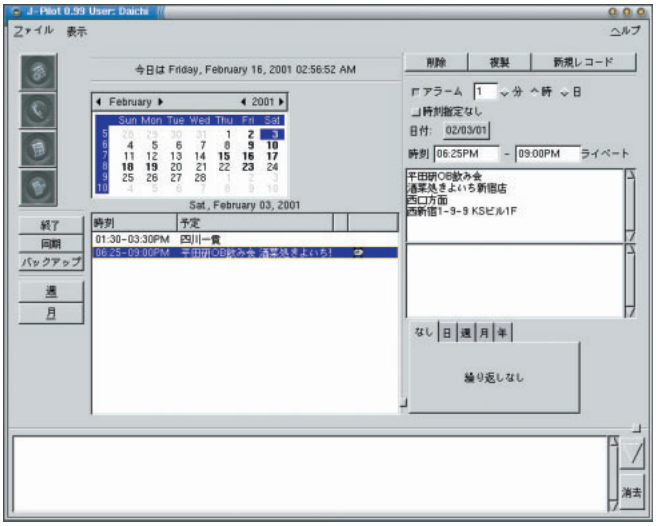
PalmIII／Vなどでは、シリアルポート接続のクレイドルでホットシンクを行う。デバイス名は、COM1が「/dev/ttyS0」、COM2が「/dev/ttyS1」だ。どちらにクレイドルを接続しているか確認しておこう。

一方、USB接続のクレイドルでホットシンクするVisorの場合は、USB対応のカーネルが必要だ。さらに、rootになってから、「modprobe usb-uhci」→「modprobe usb-serial」として、USBホストアダプタとシリアルコンバータのモジュールを組み込む。デバイス名は、「/dev/ttyUSB1」あるいは「/dev/usb/ttyUSB1」となる。

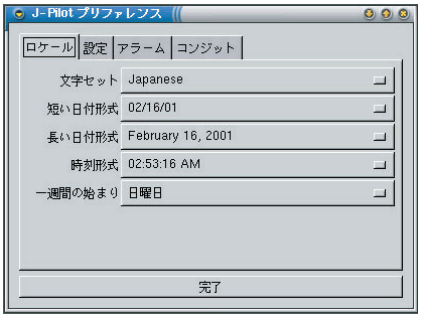
なお、一般ユーザーがホットシンクするには、これらのデバイスへのアクセスをrootに許可してもらう必要がある。たとえば、COM2を利用するなら、rootになった状態で「chmod 666 /dev/ttyS1」とすればいい。

このほか、J-Pilotを利用するユーザーを、/etc/groupに記述された/dev/ttyS1の所有グループに追加するという方法もある。

最後に、ホットシンクで利用するデバイスに「/dev/pilot」というシンボリックリンクを張る。たとえば、




画面1　J-Pilotでは、PalmOS機の予定表やアドレス帳をX上で扱える。




画面2　最初に[文字セット]を「Japanese」に設定するのを忘れずに。

COM2を利用するなら、rootになった状態で「ln -s /dev/ttyS1 /dev/pilot」とすればいい。

### ●J-Pilotの起動と初期設定

ktermなどのコマンドラインで「J-Pilot&」とすると、ウィンドウが開いて今日の予定表が表示される（**画面1**）。日本語環境では、メニューやボタンなどが日本語で表示されるはずだ。

最初に起動したときには、PalmOS機と正しくデータをやりとりするためにいくつか初期設定を行う必要がある。[ファイル]－[プリファレンス]で、ジャンル別にページ分けされた設定ダイアログを開こう。

まず、[ロケール]ページ（**画面2**）で、[文字セット]の設定を「Japanese」に変更する。これを忘れると、ホットシンク時に文字コードの変換が行われないので注意しよう。

[設定]ページ（**画面3**）では、[シリアルポート]に「/dev/pilot」と入力し、[回線速度の選択]を「57600」にする。このほか、[GTKカラーファイルの設定]を「jpilotrc.default」に変更しないと、次回からウィンドウが薄紫で表示されるので注意しよう。

### ●データをホットシンクする

初期設定が完了したら、いよいよPalmOS機とのホットシンクを行ってみよう。クレイドルにPalmOS機を収納してからクレイドルのホットシンクボタンを押し、J-Pilotのウィンドウの[同期]ボタンを押せばいい。

シリアル接続の場合はボタンを押す順番が逆でも構わないが、USB接続の場合は、クレイドルのボタンを押してから機器が認識されるまで数秒かかるので、しばらく待った後で[同期]ボタンを押すのがコツだ。

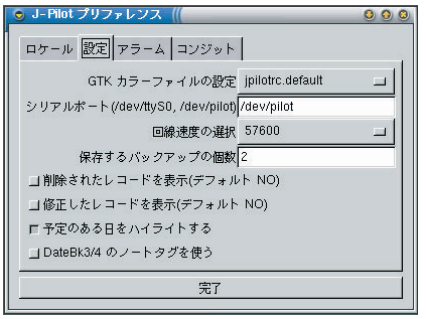
ホットシンク中は、ウィンドウ下部の領域に処理内容が表示される（**画面4**）。「完了」と出ればホットシンクは終了だ。

なお、LinuxマシンとPalmOS機では使われる文字コード系が異なるが、PalmOS機からデータを読み込む際には日本語EUC、逆にPalmOS機にデータを書き込む際にはシフトJISに自動変換しているため、日本語のデータが文字化けする心配はない。

### ●X上でPalmOS機のデータを編集

左上の4つのボタンで、予定表・アドレス帳・Todoリスト・メモを切り替えられる。Windows用のPalm Desktopと使い方が似ているので、戸惑うことはないだろう。

新たなレコードを入力するには、左上の[新規レコード]ボタンを押してから、各フィールドにデータを入力する。文字列用のフィールドには、日本語を入力可能だ。データ入力後は、左上のボタンが[レコードの追加]に変化するので、これを押して登録する。

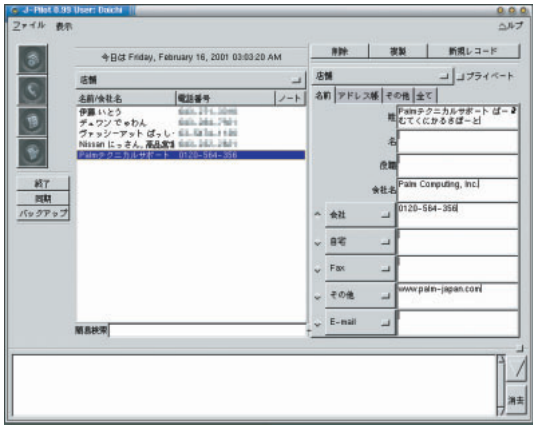



画面3　[設定]ページでは、接続先デバイスと回線速度を設定する。

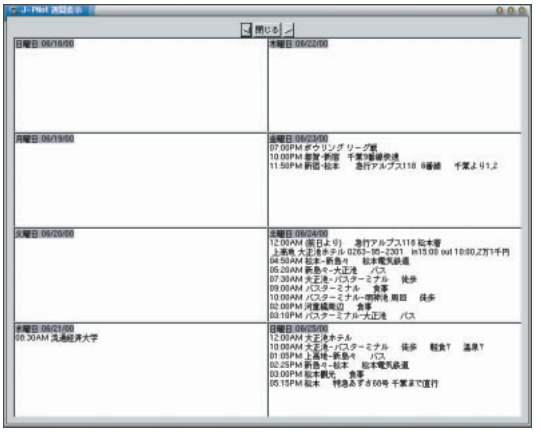
もちろん、既存のレコードの変更や複製、削除も可能だ。また、プライベートレコードは、[表示]メニューの項目で表示の有無を切り替えられる（パスワード入力が必要）。

なお、英語版PalmOSには「読み」というフィールドがないため、アドレス帳の姓・名・会社名などは、「漢字」+スペース+「読みがな」という形式になってしまう（**画面4**）。

予定表では、左の[週]／[月]ボタンを押すことで、1週間または1ヵ月の予定が別ウィンドウに表示される。特に、DateBk3風に1週間の予定が縦横に並ぶ週間表示ウィンドウは見やすいのでお勧めだ（**画面5**）。



画面4　日本語の姓名などが、読みがなと一緒になって表示されてしまう。



画面5　[週]ボタンを押すと、週間の予定を表示するウィンドウが開く。

CPUやディスクの負荷をグラフ表示

# GKrellM

バージョン：1.0.6　　ライセンス：GPL

http://web.wt.net/~billw/gkrellm/gkrellm.html
http://www.muhri.net/（テーマ）

GKrellMは、CPUの使用率やイーサネットのパケット流量、メモリやスワップの使用量などを監視して、リアルタイムにグラフやメーター表示するモニタリングソフトだ。表示内容は設定ダイアログで柔軟にカスタマイズできる。このほか、メールチェックやCD-ROMのマウントなどが可能だ。加えて、プラグインによる機能の拡張、テーマ（スキン）による外見の変更にも対応している。

## ●ビルドとインストール

GKrellMは、ファイル一式をtar＋gzipしたtarボールのほか、RPMパッケージでも配布されている。

ただし、バイナリパッケージに含まれるGKrellMは国際化未対応版なので、付属の日本語カタログが使われない。以下の手順でtarボールあるいはRPMソースパッケージから国際化対応版をビルドするとよいだろう。

tarボールを展開後、まず「./enable_nls」として国際化対応パッチを当てる。あとは、「make」としてビルドした後、「su -」でrootになってから「make install」としてインストールすればいい。

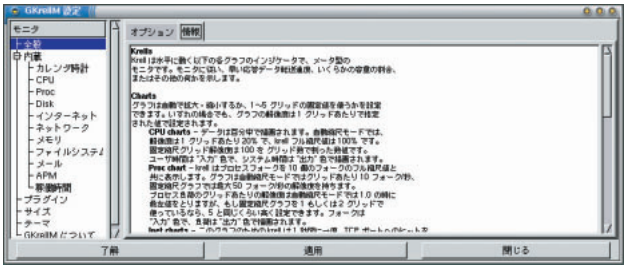
RPMを利用する場合は、rootになった状態で「rpm -i gkrellm-1.0.6-1.src.rpm」としてソースパッケージをインストールする。続いて、/usr/src/redhat/SPECSにあるgkrellm-1.0.6.specをエディタで開いて、35行目の「%build」の次に、「./enable_nls」という行を挿入する。続いて、「rpm -ba gkrellm-1.0.6.spec」とすると、/usr/src/redhat/RPMS/i386に国際化対応版のバイナリパッケージが作成される。これを、「rpm -Uvh gkrellm-1.0.6-1.i386rpm」としてインストールすればいい。


画面1　CPUやプロセス、ディスクの負荷がグラフで表示される。

## ●1秒ごとにグラフが更新される

以下では、日本語環境で国際化対応版のGKrellMを利用するので、未対応版を利用する場合は、メニューの名前などを適宜読み替えてほしい。

ktermなどのコマンドラインから「gkrellm&」として起動すると、縦型の小さなウィンドウが開いて、CPU・プロセス・ディスクのグラフや、メモリ・スワップのメータなどが表示される（**画面1**）。



画面2　国際化対応版GKrellMでは、ダイアログの説明も日本語表示だ。

情報は1秒ごとに更新され、グラフの縦方向のスケールは測定値に応じて自動的に調整される。また、各グラフともシアンとオレンジの2色で異なるデータを表示する。たとえば、CPUのグラフではユーザータイムとシステムタイム、ディスクのグラフではリードとライトといった具合だ。

なお、マルチプロセッサシステムでは、使用しているCPUの数だけグラフが表示される（**画面1**では2個）。すべてのCPUを合成したグラフを1つだけ表示することも可能だ。

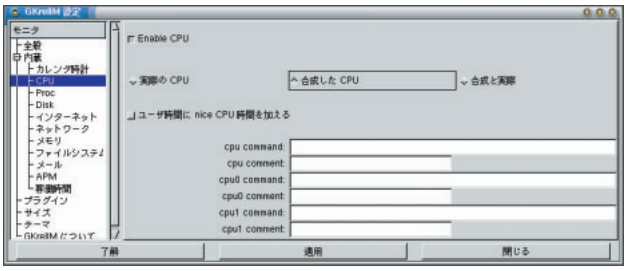
## ●表示内容は柔軟に変更できる

ウィンドウ上部のホスト名表示部を右クリックして、ポップアップメニューから[設定]を選択すると、設定ダイアログが開く。

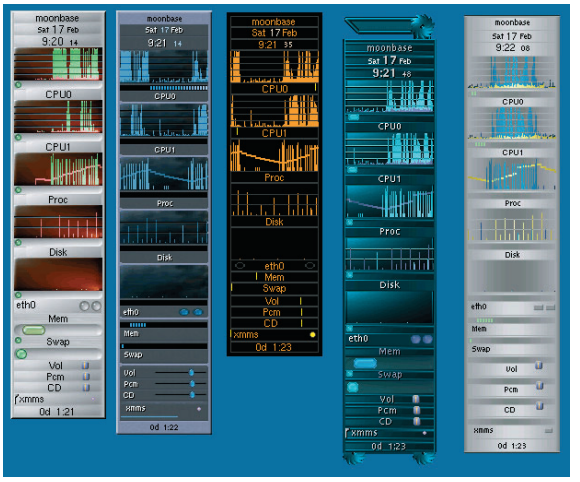
設定項目は左側のツリーでジャンル別に分類されており、クリックで選択したジャンルの設定ページが右側に表示される。国際化対応のGKrellMでは、項目名や情報が日本語で表示されるのでありがたい（**画面2**）。

GKrellM内蔵のモニタ機能については、[内蔵]以下のツリーにまとめられており、各項目の表示の有無や計測単位量の変更などが可能だ。

たとえば、複数のCPUのグラフ表示を1つにまとめるには、CPUの設定ペ



画面3　CPUの設定ダイアログで、合成したCPUのグラフ表示を選択する。



画面4　テーマ（スキン）を使うと、簡単に外見を切り替えられる。

ージ（**画面3**）で、[合成したCPU]をチェックすればいい。一方、[Enable CPU]のチェックを外せば、CPUのグラフ自体が表示されなくなる。

[ネットワーク]の設定ページのように、設定項目が複数ページに分かれているジャンルもある。PPPではなくLANカードでネットワークに接続している場合は、[ppp0]ページの[PPPを有効にする]をオフにし、[eth0]ページの[eth0を有効にする]をオンにすればいい。これで、イーサネットのパケット流量のグラフが表示される。

### ●テーマを導入して見栄えを変える

GKrellMでは、テーマ（スキン）と呼ばれる一連の画像ファイルを導入することで、簡単に外見を変更できる（**画面4**）。テーマサイトの「muhri.net」には130種類以上のテーマが用意されているので、気に入ったものをダウンロードしよう（本誌CD-ROMにも、数個のテーマを収録している）。

これらのテーマをGKrellMに組み込むには、各ユーザーのホームディレクトリにある.gkrellm/themesで、テーマのtarボールを展開すればいい。

以後、設定ダイアログの[テーマ]の設定ページで、テーマを選択できるようになる。また、GKrellMをクリックしてからP／Nキーを押すと、テーマをすばやく切り替え可能だ。

なお、テーマを作成するためのリファレンス（英文）が、GKrellMのサイトやtarボールに付属するので、これを参考にしてオリジナルのテーマを作成することもできる。

### ●プラグインで機能を拡張する

テーマ（スキン）が外見を変更するのに対し、GKrellMの機能を拡張するのはプラグインの役割だ。

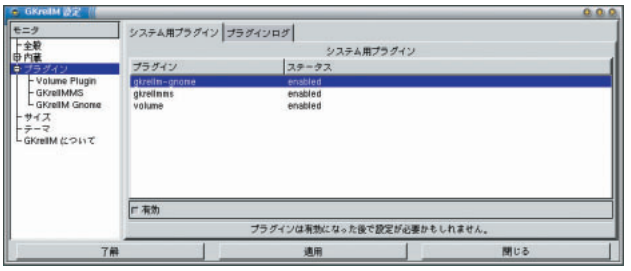
GKrellMのサイトのPluginsページには、現時点で13種類のプラグインが紹介されており、SETI@Homeや天気予報のモニタリングなどさまざまな機能をGKrellMに追加できる。

本誌CD-ROMには、GNOMEとの親和性を高める「gkrellm-gnome」、XMMS（MP3プレーヤ）を制御する「gkrelmms」、ボリュームを制御する「volume」を収録した。

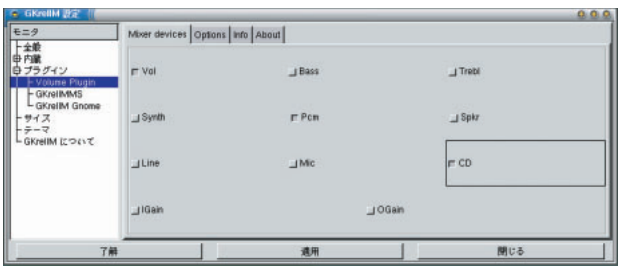
プラグインは、rootになった状態で、RPMパッケージやtarボールをビルド＆インストールしよう（gkrellmmsのビルドにはxmms-develパッケージが必要）。いずれも、/usr/lib/gkrellm/pluginsにインストールされる。

インストールしたプラグインを組み込むには、実行中のGKrellMをいったん終了する。再起動後、設定ダイアログの[プラグイン]の設定ページで、個々のプラグインを選択して[有効]をチェックしよう（**画面5**）。

さらに、プラグインごとの設定ページで固有の設定を行う必要がある。たとえば、volumeの場合は、どのデバイスのボリューム設定をGKrellMで制御する選択する（**画面6**）。



画面5　インストールしたプラグインは忘れずに[有効]をチェックする。



画面6　プラグイン固有の設定を行うページがそれぞれ用意されている。

各種のアーカイブをGUIで操作

# KArchiver

バージョン：0.87／2.0.3　ライセンス：GPL

http://perso.wanadoo.fr/coquelle/karchiveur_en.shtml

KArchiverは、tar＋gzipやtar＋bzip2で作られたtarボールをはじめ、ZIP／LZH／ARJファイルなどのアーカイブをGUIで操作できるソフトだ。ファイルの展開や追加、閲覧はもちろん、アーカイブをすばやく選択するファイルブラウザ、アーカイブ形式を変換するコンバータ、さらにはインストールなどの操作を代行してくれるウィザードまで備えた総合的な管理ソフトだ。実行にはKDE1または2が必要となる。

## ●ビルドとインストール

KArchiverは、KDE1用（0.87）とKDE2用（2.0.3）が、それぞれtarボールとRPMパッケージで配布されている。なお、KDE1用は正確には「KArchiveur」と末尾のrの前にuが付く。

現時点でKDE2を標準採用している国産ディストリビューションは少数なので、以下ではKDE1用の0.87について説明する。機能的にはほぼ同じだ。

RPMバイナリパッケージをインストールするには、「su -」としてrootになった状態で、「rpm -Uvh karchiveur-0.87-1.i586.rpm」とする。ただし、このままだとツールバーのボタンなどの画像が一部表示されないので、「cp /usr/share/icons/karchiveur＊.xpm /usr/kde/share/icons」および「cp /usr/share/icons/mini/karchiveur＊.xpm /usr/kde/share/icons/mini」としてコピーしておく。

一方、tarボールからのインストールは、「./configure」→「make」としてビルドし、「su -」でrootになってから「make install」とする一般的な手順だ。

なお、Kondara 1.1／2000では、KDE1が/usr/kde1x以下にインストールされているため、「./configure --x-includes=/usr/kde1x/include/kde」とする必要がある。

## ●起動とアーカイブの読み込み

ktermなどから「karchiveur&」とするか、KDEメニューの[Utilities]－[karchiveur]を選択すると、大小2つのウィンドウが開く。

大きいほうはメインウィンドウで、アーカイブに含まれるファイルの一覧が表示される（**画面1**）。アーカイブを開くには、ツールバーの[Open Archive]ボタンでダイアログから選択するか、ファイルマネージャなどからドラッグ＆ドロップすればいい。

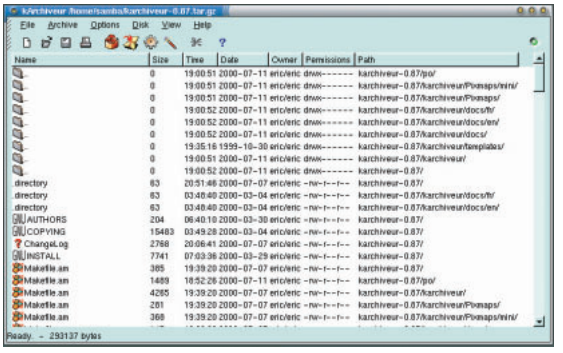
一方、小さいウィンドウは「ファイルブラウザー」と呼ばれ、現在メインウィンドウで操作しているアーカイブと同じディレクトリにあるアーカイブの一覧が表示される（**画面2**）。ファイル名のダブルクリックでアーカイブを読み込み可能だ。

このほか、[File]メニューには、指定したディレクトリ以下から再帰的にアーカイブを検索する[Search]（**画面3**）なども用意されている。

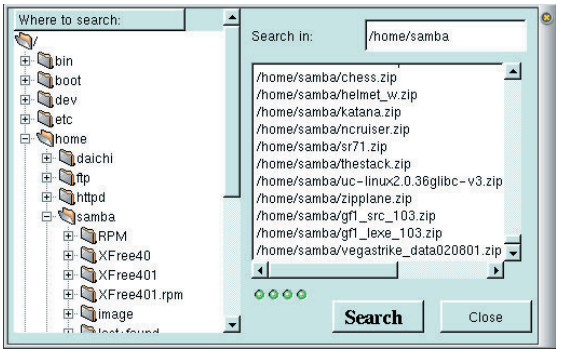
対応しているアーカイブ形式は、tar、tar＋gzip（拡張子はtar.gz、tgz）、tar＋bzip2（tar.bz2）、LHA（lzh）、ZIP、ARJ、RARなどだ。

なお、tar関連以外のアーカイブは外部コマンドを利用するので、それぞれ

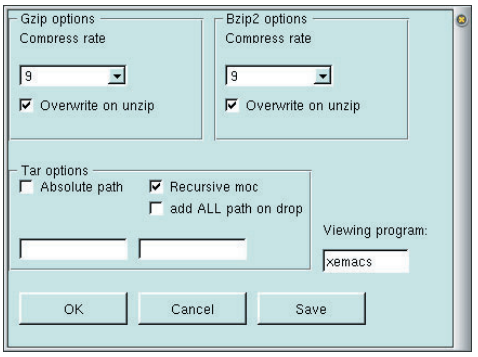
画面1　メインウィンドウにはアーカイブ内のファイルが一覧表示される。



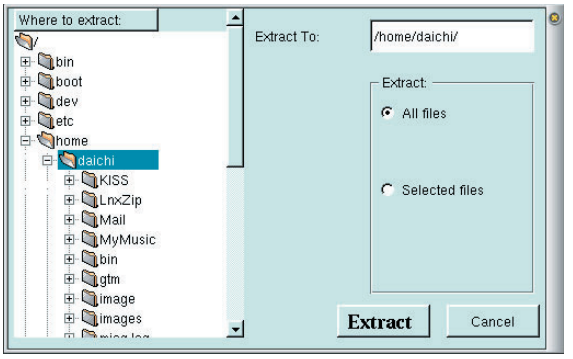

画面2　アーカイブをすばやく開けるファイルブラウザ。




画面3　指定したディレクトリ以下からアーカイブを検索して一覧表示。




画面4　tar/gzipのオプションやテキスト閲覧用エディタを変更できる。



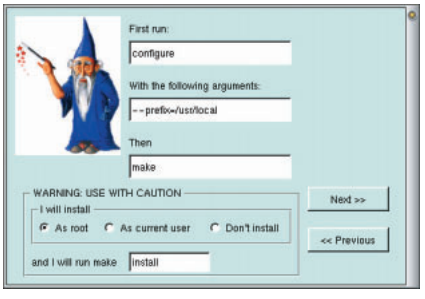

画面5　展開先ディレクトリを左側のツリーから選択しよう。

対応するコマンド（たとえばZIPならzip／unzip）が環境変数PATHに設定されたディレクトリに置かれている必要がある。

### ●基本的なファイル操作

アーカイブ内のファイルは、単にクリックするだけで複数選択できる（再度クリックで選択解除）。ドラッグによる範囲選択も可能だ。

その後、ツールバーの[View file]ボタンを押すと、ファイルの内容をテキストエディタのkeditで閲覧できる。別のエディタを使いたいなら、[Options]－[options]で開く設定ダイアログ（**画面4**）で変更しよう。




画面6　インストールやアーカイブ分割などを代行してくれるウィザード。


画面7　インストール用のウィザードではオプションの追加なども可能だ。

アーカイブ内のファイルを展開するには、ツールバーの[Extract]ボタンを押し、展開先をダイアログのツリーから選択する（**画面5**）。展開する対象は、「全ファイル」と「選択したファイル」を切り替え可能だ。

アーカイブにファイルを追加する際も同様のダイアログが開くので、対象となるファイルをツリーから選択すればいい。ファイル数が多い場合には、ワイルドカード（＊、？）を使った一括登録に切り替えよう。

### ●ウィザードを使ってみよう

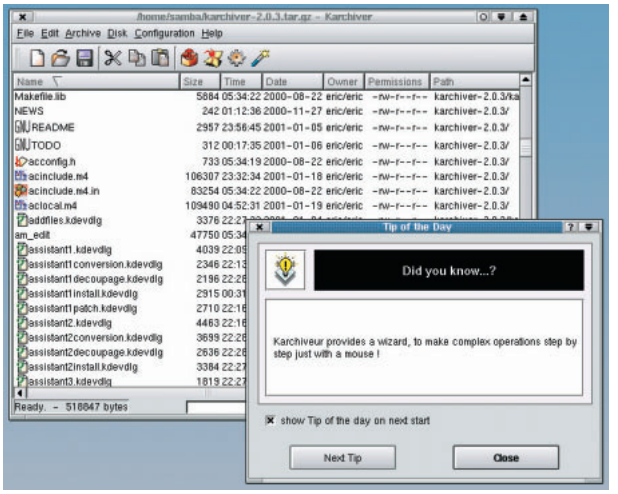
ツールバーの[Launch the wizard]ボタンを押すと、インストールやパッチ当て、アーカイブ形式の変換、アーカイブの分割・結合などを対話的に行うウィザードが開く（**画面6**）。

たとえば、[Install the archive]では、「configureスクリプトを実行後、makeでビルドし、rootになってインストールする」という一連の作業をユーザーに代わって行ってくれる。configureのオプションなどを追加指定することも可能だ（**画面7**）。

また、[Convert archive format]では、tar.gzからtar.bz2などのアーカイブ形式の変換を行えるし、[Split archive]では、アーカイブをフロッピーサイズのファイルに分割したり、分割したファイルを元のアーカイブに統合することができる。

### ●KDE2用の2.0.3を少しだけ紹介

KDE2用のKArchiver（こちらはuが付かない）では、さらに画面が洗練され、「Tip of the Day」ウィンドウも追加されている（**画面6**）。今後はどんどん機能追加されていくと思われるので、KDE2を使える環境にある人は、ぜひともこちらを利用してほしい。




画面8　KDE2用の2.0.3では「Tips of the Day」が用意されている。

対戦型ボードゲームGIPFをX上でプレイ

# GF1(GIPF for One)

バージョン：1.03　　ライセンス：GPL

http://gf1.sourceforge.net/
http://fltk.easysw.com/（FLTK）
http://www.boutell.com/gd/（GD）

GF1（GIPF for One）は、ドイツ生まれの対戦型ボードゲーム「GIPF」（ギプフ）をコンピュータ相手にプレイするソフトだ。六角形の盤面に黒と白の駒を交互に押し込み、相手の駒を奪う戦略性の高いゲームだ。GF1では、熟練度に応じて設けられたGIPFの3種類のルールをすべてプレイできる。人間同士やコンピュータ同士で対戦することも可能だ。実行にはFLTKとGDの各ライブラリが必要となる。

### ●ライブラリのインストール

GF1をインストールする前に、実行に必要な2つのライブラリを先にインストールしておこう。

「FLTK」（Fast Light Tool Kit）はC++用のGUIツールキットライブラリで、サイトではtarボールのみ配布されている。付属のSPECファイルを利用すれば、tarボールからRPMパッケージを作成することも可能だ。

tarボールからのビルドとインストールは、「./configure --enable-shared」→「make」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。

RPMパッケージを作成する場合は、rootになった状態で「rpm -ta fltk-1.0.10-source.tar.gz」とする。/usr/src/redhat/RPMS/i386にバイナリパッケージが作成されるので、「rpm -Uvh fltk-1.0.10-1.i386.rpm fltk-devel-1.0.10-1.i386.rpm」としてインストールしよう。

グラフィック描画ライブラリ「GD」のほうは、RedHat系ディストリビューションでは最初からインストールされている。tarボールを利用する場合は、Makefileを自分の環境に合わせて書き換えてから、「make」でビルドし、rootになってから「make install」でインストールする。

### ●GF1のビルドとインストール

GF1は、ビルド済みの実行ファイルを含むバイナリ版とソース版のZIPアーカイブ（tarボールではない）がそれぞれ配布されている。glibcを利用したほとんどのディストリビューションでは、バイナリ版の実行ファイルがそのまま動作するはずだ。

バイナリ版のインストールは以下のように行うとよいだろう。まず、「su -」でrootになってから、「unzip gf1_lexe.zip -d /usr/local/games/GF1」として、/usr/lcoal/games/GF1以下にファイルを展開する。

一方ソース版の展開も、ファイル名が異なる点以外はバイナリ版と同じだ。ただし、このままビルドするとキャストに関するエラーが表示される。これを解消するパッチ（gf1-g++.patch）を用意したので、「patch -p0 < gf1-g++.patch」としてソースを修正してから、「make」でビルドしよう。

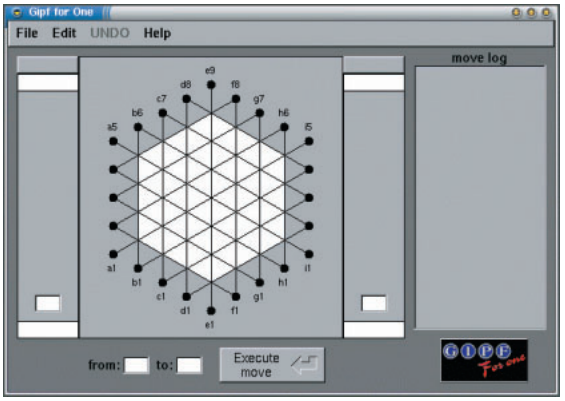
いずれの場合も、どのディレクトリからでも実行できるように、「ln -s /usr/local/games/GF1/gf1 /usr/local/bin/gf1」として、シンボリックリンクを作成しておくといい。

### ●起動とプレイ中の操作

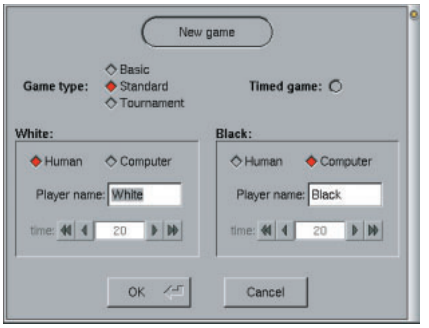
ktermなどから「gf1&」として起動すると、六角形の盤面を含むメインウィンドウが開く（**画面1**）。

[File]－[New game]を選択して、ルールや対戦相手などを設定しよう（**画面2**）。初期設定では、先攻（白）を人間、後攻（黒）をコンピュータプレイヤーが受け持つ。人間同士で対戦したり、コンピュータ同士の対戦を観戦することも可能だ。ルールは3種類あるが、最初は最も単純な「Basic」を選択すればいい。[OK]ボタンを押せばいよいよプレイ開始だ。

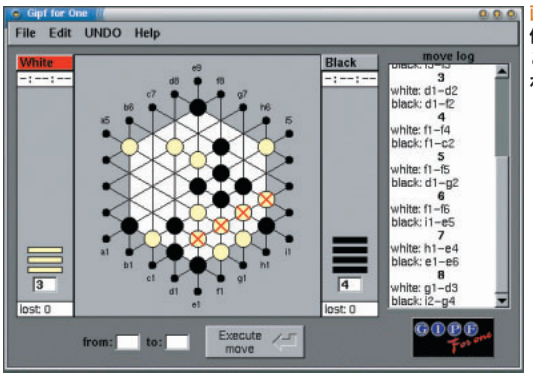
ゲーム中の操作はすべてマウスで行う。まず、白い盤面の外周に配置された黒点（ドット）のいずれかをクリックして、左右のリザーブにある駒を置



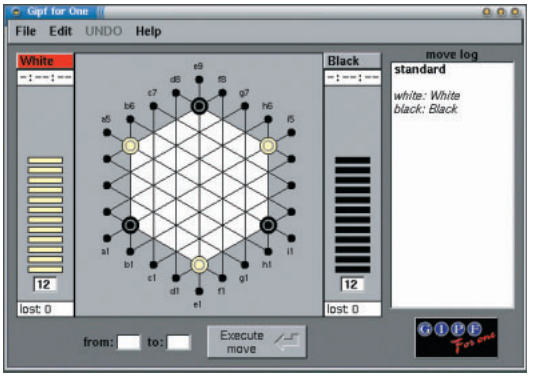

画面1　GIPFは、六角形の盤面でプレイする戦略性の高いゲームだ。




画面2　ルールや対戦相手を選択する。コンピュータ同士の対戦も可能。




画面3　同色の駒が4個連続して一列に並ぶと、盤面から取り除かれる。




画面4　二重丸で表示されたGIPF駒が登場するStandardルール。

く。陣地の概念はないので、どのドットにも駒を配置可能だ。

次に、ドットに置いた駒の移動先を指定する。具体的には、そのドットから伸びた線が、盤面と交わっている部分をクリックすればいい。移動先は赤い×印で表示される。

最後に、ウィンドウ下部の[Execute Move]ボタンを押すと操作が確定し、ドットに置いた駒が×印の位置まで移動する。このとき、移動先に自分や相手の駒が置かれていると、自動的に移動方向に押し出される。

### ●GIPFの基本ルールを習得

基本的なルールは、「同色の駒が4個連続して一列に並んだら、それらを盤面から取り除いてリザーブに戻す」というものだ（**画面3**）。

このとき、4個の駒と隙間を空けずに並んでいる同列の駒も盤面から取り除かれる。4個と同じ色の駒はリザーブに戻されるが、違う色の駒は捕獲され、それ以降は使えなくなる。

最終的に、どちらかのプレイヤーのリザーブに駒がひとつもなった時点で、ゲームの続行が不可能になり、そのプレイヤーの負けが決定するわけだ。

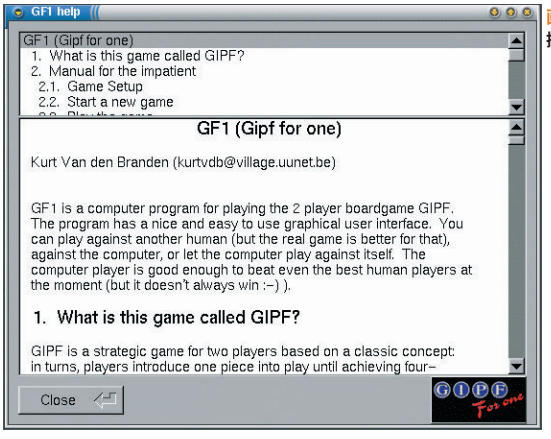
なお、自分の駒を動かした結果、相手の駒が4つ並んだ場合も（相手の）リザーブに駒が戻されることに注意しよう。捕獲する駒がない状態で4個もの駒が盤面から取り除かれると、それ以降の展開が不利になる。自分の駒が捕獲されない状態で、相手の駒が4つ並ぶようにするのも重要なテクニックのひとつだ。

### ●上級ルールではGIPF駒がカギ

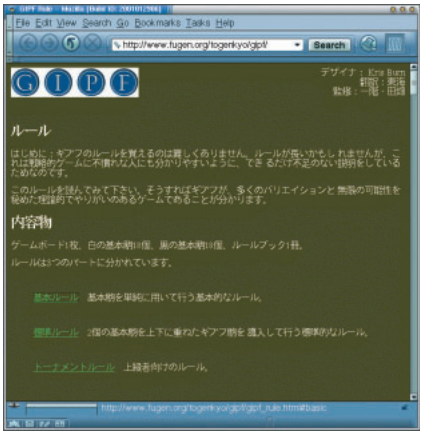
基本的なルールを習得したら、面白さが倍増する上級ルールでプレイしよう。Standardルールでは、通常の駒2個分の価値を持つ「GIPF駒」（◎で表示）が3個配置されている（**画面4**）。Tournamentルールでは、序盤にプレイヤーが好きな数だけGIPF駒を登場させることが可能だ。

GIPF駒のメリットは、4つ並んだ場合に取り除くかどうかを、駒を動かしたプレイヤーが選択できることだ。その反面、リザーブの駒の数が少なくなる、相手に捕獲されると通常の倍の影響があるといったリスクもあるので使い方が難しい。

[Help]－[Help]で開くヘルプウィンドウ（**画面5**）には、GF1の操作方法などが説明されている。GIPFの詳細なルールは、公式ページ（http://www.gipf.com/）か、ゲームサークル桃源郷の「GIPF Index」（http://www.fugen.org/togenkyo/gipf/）を参照しよう。後者には、GIPFの歴史やルールなどが日本語で紹介されている（**画面6**）。




画面5　ヘルプウィンドウで操作方法などを確認できる。




画面6　ルールの日本語訳などが用意された「GIPF Index」ページ。

一万を超える面があなたを待つ

# Rocks'n'Diamonds

バージョン：2.0.0　　ライセンス：Artistic

http://www.artsoft.org/rocksndiamonds/
http://www.libsdl.org/ (SDL)

Rocks'n'Diamondsは、2Dマップ上でアイテムを集めるアクションパズルゲームだ。レベル（面）のデザインやルール、登場するアイテムや敵などを柔軟に設定できるため、「Boulderdash」シリーズなどの過去の名作ゲームを再現できる。内蔵のレベルクリエータでオリジナルレベルを作成することも可能だ。さらに、1台のPCやネットワーク接続したPCを使った4人までの複数プレイにも対応している。

### ●SDLを導入してからビルドしよう

Rocks'n'Diamonds自体は、ビルド済みの実行ファイルとソースが一緒になったtarボールのみ配布されている。「su -」でrootになって、適当なディレクトリ（/usr/local/gamesなど）でtarボールを展開しよう。

ただし、tarボールに含まれる実行ファイルは、ゲームライブラリ「SDL」がなくても動作するようにビルドされているため、フルスクリーン表示やMODファイルによるBGM演奏など、一部の機能が無効になっている。こうした機能を利用したいなら、SDLを導入後、ソースからRocks'n'Diamodsをビルドする必要がある。

必要なライブラリは、SDL本体とSDL_image、SDL_mixerの3つだ。いずれもtarボールとRPMパッケージの両方が配布されているので、ディストリビューションに合わせて選択しよう。なお、RPMパッケージを利用する場合は、それぞれのdevelパッケージ（SDL-devel-1.1.8-1.i386.rpmなど）もインストールする必要がある。

あとは、Rocks'n'Diamodsのtarボール展開先で、「make sdl」とすれば、SDLに対応版の実行ファイルがビルドされる。そのままの状態でプレイ可能なので、他のディレクトリにインストールする必要はない。

### ●タイトル画面とメニュー操作

Rocks'n'Diamodsの展開先ディレクトリに移動し、「./rocksndiamods&」とすると、タイトルとメニューが表示される（**画面1**）。カーソル移動は↑／↓キー、選択および設定変更はEnterキーで行う。

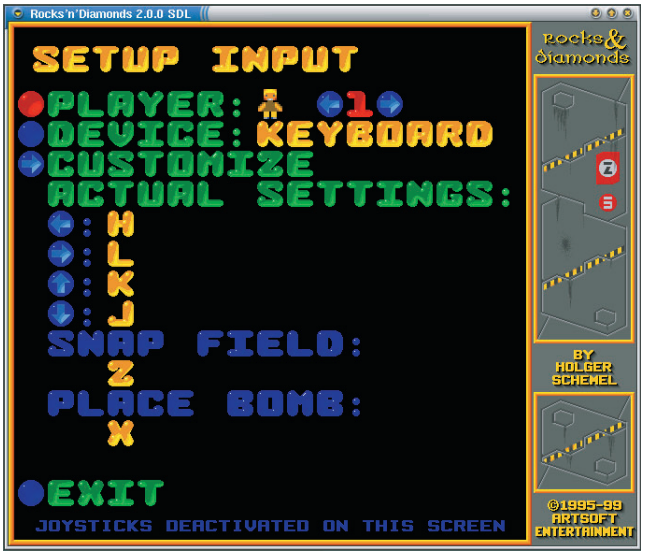
たとえば、[SETUP]を選択して設定画面に切り替え、[TIMELIMIT]を「OFF」に変更すると、制限時間を気にすることなく遊べるようになる。また、[INPUT DEVICES]では、プレイヤーごとの操作デバイス（キーボードかジョイスティック）の選択や、キー配置の変更が可能だ（**画面2**）。

ところで、Rocks'n'Diamondsでは、さまざまなルールでプレイできるレベル（面）の集合「レベルシリーズ」が多数用意されている。まずは最初から選択されている「チュートリアルレベル」をプレイして、基本操作をマスターするとよいだろう。

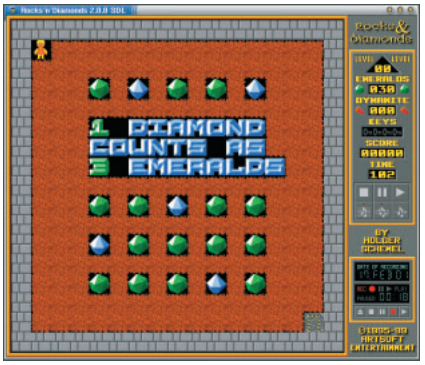
### ●チュートリアルレベルをプレイ

タイトル画面で[START GAME]を選択すると、レベル0からプレイが始まる（**画面3**）。カーソルキーで主人公を移動させて、地中に埋められた宝石


画面1　このタイトル画面で「START GAME」を選択するとプレイ開始だ。




画面2　操作に使うキーをviエディタ風にカスタマイズしてみた。




画面3　宝石をすべて獲得することがチュートリアルレベルでの目的だ。


画面4　下に移動せずに、スナップ操作で宝石だけを取る必要がある。


画面5　遊びきれないほどのレベルシリーズが用意されている。

(エメラルドとダイアモンド)を取ろう。制限時間内に指定数のエメラルド(ダイアモンドはエメラルド3個分)を集め、出口まで達するとクリアだ。いったんタイトル画面に戻り、次のレベルに進めるようになる。

レベル1では、主人公の邪魔をする(時には助けてくれる)「岩」が登場する。主人公が岩の真下を掘ったり、2つ以上積み重なった岩の横を掘ると、岩が落下してくる。主人公との間に隙間がある状態で落ちてきた岩に接触するとゲームオーバーになるので注意しよう(特にスタート直後)。

重要な操作としては、主人公を動かさずに周囲を掘ったり、宝石を取ったりする「スナップ」がある。初期設定のキー配置では、左Shiftキーを押したままカーソルキーを押せばいい。ちなみに、レベル1では、左下にあるエメラルドをスナップで取らないと、絶対にクリアできない(**画面4**)。

以下、「ナッツ」を岩で割って宝石を取り出す(レベル2)、落下で爆発する「爆弾」や右Shiftキーで配置する「ダイナマイト」を活用する(レベル3)、接触するとゲームオーバーの「虫」を岩で殺して宝石に変える(レベル5)といった具合に、さまざまな操作をひとつずつマスターできる。

### ●他のレベルシリーズに挑戦

ある程度チュートリアルレベルを進めたら、別のレベルシリーズにも挑戦してみよう。タイトル画面で[LEVEL]を選択すると、ジャンル別にグループ分けされたレベルシリーズ一覧が表示される(**画面5**)。

「Boulderdash」シリーズなど過去の名作ゲームのクローンレベル(**画面6**)をはじめ、プレイヤーから提供されたレベルシリーズなどが大量に用意されており、レベル数の合計は軽く1万を超える。

ただし、プレイヤー提供のレベルシリーズは、作成者によって難易度やパズル性がかなり異なることをお断りしておく。

もちろん、あなた自身が独自のレベルシリーズを作成することも可能だ。レベルシリーズ一覧の最後にある、あなたのプレイヤー名を選択し、タイトル画面で[LEVEL CREATER]を選択しよう。

レベルクリエータ画面では、マウスを利用して、ペイントソフト感覚で宝石やさまざまなアイテム、敵などを配置できる(**画面7**)。出口が開く宝石の数や制限時間などのパラメータも変更可能だ(**画面8**)。

画面6　過去の名作「Boulderdash」のクローンレベルをプレイ中。

画面7　レベルクリエータで、自分だけのオリジナル面を作成しよう。


画面8　出口を開く宝石の数や制限時間などを設定するINFO画面。

想像したこともないような景色が目の前に

# Gnofract 4D

バージョン：1.4　　ライセンス：フリー

http://gnofract4d.sourceforge.net/

Gnofract 4Dは、GNOMEベースのフラクタル表示ソフトだ。マンデルブロ集合とジュリア集合を統合した関数により、これまでに見たこともないような変わったフラクタル画像を表示できる。また、画像が次第に細かくなるプログレッシブ表示や、美しいグラデーションを実現するカラーマップなど、見栄えについても考慮されている。位置パラメータやフラクタル画像をファイルに保存することも可能だ。

## ●ビルドとインストール

Gnofract 4Dは、tarボールとRPMバイナリパッケージが配布されているので、ディストリビューションに合わせて選択しよう。

tarボールからのビルドは、configureスクリプトを使う一般的な手順だ。ただし、そのままだと/usr以下にインストールされてしまうため、Red Hat系ディストリビューションでは、「./configure --prefix=/usr/local」として、/usr/local以下にインストールしたほうがいい。

## ●「4次元」のフラクタル画像を表示

ktermなどから「gnofract4d&」とするか、GNOMEメニューの[プログラム]－[グラフィック]－[Gnofract 4D」を選択すると、ウィンドウが開いて、おなじみのマンデルブロー集合によるフラクタル画像が表示される。

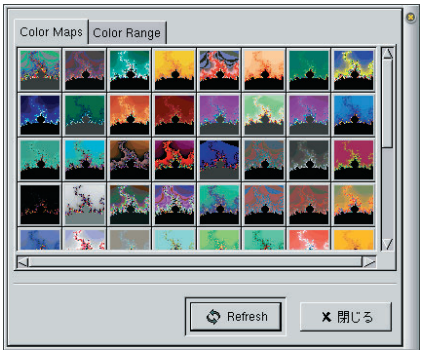
画像を拡大するには、興味ある領域をドラッグで範囲指定すればいい。左右ボタンのクリックで拡大・縮小することも可能だ。画像がしだいに細かくなるプログレッシブ表示により、概形を素早く把握できる。また、中ボタンクリックでマンデルブロー集合とジュリア集合の切り替えが可能だ。

他のフラクタル表示ソフトと異なるのは、マンデルブロー集合やジュリア集合では一部が固定される4つのパラメータを自由に変更できる点だ（数学的な記述はヘルプを参照）。同様に、視点についても6個のアングルボタンで自由に変更できる（**画面1**）。

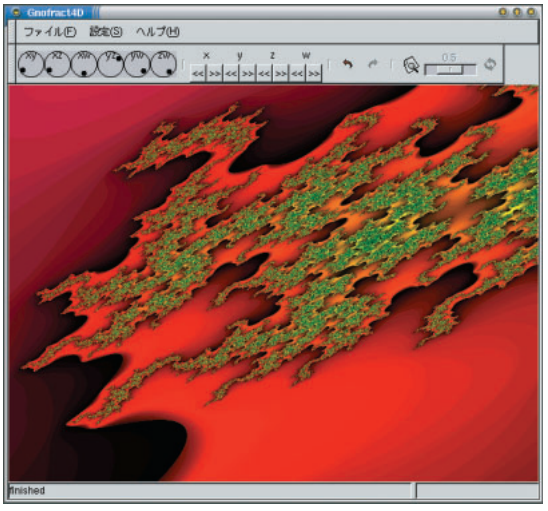
こうした変更が面倒なら、[Explorer]ボタンを押して、エクスプローラーモードに切り替えるといい。ウィンドウが9分割され、少しずつパラメータや視点の異なる画像が表示される（**画面2**）。通常モードと同様に画像を拡大・縮小することも可能だ。

画像の見栄えにこだわる人は、[設定]－[Colors]でダイアログを開いて、表示に使われるカラーマップを変更しよう（**画面3**）。70種類近くのカラーマップが用意されており、クリックするだけで切り替え可能だ。気に入った画像ができたら、[ファイル]－[Save image]でPNG形式の画像ファイルとして保存できる。

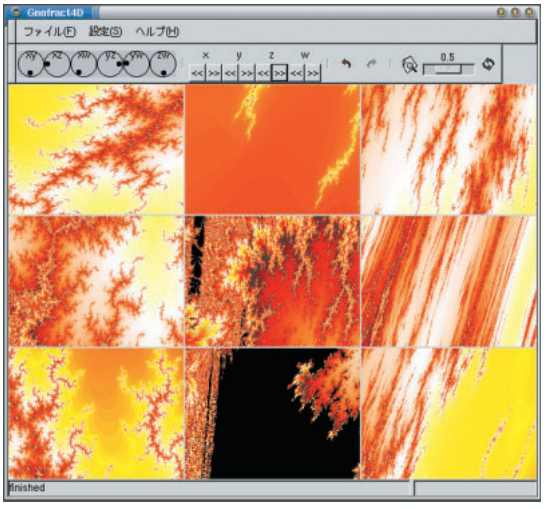
大量の計算が必要なため、マシンの性能によっては表示までにかなり時間がかかるが、刻々と変化していく様子を眺めているのも楽しいものだ。




画面3　カラーマップを変更して、さまざまなグラデーションを楽しもう。




画面1　パラメータや視点の変更により、見なれない画像が表示される。




画面2　画面が9分割されるエクスプローラーモードで周辺を探索。

カタログ作成も可能な国産画像ビューア

# G-thumB

バージョン：0.7.1　　ライセンス：GPL

http://www.prox.jpn.org/~takehiko/ware.html

G-thumBは、エクスプローラ風の外見を持つ国産の画像ビューアだ。サムネイル（縮小画像）で画像ファイルを管理でき、画像のフルスクリーン表示や連続表示（スライドショー）、ルートウィンドウ（背景）に対する画像の設定、Web公開用のHTMLファイルの生成といった機能を備えている。あらゆる機能をキーボードから制御できるのも特長のひとつ。もちろん、メニューなどはすべて日本語表示だ。

## ●ビルドとインストール

G-thumBは、tarボールとRPMパッケージの両方が配布されているので、ディストリビューションに合わせて選択しよう。tarボールからのビルドとインストールは、configureスクリプトを使う一般的な手順だ。

また、tarボールとnosrcパッケージからバイナリパッケージを作成することもできる。rootになった状態で、tarボールを/usr/src/redhat/SOURCESにコピーし、「rpm --rebuild gthumb-0.7.1-1.nosrc.rpm」とすればいい。

## ●サムネイルと画像の表示

ktermなどから「gthumb&」として起動すると、左にディレクトリツリー、右にファイルとサムネイル一覧という構成のウィンドウが開く。ウィンドウサイズの初期値が小さいので、適当に拡大しよう（**画面1**）。

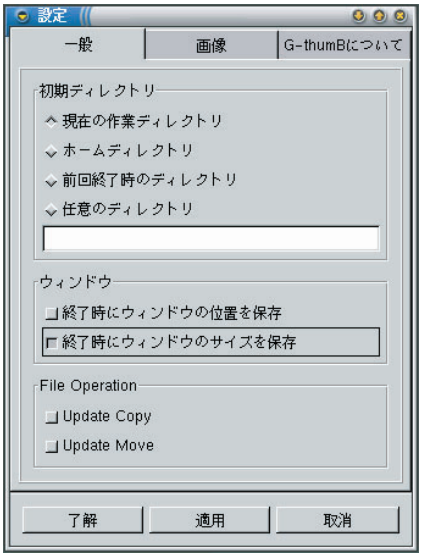
なお、ツールバーの[設定]ボタンで設定ダイアログ（**画面2**）を開き、[終了時にウィンドウのサイズを保存]をチェックすれば、次回からは現在のサイズでウィンドウが開くようになる。

サムネイルを作成するには、[表示]－[サムネイル]を選択するか、Ctrl－Tキーを押す。G-thumBには、このようなショートカットキーが豊富に用意されており、ほとんどの操作をキーボードから行える。

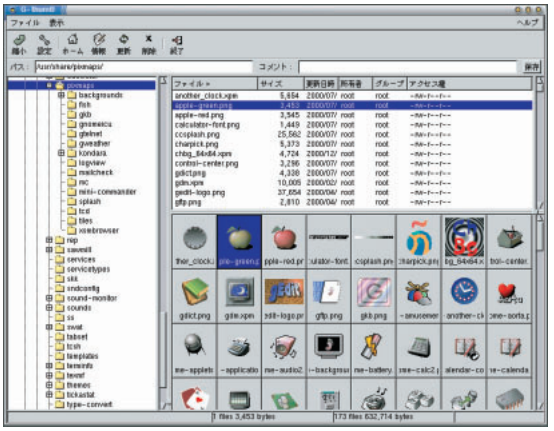
サムネイルをクリックするか、画像ファイルを選択してCtrl－O（オー）キーを押すと、実際の画像が別ウィンドウに表示される。前後の画像に切り替えるにはBS／スペースキー、ウィンドウを閉じるにはEscキーを押せばいい。このほか、他の画像を隠したフルスクリーン表示や、次々に画像を切り替えるスライドショーなども可能だ。

Web公開用のHTMLファイルを作成するには、対象となる画像ファイル（複数可）を選択後、[表示]－[HTMLカタログ作成]を選択するか、Ctrl－Cキーを押す。画像と同じディレクトリに、index.htmlと画像ごとのHTMLファイルが作成される。このファイルをWebブラウザで開くと、Webページ内に画像のサムネイルが一覧表示され（**画面3**）、サムネイルをクリックすると、実際の大きさの画像に切り替わる。

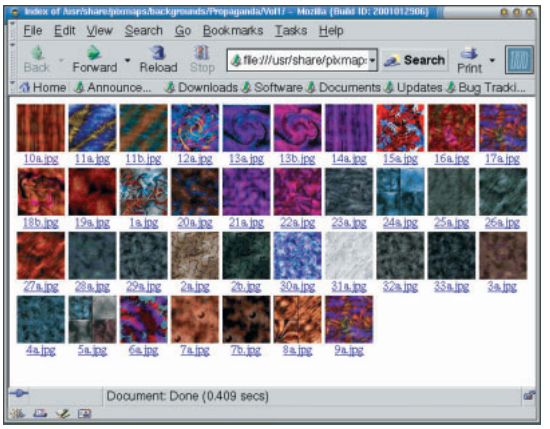
こうして作成したHTMLファイルは、大量の画像ファイル（すべてのファイルフォーマットには対応してないが）を整理するのにも有効だ。




画面2
設定ダイアログでウィンドウサイズの保存を有効にしておこう。



画面1
ディレクトリツリーやサムネイルが並ぶメインウィンドウ。




画面3
サムネイルが並んだWebページのHTMLファイルを生成できる。

Software Catch Up

根強いファンと、高い精度を持つ日本語入力メソッド

# VJE-Delta Ver.3.0 for Linux/BSD

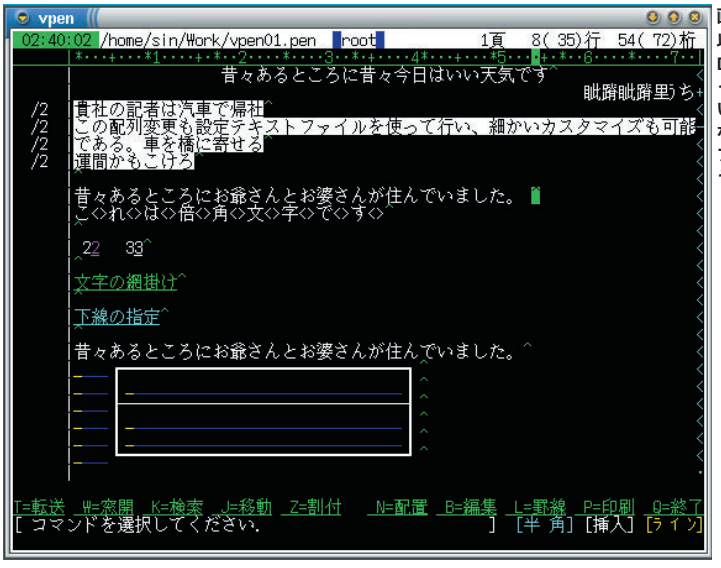
**VJE-Deltaは、MS-DOS時代から高い変換精度を持つといわれていたVJEのLinux/BSD版だ。この製品には、日本語ワープロであるVJE-Pen for Linux/BSDもバンドルされている。**

文：塩田紳二
*Text : Shinji Shioda*

価格　5800円
問い合わせ先　バックス
042-724-9200
http://www.vacs.co.jp/

VJE-Delta Ver.3.0 for Linux/BSD（以下、VJE-Delta）は、LinuxやFreeBSDに対応した日本語入力メソッド（IM。あるいは、かな漢字変換フロントエンド＝FEPといったほうがわかりやすいかもしれない）である。

かつてPC上では、ATOKと並び、メジャーなかな漢字変換FEPであったVJEは、UNIXにも比較的早い時期から対応した。これは、変換サーバと変換用クライアントに分かれた構成となっていて、現在のUNIX用IMの基本となるものでもあった。現在のかな漢字変換方式のように読みがなを入力して、漢字に変換するモジュールを汎用化し、アプリケーションからは、OS側の機能のように見えるものになったのは、PC用のソフトとして登場したのが初めだった。しかし、当時のOSであるMS-DOSは、シングルユーザー、シングルタスクであったため、かな漢字変換モジュールが辞書ファイルなどを直接操作することができたが、UNIXでは、マルチユーザー、マルチタスクであるため、辞書ファイルのアクセスなどを1カ所で行う必要があるため、現在のような変換サーバと、アプリケーション側の補佐を行う変換クライアントという方式が採用されたわけである。




画面1　VJE-Pen
以前にVJE-PenやMS-DOS上のエディタを使っていた人には懐かしい画面構成だ。言われないとMS-DOS上のソフトだと思ってしまう。

またこのVJE-Deltaには、やはりMS-DOS上で人気を博したワードプロセッサである、VJE-Pen for Linux/BSD（**画面1**。以下、VJE-Pen）がソースコード込みで含まれており、テキストのみの扱いではあるものの、文書作成を簡単に行うことが可能になっている。

なお、発売元のバックスは、すでに次期バージョンであるVJE-Delta Ver.4.0のβテストなどを行っており、近いうちにバージョンアップが行われる可能性がある。

## VJE-Deltaのパッケージとインストール

VJE-Deltaのパッケージに含まれるCD-ROMには、1枚にLinux用とBSD用のバイナリなどが含まれている。Linux用にはtgz（tar＋gzip）、rpm、deb（debian用パッケージ）の3つが含まれており、多くのLinuxディストリビューションへの対応が可能となっている。

インストール手順などは、CD-ROM上のオンラインドキュメントに記載されているが、過去のバージョンをインストールしていなければ、単にrpmな

どをそのままインストールするだけである。

なお、VJE-Delta附属のツールはJavaで記述されているため、JDKもしくはJREが必要となる。

このとき必要なJavaは、Java 1.2.xもしくは、1.1.xで、Java 1.3.xには対応していない。Java 1.3.xのみをインストールしている環境だと、文字入力ツールなどを起動すると、Javaがインストールされていない旨のメッセージが表示されてしまう。ツールの起動は、/usr/local/vje30/bin/vjeaccというシェルスクリプトから行われるのだが、ここでJavaのバージョンを取得したあとの処理がJava 1.2.xまでであり、Java 1.3.xのJavaは違うものとして処理されるからである。ちなみに、無理やりJava 1.3.xでツールを実行させてみたが、やはりうまく動かないようである。このあたりは、次期バージョンに期待というところだろう。

VJE-Delta Ver.3.0は、昨年1月に発売され、すでにアップデートファイルの配布が行われている。これは、バックスのWebサイトからダウンロードが可能である。しかし、現在発売されている入手可能な製品パッケージがアップデート済みのものかどうかを確認することはできなかった。とりあえず、インストールしたら利用する前にアップデートファイルを適用しておいたほうがいいだろう。

## VJE-Deltaの構成

かな漢字変換を行うVJE-Delta本体は、変換エンジンであるvjedと変換のフロントエンドであるvjeから構成されている。利用にあたってはまず、vjedが起動されており、かつ、vjeがIMとして指定されていて、起動していなければならない。

簡単には、環境変数"XMODIFIERS"を、

```
XMODIFIERS=@im=vje
```

としておく。こうしておいてvjeを起動したのち、同じ環境変数を引き継ぐ形でアプリケーションを起動すれば、起動したアプリケーションはVJE-Deltaを使ってかな漢字変換が可能になる。VJE-Deltaが動作するためにはこれだけの手順でよいので、システムのデフォルトのかな漢字変換とは独立して、特定のアプリケーションでのみVJE-Deltaを使うといったことも可能になっている。

VJE-Deltaは、標準ではIM切り替え設定などを自動で行わない。必要ならば、ユーザーがvjedやvjeの起動を行う必要がある。ただしVJE-Deltaは、XIMのみをサポートしているため、XIMに対応していないアプリケーションでは利用することができない。また、ktermやrxvtなどは、XIMの実装の違いからソースコードを修正する必要がある。このため、現時点では、CannaやFreeWnnなど、kinput2x経由で多くのアプリケーションから利用できるIMを起動しておき、別途、vjedやvjeを起動し、VJE-Deltaを使いたいアプリケーションの環境変数XMODIFIERSを変更して利用するほうがよいだろう。

VJEにはこのほか、辞書関連のツールとして以下のようなコマンドラインツールが含まれている。

| | |
|---|---|
| **vadd** | **単語登録ツール** |
| **vdel** | **単語削除ツール** |
| **vdispd** | **辞書内容表示ツール** |
| **vmaked** | **辞書作成ツール** |

それぞれコマンドラインから使う辞書メンテナンス用のツールで、引数を指定しなければ、必要な情報を質問してくる形になっている。これらは、古くからのPCのユーザーであれば、かつてMS-DOS版のVJEに附属していたものとよく似ていることに気が付くかもしれない。

また、VJE-Delta関連のユーティリティとして、vjeクライアントプログラムの停止コマンド（vjekill）があり、これは起動しているvjeを、設定ファイルの再読み込などのために終了させるプログラムである。

また、ソースコードの公開されているワープロソフトVJE-Penは、

| | |
|---|---|
| **vpen** | **VJE-Pen本体** |
| **vpr/vprc** | **文書印刷プログラム** |
| **vpu** | **日本語ソートプログラム** |

から構成されており、通常はvpenとい

| 種別 | プログラム名 | 概要 |
|---|---|---|
| VJE-Delta | vjed | VJE-Delta変換サーバ |
| | vje | VJE-Deltaクライアント |
| | vjekill | VJE-Deltaクライアント終了プログラム |
| VJE-Pen | vpen | VJE-Pen ワードプロセッサ |
| | vpr | ラインプリンター用印刷モジュール |
| | vprc | ページプリンター用印刷モジュール |
| | vpu | 日本語対応ソートツール |
| 辞書ツール | vadd | 単語登録ツール |
| | vdel | 単語削除ツール |
| | vdispd | 辞書内容表示ツール |
| | vmaked | 辞書作成ツール |

表　パッケージに含まれるプログラム

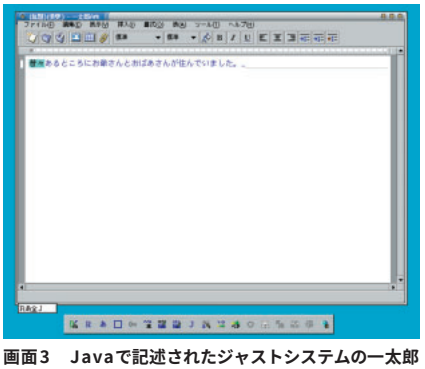
うプログラムを起動すればよい。ただしこのvpenは、日本語入力メソッドのプロトコルとしてXIMのみに対応しており、基本的にはVJE-Deltaと組み合わせて使う。

VJE-Deltaはもう1つ、Javaによるアクセサリが附属しており、VJE-Deltaウィンドウなどから起動が可能になっている。これは、部首検索や記号入力、文字コードからの入力などを行うツールだ。こちらも、前述のようにJava 1.2.xが必要である。ただし、この機能を使わないのであれば、必ずしもVJE-Deltaの実行にJavaが必要というわけではない。Javaがなければ単に機能ウィンドウが表示されないだけとなる。

## VJE-Deltaのユーザーインターフェイス

VJE-Deltaは、漢字キー（日本語106キーで動作しているとき）、あるいはCtrl＋Spaceでオン／オフを行う。また、各種モードの切り替えは、キーストロークやVJE-Deltaウィンドウ（**画面2**）のボタンで行う（VJE-Deltaウィンドウからも変換モードのオン／オフは可能）。

画面2　VJE-Deltaウィンドウ



画面3　Javaで記述されたジャストシステムの一太郎Ark上での入力

X Window Systemの国際化入力では、Root方式（IMの入力途中文字などの編集領域と状態表示が、アプリケーションウィンドウの外側に配置される）、Over the Spot方式（編集領域は現在の入力位置、状態表示はアプリケーションウィンドウ内部の固定した場所に置かれるが、表示はIM側が行う）、Off the Spot（編集領域、状態表示ともにアプリケーションウィンドウ内部の固定した位置に配置される）、On the Spot方式（入力途中文字列や状態表示はすべてアプリケーションが行う）の4つがある。このVJE-Deltaでは、この4つすべてがサポートされている（ただし、どれを使うのかはアプリケーション次第である）。特にOn the Spot方式がサポートされたため、Javaアプリケーションに対する日本語入力が可能になった（**画面3**）。

入力を開始すると、カーソルキーやファンクションキーは、基本的にはVJE-Deltaの管理下となり、変換関連の操作や入力モードの変更などは、Ctrl＋英字、Ctrl＋ファンクションキーなどの組み合せで指定できるほか、マウスでVJE-Deltaウィンドウ内のボタンを操作しても行うことができる。

いつくかの設定はVJE-Deltaウィンドウで可能だが、基本的な設定は各ユーザーのホームディレクトリに作成される、「general.sty」ファイルで行われる。

これは、テキスト形式のファイルであり、編集などは別途エディタやVJE-Penなどで行う。また、ATOK、Microsoft IME、WXG、Wnn、Cannaの制御キーと互換性のあるキー操作を実現するためのキーファイルや、ローマ字変換テーブルファイルなどが用意されており、これらを使うことで、従来使っていた変換システムと同様の操作性にすることが可能になる。さらにファイルを編集することで細かいカスタマイズも可能である。

キーボードは、

**OADG 106キーボード（JP106）**
**AT 101キーボード**
**PC-9800シリーズ用キーボード**

の3つから選択でき、さらにそれぞれのキーボードに対してキー配列をASCIIやJIS、DVORAK配列として扱うことができるようになっている。それ以外にも、「アイウエオ」配列や「ABC」配列といった配列で使うこともできるが、特殊な打鍵方法を使う配列を定義することはできず、単に通常のキーボード（ShiftやCtrlキーなどによる修飾のみ）でキー配列のみを変更するだけの機能である。

この配列変更も、設定テキストファイルを使って行い、細かいカスタマイズも可能である。ただし、この配列が有効になるのはVJE-Deltaで入力を行っている間のみである。

VJE-Deltaウィンドウはいわゆるツールバーになっており、ほとんどのボタンはトグルボタンか、いくつかのモードがクリックされるたびに変更されるボタンとなっている。また、同時にこのボタンはインジケーターを兼ねており、モードや機能のオン／オフの状態を見ることができる。このボタンが小さいと状態がわかりにくく、逆に大きすぎると邪魔になるが、これについては前述のgeneral.styファイルで設定が可能で、カラーやモノクロ表示の切り替え、サイズの変更が可能である。

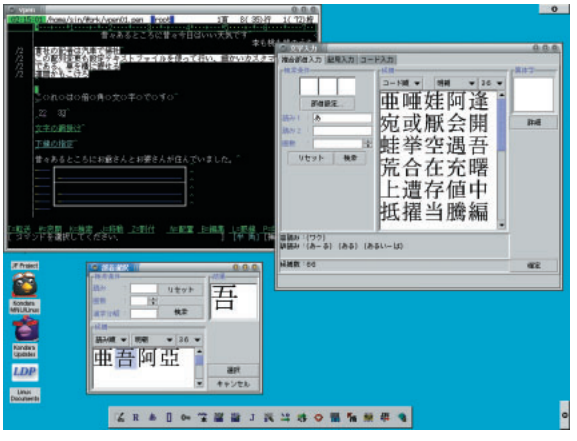
文字入力ツールは、「複合部首入力」、

「記号入力」、「コード入力」などにより（**画面4〜6**）、読みがなのわからない文字などを入力するためのものだ。たとえば、複合部首入力では、部首や音訓、画数などで文字の検索が行え、候補文字の中から該当の文字を探すことができる。また、選択した文字について詳細情報（文字コードなど）を表示する機能（**画面7**）もある。

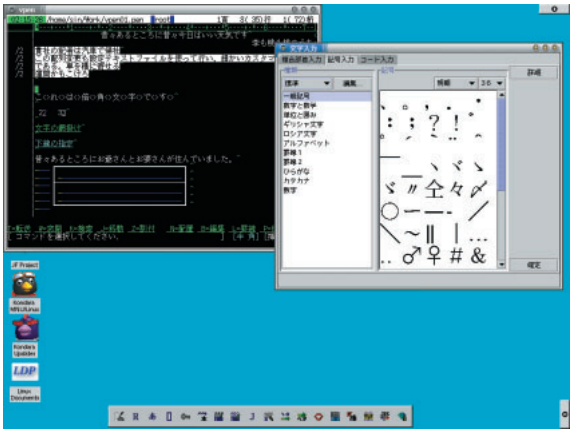
## VJE-Deltaの使い心地

かつては、ATOKと1、2を争うといわれた変換エンジンを使っているためか、文章の変換精度はかなり高く、少々長い文章を入れても、不自然でない変換結果が得られる。日本語の特質により、文章からは同音異義語を特定できないことがあるわけだが、そうした部分を除けば、ある程度正しく変換できると安心して使えるわけである。かつて筆者がMS-DOSで使っていた頃、VJEはATOKと比べると、辞書にない単語が連続するような、かな漢字変換エンジンにとって不利な文章を入力する場合でも、自然な文節区切りが行われていた印象がある。ただし最近は、当時に比べると辞書ファイルのサイズが格段に大きくなっているし、変換のために利用できるメモリもかなり大きい。そういう意味では、いままでに鍛えられた辞書を持つぶん、ほかのUNIX由来のかな漢変換エンジンよりは有利なところもあるだろう。

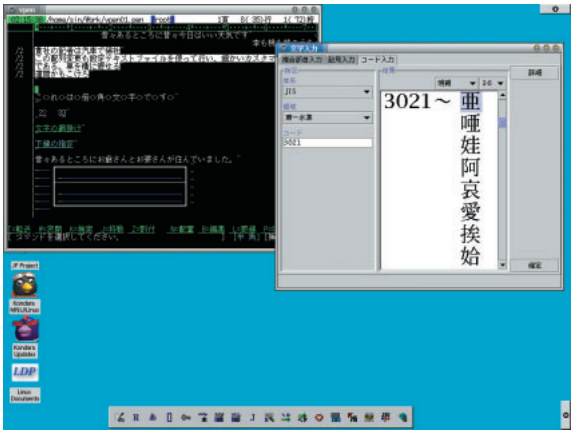
それでは、現在Linuxの上で動いているATOKと比べてどうなのか？　というと、これはかなり難しい比較となるだろう。カタログスペックから推察すると、直前までに確定した文章を記憶し、それらを元に同音異義語の選択を行うATOKが有利な部分はあるが、たとえば、おもに単文節や、2〜3文節程度入力して変換を行うような使い方だと、辞書を使い込んだ状態ではあまり差が出ないのではないかと思われる。ATOKはどちらかというと、一度に数文節以上、あるいは1文を入れての変換は効率が良いように見えるが、単文節ごとに変換を繰り返すような使い方では、ときどき大きな変換ミスが出るような印象がある。これに対して、VJEはこうした入力方法でも、あまり誤変換したという印象がない。これはきちんと測定したわけではなく、ざっと使った感覚である。しかも、単文節に区切って入力するのは、筆者のように単文節変換時代からワープロを使い続けているユーザーに多い入力のスタイルであり、最近のように、1文まるごと入れても何の問題もない漢字変換エ



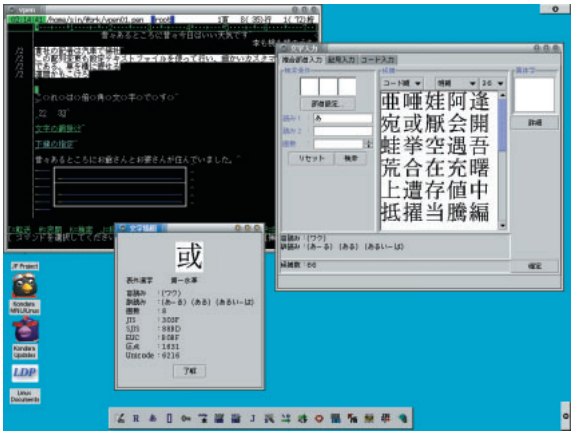
画面4　複合部首入力
部首による漢字の入力方法だ。読みがわからない漢字の入力に便利である。



画面5　記号入力
記号をジャンルから選んで入力することができる。こういった記号を使って、装飾を行うこともできる。



画面6　コード入力
現在ではほとんど使われない機能かもしれないが、固有名詞で使われる特殊な漢字を入力することができる。



画面7　文字情報
漢字の文字情報が表示できる。漢字コードや読みなどが表示されるので、正しい漢字かどうかが判断できる。

ンジンを最初から使っているユーザーとは違ったスタイルであるため、評価の基準にはならないかもしれない。

ただ、細かい使い勝手という点でいえばこれからという感がある。このバージョンでは、文字入力ツールがGUIで提供されたが、辞書関連ツールなどもGUI化してほしいところである。また、JavaによるGUIツールは、速度などの点や、UIとしての問題は感じないが、できればGTKなどを使うネイティブコードのプログラムにしてほしいところだ。JDKやJREはそんなに小さいものではないし、日常的にJava 1.3.xを使っているにも関わらず、VJE-DeltaのためだけにJDK 1.2.xを入れるのもムダな感じがする。

## VJE-Pen for Linux/BSD

VJE-Penは、かつてMS-DOS上で広く使われていたVJE-Penを、LinuxやBSDに移植したものだ（**画面8**）。VJE-Penは、テキストベースのワードプロセッサで、罫線図形による簡易作図機能を持つ。現在の感覚でいえば、「文字属性や禁則機能付きで、印刷機能を持つエディタ」という感じのソフトである。

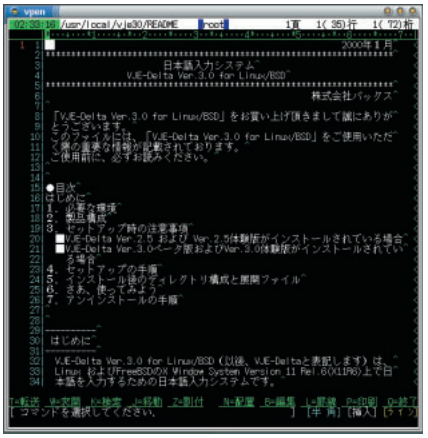
文字属性には、半角、全角、倍角、1/4角（上付きまたは下付き）やアンダーライン、網掛けといった修飾機能がある。右寄せやセンタリングは、スペースなどを使って、文章の開始位置を変更するもので、行の属性として設定するわけではない。また、各種属性も文字色などで表示される（**画面9**）。

以前ここで紹介した、オムロンソフトウェアのdp/Noteなどと比較的よく似た作りになっている。MS-DOS時代の多くのパソコンは、テキストモードと呼ばれる画面モードを持ち、ビデオメモリ上に文字コードを書き込むことで直接文字表示が行えた。VJE-Penはそうした時代のワープロである。のちに、MS-DOS上でもグラフィックモードを使って、細かな表示が行えるソフトが登場したが、VJE-Penは速度が遅くなるこうした方向には走らず、しばらくテキストモード専用のままのソフトであった。このため、軽快な入力を望むユーザーが使い続けたワープロソフトでもあった。しかし、Windowsが普及し、CPU速度が向上してWindows上でもかなり軽快な入力ができるようになった時点で、文字の細かいフォント指定やグラフィックの扱いが可能なWindows用アプリケーション（当時は、Microsoft Wordや一太郎だったのだが）に取って代わられるようになってしまった。このVJE-Penは、ある意味そのリバイバルとでもいうべきソフトである。

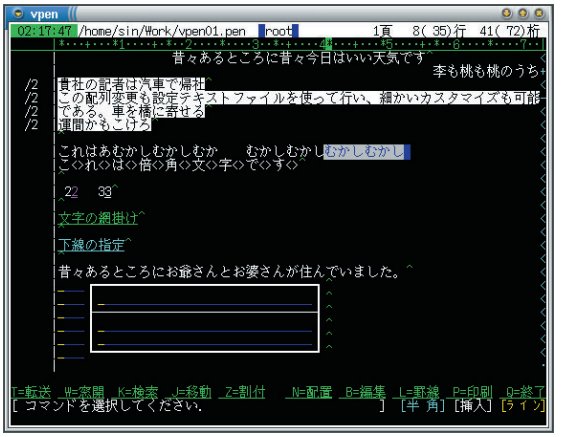
画面構成などを見ると、X Window System上にMS-DOSのテキスト画面モードと同じような動きをするウィンドウを作り、その中でテキストベースで動作しているようである。

なお、VJE-Penのソースコードが製品に附属しているため、ユーザーが自由に書き換えて利用することも可能である（ソースコードを理解する技量が必要だが）。やろうと思えば、フレームバッファなどにも移植が可能なのかもしれない。また、ちょっとソースコードを見た限りでは、「curses.h」などいうものもあったので、ターミナルなどでも動作できるのかもしれない（あくまでもちょっとソースコードを見た限りという話なのだが）。

VJE-Penは、Ctrlキーと英文字の組み合せや、ファンクションキーなどで操作を行う。たとえば、ファイル関係のメニューを開くには、ファンクションキーのF1キーを押す。すると、ファイル関連のメニューが表示され、ここではカーソルキーの上下または、メニューに書かれているアルファベットでメニュー選択が行える（**画面10**）。同様の操作をCtrlキーで行うには、Ctrl＋Tキーを押す。すると、今度はウィンドウ下の2〜3行に次に選択可能な機能がガイドとして表示され、機能



**画面8　VJE-Pen**
見た目は単なるエディタのような画面構成となっている。



**画面9　各種属性**
各種属性が色や記号によって表現されている。そのぶん、動作は軽快だ。

名の前に表示されているアルファベットキー（Ctrlキーとの組み合せでも同じ）を押すことで、それぞれの機能が選択できる（**画面11**）。この2つの機能は似ているのだが、実際にはCtrlキーを押したときのほうが多くの機能が表示される。なお、ファイル名の入力用にディレクトリの表示も可能で、そこからファイルを選択できる（**画面12**）。

このユーザーインターフェイスでは、不慣れなうちはファンクションキーとメニューによる操作を行い、少し慣れてきた段階でCtrlキーと表示されるガイドを見ながら行い、慣れたら画面を見ることなく、単にキーの組み合せのみを使って操作するといったことが可能になる。

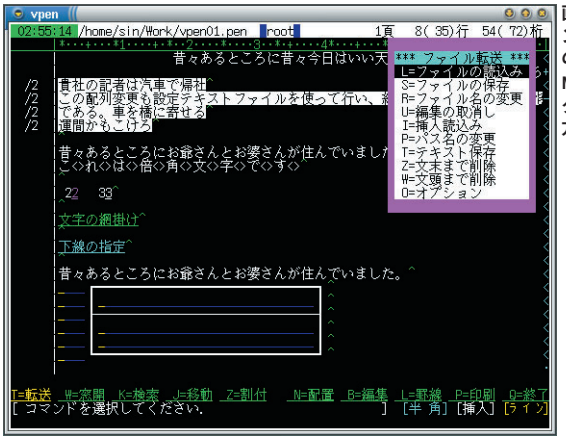
VJE-Penのカスタマイズは、起動オプションで行うことができる（**画面13**）。ただし、多くの機能は、VJE-Pen中の設定でも可能で、こちらで設定した場合には設定が保存される。なお、表示行数の指定や、表示色のカスタマイズなどは起動オプションでしか設定ができない。

印刷機能は外部プログラムを利用して行うが、そのパラメータ設定は、VJE-Pen内部で行える。

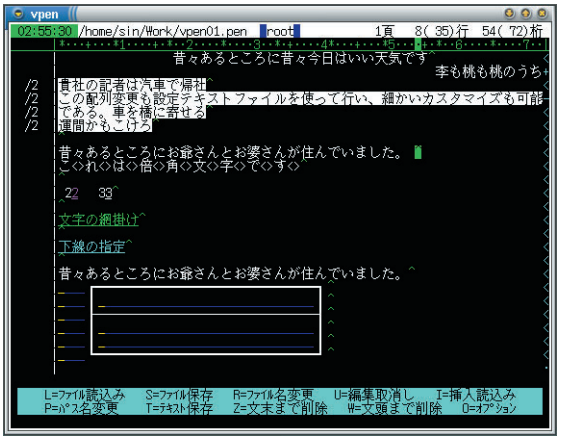
VJE-Penは、最近のWindows用ワープロに慣れた身には、かなり厳しい仕様とも思えるが、エディタに印刷機能が付いていると思えば、そこそこの使い勝手である。また、VJE-Deltaと密に動くため、たとえばVJE-Deltaをオンにして文章を入力している途中であっても、別のファイルを開くためにファイル名の入力を行う場合には、自動的にVJE-Deltaがオフになる。単に日本語入力を可能にしているアプリケーションとは、このあたりが違っている。また、オープンソースなので、気に入らない、機能が足りないという場合には、自分で改造することもできる。ある意味、プロ向けとも言えるだろう。

## 雑感

VJE-Deltaは、VJE-Penとそのソースコードまで付いて、パッケージ価格で5800円は安いとは思うし、安定した変換を望むなら導入を考えてもいい製品だろう。ただ、現時点でいうと、ATOKとどちらがいいのかと聞かれるとちょっと「？」となってしまう。そろそろ、次バージョンが登場する頃なので、新バージョンの登場を待ってから検討を始めるのがいいかもしれない。

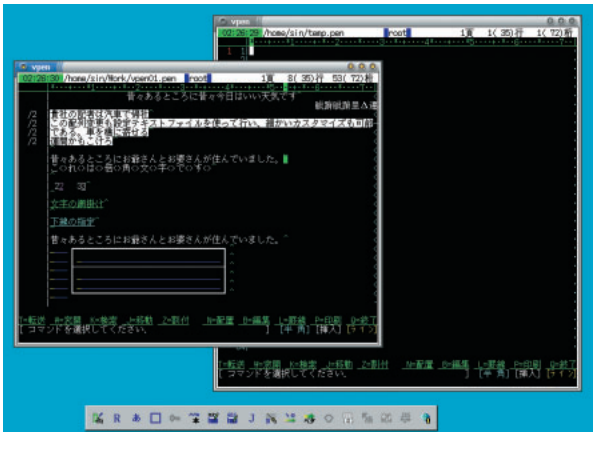



画面10　ファンクションキーによるメニューの表示
MS-DOS時代のエディタソフトと同等の操作方法だ。




画面11　Ctrlキーによるメニューの表示
画面下に、次に選択できる機能がガイドとして表示される。


画面12　ディレクトリの表示
この画面でファイルを選択することが可能だ。かなり懐かしい感のある画面だ。



画面13　起動オプションを指定してVJE-Penを起動
起動オプションを指定し、ウィンドウサイズを変更して起動した（右）。

# 隠喩としてのコンピュータ

## P2Pはクライアント／サーバを超えられるか？

文：豊福 剛
*Text：Tsuyoshi Toyofuku*

プログラムにはバグがつきものであることは、昔も今も変わらない。

WindowsでNetscape Communicator 4.73を使っていたら、<font>タグで囲まれている部分が日本語フォントで表示されないという奇妙な現象があった。HTMLは次のようなパターンになっていた。

```
<body>
<img src="hoge.jpg">
<font size=+1>ほげほげ</font>
たらたら
ほげたらぽん
  :
```

これをローカルで開いて表示すると、ちゃんと日本語フォントで表示されるのだが、サーバにアップして、リモートに開いて表示すると、「ほげほげ」の部分が英語フォント表示になってしまうのだ。<font>だけでなく、<a>、<b>、<h＊>、<form>で指定した部分もすべて英語フォント表示になってしまうのである。

原因は<img>タグにあった。hoge.jpgがサーバ上になかったことと、画像のサイズ指定を省略したからだったのだ。

わたしはこういう現象があると楽しくなってしまうタイプの人間である。<img>タグのalt属性に日本語を指定してみた。サーバに画像がない状態だと、altで指定した文字列は、一瞬日本語で表示されたあと、やはり英語フォント表示になってしまうのである。

ブラウザのレンダリングまわりのコードは、Netscapeに限らず、いまや途方もないバロック状態になっていて、あちこちに魑魅魍魎が潜伏しているのだろう。

### ブラウザ・フォンにもバグはある

ブラウザのバグといえば、23万台もの回収となったNTTドコモのP503iだが、画像を縮小表示する機能にバグがあったらしい。そのバグにひっかかると、携帯電話の電源が落ち、アドレスデータや保存メールなども削除されてしまうという。携帯電話の組み込みシステムとなると、パッチを当てて修正というわけにもいかないのが致命的だ。iモード端末では、ほかにもKO209i、ER209i、SO502iWMのバグが続いていたところだった。

こういう現象が続くと、少しばかり重苦しい気分になってしまう。組み込みシステムである以上、ハードウェアやソフトウェアに余裕がないのは前提であるとはいえ、おそらく背景には、その開発プロセスに想像を絶する苛酷な状況があるのだろう。その苛酷さが伝わってくるから、重苦しい気分にさせられるのだ。

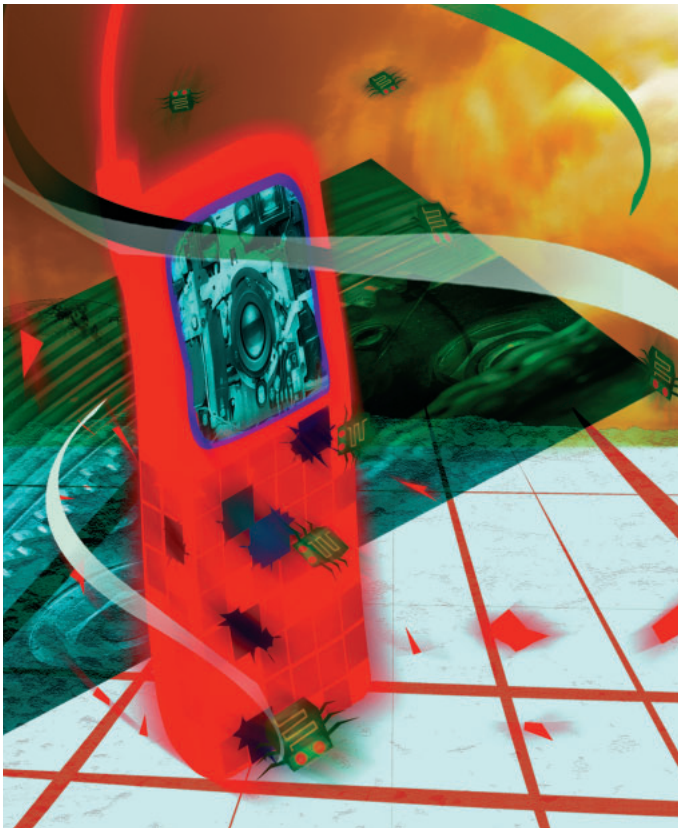
バグのあった携帯電話が回収されたあとの処理も気になるところだ。プログラムだけ書き換えるというのが可能なのか、部品交換になるのか、それとも廃棄処分になるのか。そもそも携帯電話というのは、大量生産、大量消費、大量廃棄の典型のような商品である。電池などを中心にして、一部リサイクルの試みもあるようだが、その多くはどこかに山積みされて、粉々に粉砕されてしまうのだろうか。

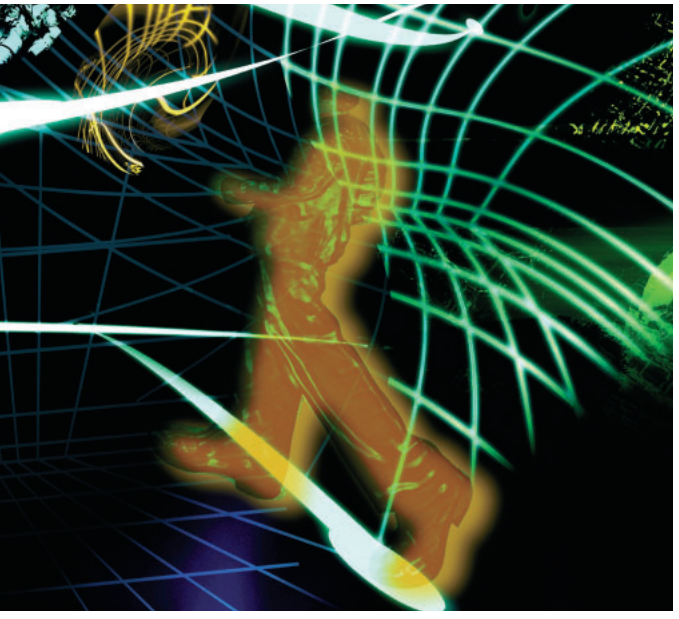
そんなことまで考えさせられるほど、携帯電話のプログラムのバグの与える経済的影響には、計り知れないものがある。しかし、プログラムが複雑になればなるほど、厄介なバグが潜む可能性も増大するわけで、どんなに綿密なテストを実行したとしても、限界はある以上、プログラムの書き換えで対処できるような構成にならないものかと思う。

この数年SOHO向けのルータを使ってきたが、ファームウェアを更新したことが何度かあった。だからといって、ルータのプログラムを作成したエンジニアが手抜きをしていたとは思えないのだ。ルータのファームウェアは更新できて当然であるし、もし、それができないようなルータがあったら、そのほうが問題である。ルータも携帯電話もコンピュータ応用製品であり、白物家電とはプログラムの複雑さのオーダーが違うはずだ。それだけに、携帯電話のプログラムが完璧であることを前提にするのは無理があると思うのだ。

## クライアント／サーバの深層心理

携帯電話のプログラムの複雑さが一方にあるとしたら、他方にはサーバ側の配置の複雑さがあるのだろう。最近はラックマウント型サーバ製品の広告が目立つのだけれども、データセンターと呼ばれるようなところには、ビルのフロア全体を膨大な数のサーバが埋め尽くしているのだろう。メインフレームに対抗する形で登場してきたワークステーションやパソコンが、インターネットという分散型のネットワークにおいて、ある特定の場所に集積されて配置されるに至るというのは、なんとも皮肉な歴史の展開である。





もともとパソコンが誕生した背景には、個人がコンピュータを所有したいという欲求があったのだと思う。道具が身体の延長であるとしたら、パソコンも身体の延長として考えることができる。それと同時に、パソコンは空間でもあった。その空間にはデータを置くことができるだけではなく、その空間に対して操作を行うことで、空間からの反応が返ってくる、インタラクティブな環境でもあったのだ。

それでは、パソコンがネットワークにつながれたときには、何が変るのだろうか。それまでは閉じていた空間が、別の空間と連結可能になることによって、開かれた空間に変るのである。ただし、連結されたからといってひとつの空間になるわけではなく、ある空間と別の空間には境界があり、それぞれの空間は名前やアドレスによって区別できるのである。

ある空間と別の空間を区別することができるのは、名前やアドレスが割り当てられているからであることは重要である。この名前やアドレスは、基本的には相互契約によって割り当てられるのであるが、連結する空間の数が増えるにしたがって、相互契約を結ぶ代わりに、名前やアドレスを設定する機能を代理するものが要請される。TCP/IPの場合、DNSやDHCPがこれに相当する。

パソコンを所有している感覚そのものは変わらないし、自分の空間を自分でコントロールすることが可能であることも変わらない。また、ある空間から別の空間に対して、何が操作できて、何が操作できないかは、それぞれの空間がいかなるインターフェイスを公開しているかによって決定される。そして、空間と空間の間で契約されるインターフェイスの内容によって、ネットワークの形態が決定される。そのような形態の代表的なものとしてクライアント／サーバがある。クライアントには、自分の空間に対するアクセス要求には対応したくないが、別の空間に対するアクセスは要求したいという欲求がある。そしてネットワークを構成するノードが、等しくこのような動機を持つとき、他からのリクエスト要求に応答することだけを行うサーバが要請されるのである。このようにしてクライアント／サーバは誕生する。

## 集中か分散か、それが問題だ

クライアント／サーバにおいて、クライアントが自分の空間に対するアクセス要求に対応したくない理由は何か。

それは自分のリソースを占有したいからであり、自分の都合に合わせて自分のマシンをコントロールしたいからである。クライアントにとってサーバは、自分のためにリソースを提供してくれる非常に都合のよい存在だ。

そしてクライアントは、リソースやコントロールの所有をさえ、サーバに委ねるに至る。それらがここになければならない必然性は、ネットワークの浸透によって薄れる。ディスプレイとキーボードがあればよく、ディスク・ストレージはリモートであっても構わないし、コンピューティングもリモートで実行されることを望むのである。クライアントはもともとコンピュータを所有することなど望んでいなかったのだ。

サーバにはリソースが集中する。リソースには、ある特定のクライアントだけにアクセスが許されたものもあれば、不特定多数からのアクセスを前提にしているものもある。サーバは24時間365日常時稼動しているので、いつでもどこからでもアクセスできる利便性がある。多くのアクセスを期待するリソースが、アクセスが集中する特定のサーバの空間に集中することによって、さらにアクセスの集中を増大させる。

このようなクライアント／サーバとは別の形態として、P2P（ピアトゥーピア）も成立しえる。P2Pにおいては、別の空間に対するアクセスは要求するものの、同時に自分の空間に対するアクセス要求にも対応する点に、クライアント／サーバとの違いがあるといえるだろう。クライアント／サーバでは、インターフェイスを公開しているのはサーバだけであるのに対して、P2Pではネットワークのノードが平等にインターフェイスを公開しているのだ。

それではP2Pに期待が寄せられているのはなぜか。理由は、サーバへの過剰な集中による弊害を分散化によって緩和するためであるだろう。クライアント／サーバにおけるサーバの視点からP2Pを見た場合、それはサーバ機能をクライアント側に分散させる戦略と解釈できる。これをクライアントの視点から見た場合、クライアントがコンピュータを所有することを放棄するのではなく、自分のリソースを自分のためだけに使うのでなく、その一部を他者に対しても提供するということなのだ。

問題は、P2Pによる分散が有効なリソースとは何か、である。はたして、P2Pに適したデータベースというものは可能なのか。それともデータベースとは集中を本質とするものなのだろうか。

*Profile*

**とよふく つよし**

1962年生まれ。メディアデザイン研究所技術顧問。訳書に『Javaプログラムクイックリファレンス』『Java分散コンピューティング』（オライリージャパン）『GIMPパーフェクトガイド』（エムディーエヌコーポレーション）などがある。

オープンソース
・
アクティビズム

# 第三世界に無料のシステムを

文：安田幸弘
*Text : Yukihiro Yasuda*

## 捨てられるP5/200と現役486

先日、知人から「パソコンを買い換えようと思うんだけど」という電話があった。すっかり忘れていたのだが、10年ぐらい前、「パソコンを買いたい」というのでぼくがパソコン選びを手伝ったことがあって、その時に買ったパソコンを今までずっと使い続けてきたのだそうだ。

「長く使いたい」というので、少々割高だったが当時としては最新鋭の80286マシンにたっぷりメモリを積んで（といっても640K＋256KバイトのEMSとかそれぐらいだけれど）、ワープロの「松」を乗せたのだが、まさか10年前のパソコンをいまでも使っているとは思わなかった。……もっとも、インターネットだのデジカメだのと言わなければ、それで十分なのかもしれないけど。

これは極端な例かもしれないが、日本ではまだ使えるパソコンが次々と廃棄されている。ウチでもPentium 120MHzのマシンにFreeBSDを入れたサーバが現役だったりするのだが、最近は、Pentium 200MHz級のパソコンも廃棄の対象だという。これはさすがにもったいないと思うのだけれど、リースの契約だとか税金だとかの問題で、譲渡はできず廃棄しなけりゃいけないことになっているのだそうだ。

そうかと思うと、パソコンが足りなくて困っているところもある。先日も、ある国際ボランティア団体から「パソコンの調子がおかしい」というので見に行ったところ、古い486機のメモリ制御のあたりが壊れてしまったらしい。買い換えを勧めたのだが、「パソコンの予算が取れない」という。この種のボランティア団体だと、会社と違って情報化投資の予算が組めず、メーカーから寄付してもらった古いパソコンをだましだまし使っているのだそうだ。

Pentium 200MHzが廃棄される「IT革命」の時代に、16Mバイトのメモリを積んだ486機でWindows 95あたりを頑張って使い続けるNGOがあるというのも不思議な話である。

## 中古パソコン援助

しかし、それでも日本国内のNGOなら、知り合いからパソコンを譲ってもらうという手もないわけではない。ところがこれが海外となると、さらにややこしい。

知人に「途上国に古いパソコンを」というプロジェクトをやっているやつがいる。彼らのプロジェクトは、日本で廃棄された古いパソコンを海外のスラムに送り、コンピュータ学校を開設してパソコンの技能を習得させるとともに読み書きの教育を与えて就業機会を増やし、生活の向上を目指すというものなのだが、これが口で言うほど簡単じゃないらしい。なにしろほとんどの途上国は、コンピュータの輸入に関税がかかる。最新のパソコンだろうが10年前のパソコンだろうが、コンピュータはコンピュータ、まともに通関させると1台搬入するのに数百ドルかかることがある。ノートパソコンなら、「個人用だ」と言って税関を通るという手もあるんだろうけど、廃棄されるパソコンはほとんどがデスクトップなので、なかなかそうもいかない。しかも、「これはパソコンではなく、廃棄された機械」だというと、今度は廃棄物輸出の規制だかなんだかにひっかかって、それもできないという。

結局、対象国の政府に働きかけ、プロジェクトの主旨を説明して特別な許可をもらって通関するのだそうで、そこが彼らのプロジェクトのノウハウなんだそうだ。「デジタルデバイドの解消のために」と叫んで日本企業から途上国への多額の援助が行われているというのに、民間団体が古くなったパソコンを送ろうとするといくつもの難関を通過しなければならないというのも、ずいぶん変テコな話ではある。

## マイクロソフトの太っ腹

ところで彼らのプロジェクトだが、関税のほかに、実はもうひとつの難関があった。ソフトの問

題である。いくら古いパソコンだといっても、ライセンスの関係上、OSをつけて送ることはできない。かといって、何百台もの中古パソコンに、正規のライセンスを購入するなんてことは不可能だ。

しかし、この問題は比較的あっさりと解決したらしい。あのマイクロソフトが太っ腹にも、これらのパソコンで使うために総額数千万ドルに及ぶソフトウェアのライセンスを無償で発行してくれるのだそうだ。……その実体が数枚のCD-ROMとライセンス証書だったとしても、数千万ドルは数千万ドルだ。ピューリタン的な伝統の残るアメリカでは、マイクロソフトに限らず、成功した企業がこうした社会的な活動に多額の寄付をすることはよくあることなのだそうだが、マイクロソフトの寄付がなければ、このプロジェクトはただの廃棄物輸送プロジェクトにしかならなかっただろう。

もちろん、「なぜ無料のオープンソースを使わないのか」という意見もある。このプロジェクトが始まった1995年当時、オープンソースが今ほど成熟しておらず、誰もLinuxなんて聞いたこともなかったから、というのが一番素直な答えということになるのだが、いずれにしてもこのプロジェクトが有償ソフトを前提とするものであることに違いはない。

昨年、ある途上国のNGOのスタッフにこのプロジェクトの受け入れを打診したとき、彼の返事は「社会教育という側面が持つ意義はともかく、そのような教育活動もマイクロソフトの潜在的な顧客を増やし、結局、将来、途上国にとっては貴重な外貨や人材の流出を招くことになるだけだ」というものだった。

このレベルの話になると、半分政治的な主義主張の領域になってしまうので話はそれっきりになってしまったのだが、もちろんオープンソースでやれればそれに越したことはない。ただ、何十台、何百台というばらばらの中古パソコンにLinuxをインストールする手間だとか、日本人を含めて非英語圏の住民にとって自国語で使えるアプリケーションが不足しているという問題を考えると、政治的に正しいかどうかという以前に、「こりゃー無理だわ」というのが正直なところ。

## いつの日にか

しかし、その後、インドの通信系NGOのスタッフから、脱マイクロソフト、あるいは知的所有権ビジネスに対する対抗ムーブメントとして、非営利セクターへのLinuxの普及活動を始めたという話を聞いた（もっとも、本音は「だってタダなんだもん」というところにあるらしいのだが、まあそれも立派な理由だ）。

スラムでの教育とはちょっと主旨が違うけれど、この種のコンサルティングは結構あちこちで始まっているらしい。インストールの問題やらローカライズの問題にしても、彼らに言わせれば「Windowsだって、最初からすべての互換PCに何の問題もなくインストールできたわけでもないし、各国語のアプリケーションが揃っていたわけでもなかったじゃないか。だいたい、5年前に、Linuxが商業サイトで使われるようになるなんて、誰が考えただろう？」ということになる。目下の問題は、オープンソースの使い手がまだ少なく、なかなかサポートにまで手がまわらないということだそうで、技術スタッフはあの広いインドのあちこちを飛び回っているという。

もしかすると、例のプロジェクトも、スタートがもう少し遅ければオープンソースでスタートすることができたのかもしれないとも思う。日本や非英語圏では（人口の5%が英語人口のインドを英語圏と言えるかどうかはわからないが）、アプリケーションのローカライズという問題が残るものの、オープンソースをベースにした支援プロジェクトを立ち上げることができる日はそんなに遠くないのかもしれない。

*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*

**1・23 AOLジャパン、ドコモAOLへ**
**1・31 セガ、ドリキャス打ち切り**
**1・31 Intel、超低電圧版モバイルプロセッサ発表**
**1・31 ソニー銀行6月開業予定**
**2・10 TurbolinuxとLinuxCare合併前に人員削減か？**

さて、そろそろ新年度という時期なのだが、筆者のように会社勤めをしていないと、あまり年度という感覚が強く働かないので、ひと区切りという気が全然しない。

学生時代であれば、進学、進級、落第と毎年4月には区切りあったし、生活にも変化があった。

会社勤めをしていた頃は、やはり年度末が忙しく、そこでひと区切りといった感じだった。しかし、ライター稼業を始めてからは、こうした行事や生活の変化もなくなってしまい、区切り感がまったくなくなってしまったわけである。

なので、せいぜい、「春になったか」ぐらいを感じる程度。仕事は相変わらず忙しいので、年に一度ぐらいは、こうした区切り感も欲しいなぁと思うこの頃である（単に休みたいだけなんじゃない？）。

### IntelはIDFを控えている

Intelは、毎年2回、開発者向けにIDF（Intel Developer Forum）というコンファレンスを開催している。今年のIDFは2月26日から、前回は昨年の夏に行われた。毎回、ここでIntelはイベントの目玉として、いろいろな発表を行う。ここでの発表は、半導体関連の学会などの発表に比べると、製品化が近いものや、Intelが力を入れている製品などが中心。OEM各社にPCなどを製造するための技術を提供しているIntelのIDFでの発表は、今後の動向を考えるうえで重要なものだ。

前回のIDFは、「対Transmeta」という雰囲気が大きかった。実際、プレゼンテーションなどでは、ノートパソコンにおいてCPUは、いまやそれほど大きな電力消費をしていないとか、ユーザーはより大きな処理能力を望んでいるなどと言っていたが、そのあとTransmetaのCrusoeを日本メーカーの何社かが採用すると、「超低消費電力版CPU」の開発なんて話が出始め、1月末には実際の製品を発表、IBMがこれを採用した製品を登場させた。

発表後、すぐ製品を出荷できたのはIBMのみ。というのも、IBMは昨年6月に米国で行われたPC EXPOで、日本のメーカー（日立や富士通）と一緒にCrusoe採用のThinkPad試作機をTransmetaブースに展示していたのだが、実際の製品の採用を見送っていたのである。一説には、IBMは試作機を公開することで、Intelに揺さぶりをかけ、低消費電力CPUの開発を進めさせたのだとも、危機を感じたIntelが、低消費電力版CPUの早期提供を材料にIBMを説得したのだとも言われている。

どちらにしても、Intelは低消費電力版CPUを製品化することで、Transmetaと全面対決になったわけである。ただし、低消費電力とは、低い電源電圧で高いクロックで動作するCPUを意味し、これは実際には、従来の電源電圧ではより高いクロックで動作するCPUでもある。つまり、高クロック版のデスクトップ用CPUと超低消費電力モバイルCPUの差は、紙一重なのである。このためIntelは、高い金で売れる高クロック製品を、モバイル用の低クロック製品としても販売しなければならないというジレンマを抱えてしまうわけだ。

さて、そんなIntelだが、1月にネットワーク関連機器のメーカーでもあるXircomを買収している。Xircomは、モデムやイーサネットカード、最近では、VISOR用モデムなどワイヤレス機器にも力を入れている企業。日本でDataSlimとして販売されているRexの発売元でもある。こうした製品が今後

どうなるのかはIntel次第であるが、米国では、PCの売れ行きが落ちているにも関わらず、PDAの売れ行きは好調なので、こうした分野にIntelが注力するという方向の現れともいえるだろう。

そういえば、Intelは前回のIDFでXscaleを発表した。これは、現在のStrongARMの後継CPU。そして、Palmが来年にはARMアーキテクチャへの切り替えを計画しており、すでに携帯電話に多くのARMプロセッサが使われていることを考えると、今後の注目はIAよりもARMかもしれない。ARMを採用したPSIONなんかでLinuxが動いているが、そろそろこっちのほうも人気が高まるのかも（国内では入手が難しいのが難点だが）。

## Microsoftってまだ裁判してたんだ

すでに人々の記憶からも消えようとしているMicrosoftと司法省の裁判だが、現在、ワシントンDCの連邦控訴裁判所に場所を移し、いまも続いている。ここで、最初の裁判でJackson判事が出した結論についての論争が行われる予定。まあ、アメリカの裁判ではよくあることで、控訴したときに、最初の裁判のやり方がおかしかったと申し立てる方法である。

Jackson判事の言動の中にはMicrosoftに対する偏見が伺われるようなものがあり、これをテコに最初の裁判が無効となる可能性もあるらしい。

また、分割による是正命令についてもここで争われる予定だが、これについても見直しとして地裁に差し戻される可能性もあるという。重要な点は、違反があったかどうかの判断ではあるが、審理のプロセスに問題ありとなれば、違反の有無に関わらず、やり直しとなり、その結果、違う結論が出る可能性もないわけではない。

しかし、Microsoftにしても、たとえ差し戻しになったとしてもまったく無傷というわけでもない。実際、株価が下がったり、つきまとう悪いイメージなどがあるからである。なお、この控訴裁判所での判決は、4〜5月ぐらいに出るのではないかというのが大方の予想である。まあ、春には、また、ひとつの決着が付いてるというわけだ。

決着といえば、Microsoftに対してSUN Microsystemsが起こしていたいわゆるJava裁判だが、こちらも1月の24日に和解が成立した。SUNによれば、MSは、SUNに2000万USドルの和解金を払い、Java互換の商標を使うことを永久に禁じられるそうである。また、SUNは、Microsoftとのライセンス契約をうち切るとのこと。

とはいえ、すでにMicrosoft関連製品では、Javaは主要な言語ではなくなっており、逆にMicrosoftは、.NET構想では、Javaによく似ているけどJavaではないC#という言語を採用している。簡単にいえば、Javaという名前は使わないものの、そのコンセプト（たとえば、Microsoftは、共通ランタイムを使ってさまざまなアーキテクチャ上で.NET構想を実現可能だと主張している）は、使い続けるわけである。

また、MicrosoftがJavaから手を引いた以上、たとえサードパーティがJavaを提供したとしても、それは、公にされたAPIなどを通してしかシステムと関係できないのである。このため、OSに近い部分のプログラミングにJavaが使えないという可能性もある。今回の裁判は、逆にJavaをWindowsの中心部分から遠ざけるという結果にもなったわけだ。もっとも、だからといってJavaアプリケーションがすぐ動かなくなるわけでもないが……。

## AOLは日本ではドコモAOLに

さて、NTTドコモから出資を受けたAOLジャパンだが、社名をドコモAOLとした。NTTドコモは事実上AOLジャパンを子会社にしたのである。アメリカでの人気に比べると日本ではAOLは少々盛り上がりに欠けるところがあった。すでに多くのプロバイダがユーザーを獲得しており、インターネット接続＋コンテンツというAOLのやり方では、多くの顧客を獲得できなかった。

これからは、AOLは日本に対して、コンテンツなどのサポートはするものの、国内での主導権はNTTドコモが取るということになる。簡単にいえば、日本の実状に合わせて、AOLはコンテンツビジネスに集中し、ドコモはiモードや次世代携帯電話用のコンテンツを手に入れる、ということであろう。

だが、携帯電話の会社が行うビジネス、はたして自宅でダイヤルアップするようなユーザーに受け入れられるようになるのだろうか？　逆にAOLのコンテンツって、iモードの小さな画面でも面白いのでしょうかね？


IDF（Intel Developer Forum）で今年のIntelの動向が見える
http://www.intel94.com/idf/

正しきタコへの道

# 初級Linuxer養成講座

## 第19回　バックグラウンド実行とリダイレクト

**前回は、bash上でLinuxの「マルチタスク」機能を使う方法について説明した。しかし、実際に使い始めてみると、思ったとおりに動いてくれないことも多い。バックグラウンド実行をうまく使いこなすためには、以前に解説した「リダイレクト」機能をうまく組み合わせなければならないのだ。リダイレクトの復習も兼ねて、バックグラウンド実行のコツを習得しよう。**

文：竹田善太郎
*Text : Zentaro Takeda*

本号に掲載されているTVキャプチャカードの活用記事をまとめるために、キャプチャの実験をいろいろやっていたとき、実験用に使っているマシンのCPU（Pentium II 350MHz）の限界を感じたため、秋葉原にCPUを買いに出かけた。なにせ古いマシン（前々号でやり玉に挙げたNLXマシン）なので、最近主流のCoppermineタイプのCPUは使えず、旧式のKatmaiタイプのPentium III（以下「P3」）を探したのだ。ところが、パーツショップのCPU売り場を真面目に覗いたのが半年ぶりだったせいか、品揃えがすっかりさま変わりしているのに驚かされた。

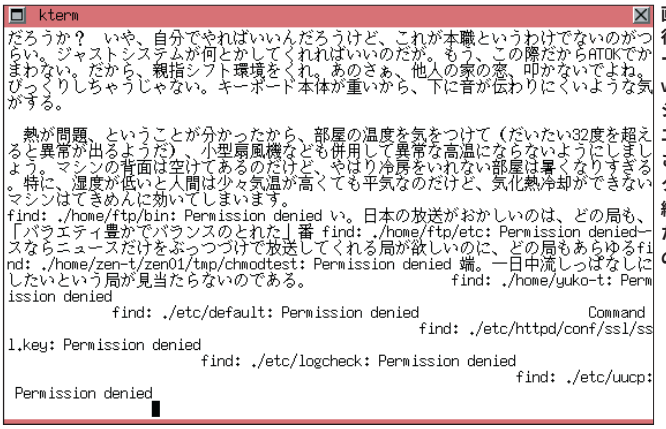
600MHz以下の旧式P3は、まったく姿を消していて、店頭のリストに並んでいるのは、700MHz以上のCoppermine P3やCeleronだけなのである。やっと1軒の小さなショップで600MHzのKatmai P3を見つけたものの、なんだか**不当に高い値段**がついていたので、買わずに出てきてしまった。マザーボードごと交換して、最新のCPUを使えばよいだけの話なのだが、NLXマシンなのでそういうわけにもいかない。

そこで、次善の策として**中古部品**を探すことにした。しかし、中古のCPUもなかなかやっかいで、**オーバークロック**状態で使われていたものだと、寿命が短くなってしまう恐れがある。したがって、状態の不明な**バルク版**の中古には手を出せない。リテール版（メーカー正規品）の中古で、かつ**CPUクーラーやケースに改造された形跡がない**ものを探すのに、ひと苦労した。結局、マザーボードが対応する最高クロックの600MHzの品は見つからず、550MHzのもので我慢することにした。

しかし、CPU 1個を探すのにこれだけ苦労させられるとは思わなかった。Socket 7のPentiumの時代は、比較的長期にわたってソケット互換のCPUが入手できたように思うのだが、最近のCPUの世代交代の速さは異常ともいえる。**CPUやマザーボードなんて使い捨てさ**と言える人はいいが、零細自営業者の身としてはそうもいかない。1年も経たずに時代遅れになるPCを、**法定償却期間の6年間**目いっぱい使おうとするのはさすがに非現実的かもしれないが、せめて3年間くらいは新品の部品が買えるようにしておいてほしいものだ（筆者注：最近になって法定償却期間は5年に短縮されたようだが、従前に購入したPCには適用されない）。

### リダイレクト再び

さて、前回説明した「バックグラウ




**画面1　バックグラウンドで実行中のコマンドからのメッセージ**
**viやEmacsなどのアプリケーションで作業している最中に、エラーメッセージなどが表示されると、画面が乱れてビックリさせられることになる。編集中のデータに影響はまったくないのだが、あまり気分の良いものではない。**

ンド実行」をいろいろ試してみると、バックグラウンドで実行中のコマンドから出力される文字列が、不意に画面上に表示されて**作業の邪魔**になったり、実行していたはずのコマンドが、気がつくと止まっていたりすることがある。コマンドの実行中のエラー発生や、作業の進行状況をユーザーに伝えるメッセージが表示されるわけなので、**まったく無用**のメッセージというわけではないのだが、たとえば、viなどでファイルを編集している最中にこのようなメッセージが表示されると、画面が乱れてしまって**びっくり**させられることになる（**画面1**）。

viやEmacsの場合、Ctrl-Lを入力すれば、乱れた表示を元に戻すことはできるのだが、表示されたメッセージが重要なものだった場合、その内容を確認する前に再表示を行ってしまうと、もう一度確認するすべはない。

このように、バックグラウンドでコマンドを実行する場合には、コマンドから出力されるメッセージを何らかの形で**交通整理**する必要があるのだ。そこで利用するのが、以前（本誌2000年10月号の当連載）で説明した**リダイレクト機能**である。コマンドからの出力を、ファイルなどにリダイレクトしておけば、別のコマンドの実行中にメッセージが不意に表示されることはなくなるわけだ（**図1**）。

リダイレクトの使い方についてはすでに解説したのだが、ここで改めて、基本的な使い方だけおさらいしておこう。

## 必要なメッセージを保存する

Linuxのコマンドから出力される文字列は、おもに**標準出力**と**標準エラー出力**の2種類に分けられることは説明した。標準出力は、たとえばファイルの内容を加工した結果など、コマンドの処理結果を出力するのに使われる。標準出力を画面ではなくファイルに保存したい場合には、次のように＞というリダイレクト文字を使う。

```
$ grep "シロクマ" zoo.txt > result.txt
```

結果の保存を、既存のファイルの末尾に追加する形で行いたい場合には、＞＞というリダイレクト文字を使う。

```
$ grep "ツキノワグマ" zoo.txt >> result.txt
```

上のようなコマンドラインを、バックグラウンドで実行したい場合には、コマンドラインの末尾に&をつければよい。すると、grepコマンドの実行結果は画面に出力されず、すべてresult.txtというファイルに保存されるので都合がよいわけだ。

```
$ grep "ツキノワグマ" zoo.txt >> result.txt &
[1] 1123
```

バックグラウンドでコマンドを実行すると、上のようにジョブ番号とプロセス番号が表示されるが、これは出力をリダイレクトしたときでも画面に表示される。実行したコマンドが表示しているのではなく、シェルプログラム（bash）自身が表示しているメッセージだからだ。

コマンドの標準エラー出力もリダイレクトすることができる。リダイレクト文字2＞を使うのだ。

```
$ grep "アナグマ" zoo.txt 2> error.txt &
```

このコマンドラインを実行すると、grepコマンドの結果は画面上に表示され、エラーが発生した場合だけ、そのメッセージがerror.txtに表示される。標準出力とエラー出力をそれぞれ別々のファイルに保存したい場合なら、両方のリダイレクト指定を併記すればよい。

```
$ grep "アナグマ" zoo.txt > result.txt 2> error.txt &
```

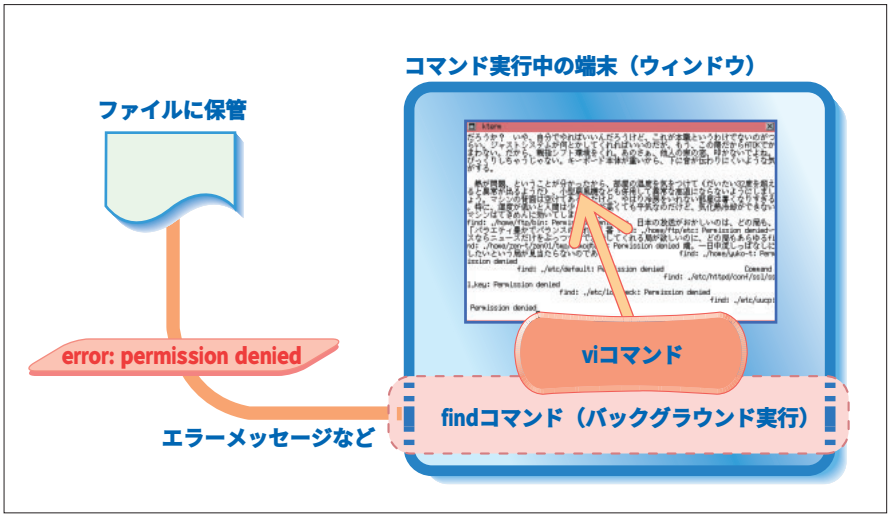



図1　コマンドからのメッセージを「交通整理」する
コマンドからの各種メッセージを、ファイルなどに保存するようにしておけば、画面1のような事態を避けることができる。

リダイレクト文字&>を使えば、標準出力と標準エラー出力を同じファイルに保存することもできる。

```
$ grep "ハナグマ" zoo.txt &> result.txt &
```

ただし、このようにすると、コマンドの結果とエラーメッセージが**ごちゃまぜ**の状態で保存されることになるので、用途によっては不適切なこともある。

いずれにせよ、バックグラウンドでコマンドを実行する場合には、コマンドからの出力にどう処理するかを、つねに考えておかなければならないのだ。これは、コマンドラインから実行する場合だけでなく、今後解説する「スクリプト」を作成したり、コマンドのタイマー実行（cron機能）を使ったりするときにも必須となるので、頭の片隅にとどめておいてほしい。

## 要らないメッセージを無視する

コマンドの中には、実行時にコマンドのバージョンを表示したり、処理の途中結果を表示するものがある。このようなコマンドをバックグラウンドで実行するときには、途中結果などをファイルに保存しておくこともできるが、**わかりきった処理**をするようなときには、そのような結果は必要ない場合もある。

このようなコマンドはたいてい、**メッセージを表示しないようにするオプション**（-qオプションとか--quietオプションなど）が使える場合が多いのだが、まれにそのようなオプションがない場合や、オプションを調べるのが面倒くさい場合もあるだろう。

### ■nullデバイスの使い方

このような場合には、画面上に不要なメッセージが表示され続けるのを我慢しなくても、出力されるメッセージを**闇に葬り去る**ことができる。本連載の第13回でもちょっと触れたことがある**nullデバイス**（「ナルデバイス」あるいは「ヌルデバイス」と読む）という特別なファイルをリダイレクトの出力先に指定すればよいのだ。たとえば、WAV形式のファイルをMP3形式に圧縮する「bladeenc」というコマンド（Linux標準のコマンドではない）を例にとってみよう。このコマンドには「-quiet」というオプションがあって、これを使えば途中経過のメッセージを表示しないようにできるのだが、このオプションを忘れてしまった場合には、次のように起動すればメッセージを表示しないようにできる。

```
$ bladeenc track1.wav track1.mp3 > /dev/null &
```

/dev/null宛に出力されたデータは、**何の処理もされずに**すべて破棄される。このため、画面上には余分なメッセージが表示されることなく、コマンドは**粛々と**処理を続けることができるのだ。/dev/nullは、いわば**ブラックホール**のようなもので、吸い込んだデータは二度と表に出てくることはないのだ（**図2**）。

余談だが、/dev/nullは**読み出し**もできるファイルである。次のようなコマンドを実行してみてほしい。

```
$ cat /dev/null > nullfile.txt
```

すると、カレントディレクトリにnullfile.txtというファイルが作成されるが、その中身は**空っぽ**である。

```
$ file nullfile.txt
```

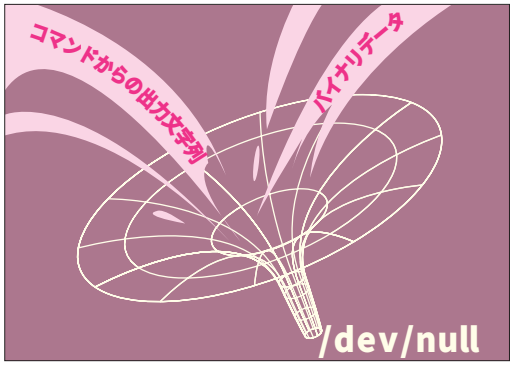
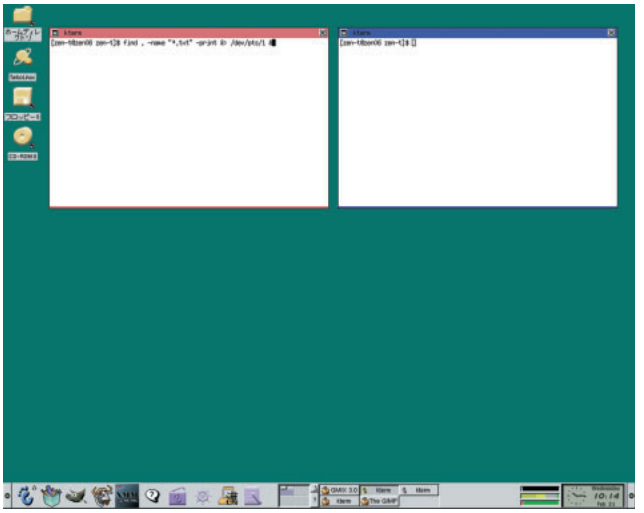



**図2　/dev/null**
特別なファイル/dev/nullは、いわば「ブラックホール」のような存在である。このファイルに書き込まれたデータは、なんの処理も行われずに、破棄されてしまう。



**画面2　X Window Systemの環境で、ktermの画面を2つ開く**
ここでは、左側の画面を最初に、右側の画面を2番目に開いている。別のウィンドウを開いていない状態なら、最初に開いた端末が「0番」、次に開いた端末が「1番」になる。

```
nullfile.txt: empty
```

標準エラー出力にメッセージを出力するようなコマンドについても、エラー出力もいっしょに/dev/nullにリダイレクトすれば、さらに**静かに**なる。

```
$ bladeenc track1.wav track2.mp3
&> /dev/null &
```

エラーが発生しないことがわかりきっているコマンドを実行したり、エラーが起こっても気にしないような場合には、上のようにすべての出力を/dev/nullに放り込んでしまうこともできる。実際、タイマー機能を使ってコマンドを自動実行するような場合には、このように/dev/nullを活用することが多い。

## メッセージを別のウィンドウに出す

コマンドの途中経過やエラーメッセージは必要だが、作業中のコンソール（ウィンドウ）には出してほしくない、というような場合には、**メッセージだけを別のコンソール（ウィンドウ）に出す**こともできる。たとえば、X Window Systemを立ち上げている状態なら、バックグラウンドで実行中のコマンドのエラーだけを別のウィンドウに表示させたり（**図3**）、コンソール上で作業中の場合には別の「仮想コンソール」（Altキー＋ファンクションキーで切り替える画面のこと）にメッセージを出すようにもできる。

たとえば、X上で作業している場合を例にとってみよう。前ページ**画面2**のように、複数のターミナルウィンドウ（ktermやxtermなど）を開いておいて、左側のウィンドウ上でファイルを検索するコマンドをバックグラウンドで実行させる。

```
$ find . -name "*.txt" -print &>
/dev/pts/1 &
```

ここで、> /dev/pts/1というのが、出力先のウィンドウ（仮想端末）を指定している部分で、末尾の「1」という数字が出力先の端末の番号を示していると考えてほしい。各ウィンドウに割り当てられている端末番号を調べる方法については後述する。

ともかく、このコマンドを実行すると、ファイルの検索が進むにつれて、その検索結果は、コマンドを実行したウィンドウではなくて、もう一方のウィンドウに表示されるようになる（次ページ**画面3**）。

Xを起動せずに、コンソール上で作業している場合も同じような方法で、コマンドからのメッセージを別のウィンドウに表示させることができる。

```
$ find . -name "*.txt" -print &>
/dev/tty2 &
```

出力先を指定する書式が少し変わっているが、ここでは仮想コンソールの2番に出力しようとしている。ただし、ディストリビューションによっては、ここで「Permission denied」というエラーが発生することもある。このような場合は、rootユーザーの状態でないと、このテクニックは使えないことになる。

### ■端末番号を調べる方法

コマンドを入力している画面の端末番号を調べるには、**ttyコマンド**を使う。

```
$ tty
/dev/pts/0
```

このように、現在使用中の端末のデバイスファイルが表示されるので、そのコンソールやウィンドウに対して出力したい場合には、ここで表示されたデバイスファイルに対してリダイレクトすればよい。

複数のユーザーがログインしているような環境の場合、どのユーザーがどの仮想端末を使っているかを調べることもできる。**wコマンド**を

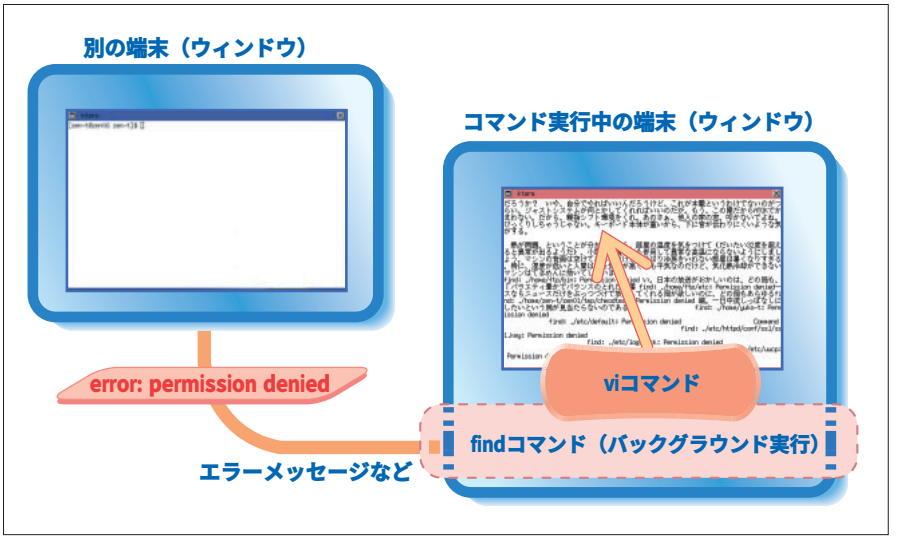



図3　コマンドからのメッセージを別のウィンドウに表示する
コマンドからのメッセージをファイルではなく、別のウィンドウに表示させるように、リダイレクトすることもできる。

使うのだ（**リスト1**）。

ここでは、何人かのユーザーが、X上で複数のウィンドウを開いたり、別のマシンからログインしている状態がわかるが、それぞれのユーザーがどの仮想端末を使っているかもわかるようになっている。ここで、別のユーザーの端末に対して文字列を出力したらどうなるかと考えるかもしれないが、現在のLinuxでは別のユーザーの端末に対しては、書き込みも読み出しもできないようになっている。セキュリティ上の問題が多いからだ。しかし、大昔の、のどかな時代には、友人のコンソール宛にメッセージを直接送りつける**簡易チャット**のようなことも、日常的に行われていたようである。

### ■「仮想端末」について

ここで、**仮想端末**とはどのようなものかについて、ごくおおざっぱに説明しておこう。Linuxでは、X Window Systemを使ったり、仮想コンソール機能を使うのは、1台のマシンに複数の**端末**（ターミナル装置）を接続するのと同じこととみなされるのである（**図4**）。あるいは、LANなどのネットワークを経由してtelnetでログインする場合も同じである。

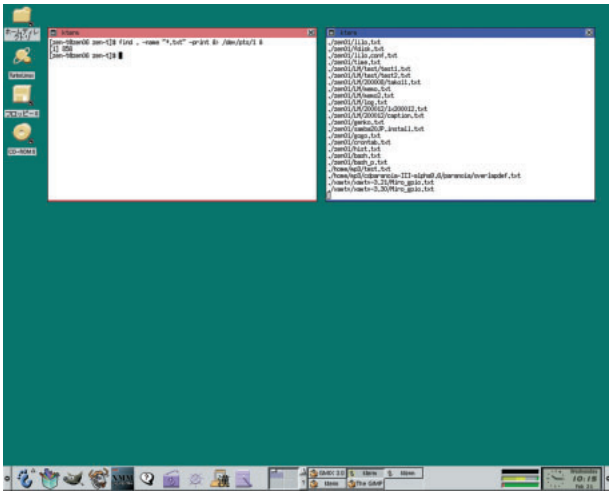
1台のマシンに複数の端末が接続されている場合、ユーザーからの入力（キーボード入力）や画面への出力、すなわちプログラムとユーザーとの間の通信を行う際に、どのユーザー（端末）との間でのやりとりなのかをきちんと整理できないと、複数のコマンドとユーザーの間の入出力がごちゃまぜになってしまう。このため、Linux上で動いている大多数のプログラムは、現在自分がどの端末とやり取りしているのかを常に把握していて、入力や出力をその端末との間でのみ行うようになっているのだ。つまり、**標準入力**や**標準出力**は、リダイレクトで接続先を変えない限り、**プログラムが起動された端末**に対して接続されているのである。

標準入力や標準出力は、リダイレクトという方法で簡単につなぎ変えることができる。このとき、接続先にファイルではなく、仮想端末を表わす特殊なファイル（/dev/pty/?や/dev/tty??など）に指定すれば、プログラムの接続先を別の端末に変えることができるわけだ。先ほどの例では、出力だけをつなぎ変えていたが、入力元を変えることもできる。

ただし、標準入力については、別の端末につなぎ変えても、あまり役に立たないことが多い。ここで、ちょっといたずらをしてみよう。Xを立ち上げた状態で、ktermなどのターミナルウィンドウを3つ開いてみよう（**画面4**）。そのうちのひとつで、viを起動してみる。ただし、このとき次のようなコマンドラインで起動して、入力元を別の仮想端末に指定する。

```
$ vi test.txt < /dev/pts/1 &
```

すると、**画面5**のように、一見なんの問題もなくviが起動するが、実際に使ってみると、キーボードから文字を入力すると画面には表示されるが、まともな編集作業ができないことがわかる。そして、入力元に指定した別のウィンドウでキー入力をしてみると、入力した文字が、viを起動したウィンドウに表示されるようになっているのがわかるだろう（**画面6**）。ただし、この状態でまともにviを操作できるかというと、そうでもなくて、入力された文字はすべてがviのウィンドウに送られるのでなく、タイミングによって送られたり送られなかったりするので、結局両方のウィンドウ上とも、まともに



画面3　コマンドのメッセージを別のウィンドウに送った場合
findコマンドの実行結果やエラーメッセージは、すべて右側のウィンドウに表示されるようになった。ただし、この状態では、結果を逆スクロールさせて見ることはできないので、あまり便利ではないかもしれない（ktermの設定を変えれば逆スクロールも可能になるのだが、ここでは触れない）。

```
$ w
  8:51am  up  1:06,  6 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
zen-t    tty1     -                 7:48am 57:39   4.10s  0.00s  sh /usr/X11R6/b
zen-t    tty3     -                 8:19am 30:31   0.06s  0.03s  -bash
zen-t    pts/0    -                 7:54am 58.00s  0.13s  0.12s  bash
zen-t    pts/1    -                 8:16am 12:36   0.07s  0.03s  vi
zen-t    pts/2    -                 8:37am  7:43   0.04s  0.04s  bash
yuko-t   pts/3    zen05             8:51am  0.00s  0.08s  0.02s  w
```

リスト1　wコマンドで仮想端末の番号を得る

作業できなくなってしまう。

この状態を元に戻すには、最初に起動したviを強制終了させるしかない。3番目のウィンドウに移り、psコマンドでviのプロセス番号を調べてkillする。

```
$ ps x | grep vi
589 pts/1    S    0:00 vi

$ kill -KILL 589
```

うまくプロセスを強制終了できれば、すべてのウィンドウは元通りの状態に戻る。

このように、一般のユーザーにとって、仮想端末に対するリダイレクトが必要になる場面はそれほど多くはないのだが、Linuxの内部ではこのような処理はきわめて頻繁に行われていて、とくにX Window Systemの環境においては、この機能なしでは、アプリケーションをまともに動かすことすらできない。自分で使うことは少ないかもしれないが、このような機構があって、縁の下で働いているのだ、ということに**思いを馳せる**のもよいだろう。

●

さて、次回は、いままで説明してきたバックグラウンド実行とリダイレクトの知識を生かして、「シェルスクリプト」に挑戦してみたいと思う。

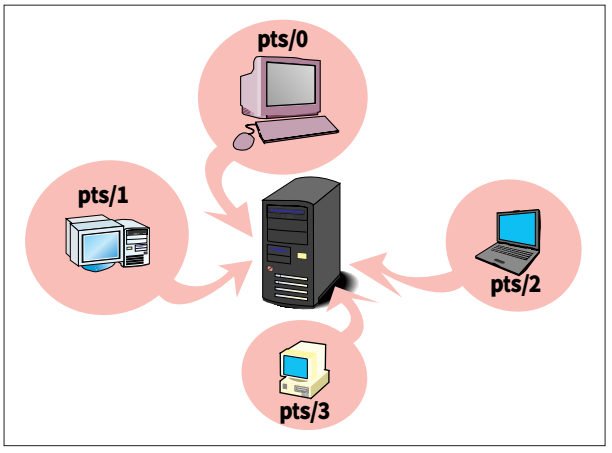
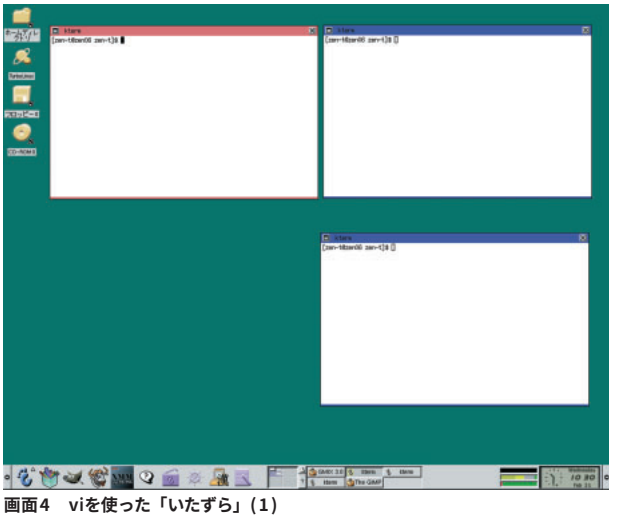
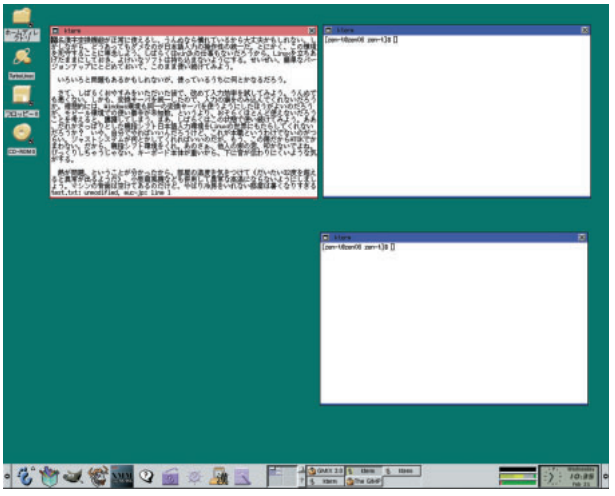


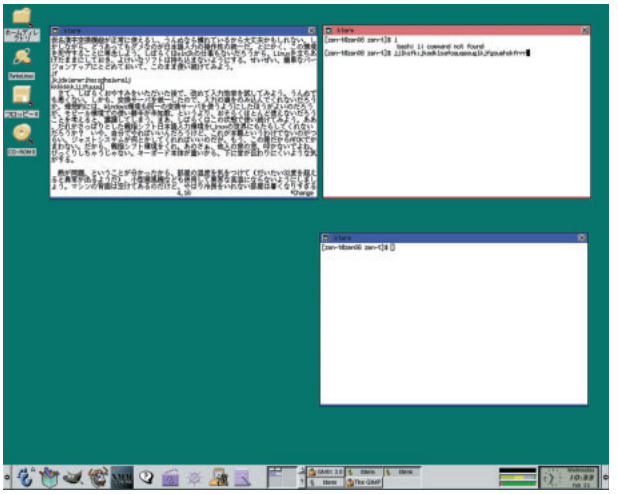

図4　複数のウィンドウ（仮想端末）を開く
X Window Systemで複数のktermウィンドウを開いたり、コンソール上で仮想コンソールを利用すると、1台のマシンに複数の「端末装置」からログインしているのと同じこととみなされる。それぞれのウィンドウごとに1台の「仮想端末」が割り当てられるのだ。



画面4　viを使った「いたずら」(1)
ktermの画面を3つ開く。上列左上から順に/dev/pts/0、/dev/pts/1、/dev/pts/2の仮想端末がそれぞれ割り当てられているものとする。



画面5　viを使った「いたずら」(2)
標準入力を右の仮想端末に指定してviを起動すると、きちんとファイルを読み込んでviの編集画面が表示される。一見なんの問題もなさそうなのだが……。



画面6　viを使った「いたずら」(3)
しかし、左の画面では一切の編集作業ができない。キー入力は画面に表示されるのだが、それだけで、ファイルの編集は一切できないし、viのコマンドも受け付けてくれない。この状態から抜け出すには、右下のウィンドウからkillコマンドでviを「殺す」。場合によっては、第3のウィンドウを使わなくても、viを表示中のウィンドウでCtrl-Zキーを押してviを一時停止して、killコマンドを使うこともできるが、うまくいかない場合もあるようだ。

Linux ガベコレ

# Linux Garbage Collection

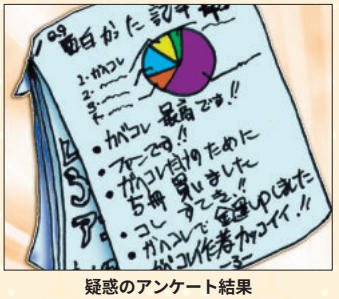
## 目からウロコの用語辞典

文: しのはらひろあき
*Text : Hiroaki Shinohara*

**第12回**
**【4色】（よんしょく）**
**【家庭内LAN】（かてい・ないらん）**
**【Ethernet】（いーさねっと）**
**【ホームサーバ】（ほーむさーば）**
**【Samba】（さんば）**
**【Netatalk】（ねたとーく）**

## 4色
**【よんしょく】**

「今月はこのページ、4色になりますから」「ええっ！　ふだんは1色でおまけに巻末掲載と不遇なのに。おかげで連載開始1年にして存在自体知らない読者が84%（編集部調べ）だというのに……。なにかあったんですか？」「私もこんなページに4色というのは資源の無駄遣いだと思うんですが、上のほうの指示ですから。なんでもインターネット経由のアンケートで“編者が不憫でならない”“打ち切りになるまえに一度だけでも”“原稿料をアップしてあげてください。とっても苦しいんです”という要望が大量に届いたそうで。ただ、なにかあやしいんですよねえ。ログを調べたら回答の多くが海外のプロキシサーバ経由だったり、コメントの文体が似たり寄ったりだったり、1時間で500通も同様の回答が集まったり。心当たりありませんか？」「（ぎく）あ、ありませんよ。世の中にはやさしい読者さんがいらっしゃるということじゃないですか？　ありがたや、ありがたや……」「まあ、いいでしょう。ただ、4色だといっても何をやるかが問題ですねえ」「たとえば、こんなのはどうですか？　縁の下の力持ちで、ふだん陽の当たらない担当さんの紹介とか」「はあ？」「色が使えることを活かして“ライターが締切を守らないことに怒り顔を赤くする担当さん”とか、“あと2時間でページが白いままの雑誌が店頭に並ぶことになるのを恐れて青ざめる担当さん”とか、そういう素顔をありのままにお届けするわけですよ。はっはっは」「……誰のせいで赤くなったり青くなったりしてると思ってるんだ」「……失礼しました。でも、4色っていっても、使える色が3つ増えるだけなんですよね。どうせなら上の方も16色とか、256色とか、65536色とかにしてくれれば、ほかの展開を考えられたのに。このページのタイトルロゴをレインボーカラーに染めてみたり。まあ、1670万色とまではいいませんが」「……もしかしてあなた、ライターとして基本的な知識が欠けてませんか？」「はっ？　どどどどうゆうことですか。ぼくだって、いっぱしのライターです！」。



疑惑のアンケート結果

**※とってもLinuxに関係あるひとくちメモ：「4色ページ」とはカラーページのことを指す。**

## 家庭内LAN
**【かてい・ないらん】**

挙国一致体制でのIT革命が進行するなか、見過ごされがちなのが家庭内の情勢である。バブル崩壊以来のなべ底不況を、ボトムアップの消費拡大により解決すべく奮闘する男性。「このPentium 4購入が日経平均を押し上げるのだ」「30Gバイトプラッタのハードディスクを買えば円高基調につながる」と信じ、アキハバラに日参する彼らを理解しようとしない妻、彼女は多


武装

い。崇高な経済行為を単なる物欲としか受け取れない女たちは、家庭内で男性陣と激突を繰り返している。これが“家庭内乱”で、IT化の潮流にのって乱をLANと書き換えるとなにかナウい感じがする。特に家庭内でのネットワーク敷設について、「最近、夜の“つながり”はご無沙汰なのにパソコンなんかつなげてうつつをぬかしやがって」と女性の反感を買いやすいのは、言い得て妙であろう。

編者も私財をなげうって自宅に無線LANを導入、内需拡大に貢献したが、愚妻は「そんな金があるなら指輪がほしい」「3ドアの冷蔵庫に買い換えろ」と理解がなく、すでに一触即発の状態に陥っている。あげく、先月届いたクレジットカードの利用明細を見たら、いつのまにかダイヤの指輪が購入されていた。問いただしてみると「10%オフで安かったから」「リボ払いにしておいたから」、あげくのはてに開き直って「ダイヤの結婚指輪、あなたが買ってくれないから自分で買ったの」と来る。結婚前には「指輪なんて、カタチやモノはどうでもいいの。いっしょにいられれば」といっていた人物と同一とは思えないセリフである。たしかに外見は体重2割増になって変わったが、中身まで変わっていたらしい。おまけに最近では、幼い娘まで「LANカード買うお金があるならどうして“とっとこハム太郎”のゲーム買ってくれないの？」などといい出す始末。ちなみにハム太郎とは、両面をフライパンで軽く焼くとおいしそうな名前だと思う。

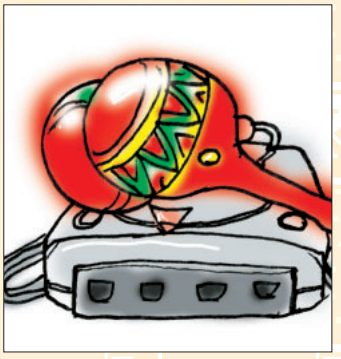
## Ethernet
**【いーさねっと】**

家庭内LANでもよく利用される、ネットワークの規格。凡百の解説書には「かつて宇宙を満たしている媒質と信じられた“エーテル”にかけて命名された」などとまことしやかに語られているが、正しくは思いつきで設計されたための信頼性の低さゆえ、パケットを2つ流すとA or Bどちらかのパケットしか相手に到着しない＝either A or B netから来た名。もちろん、こういう無責任なネットワークを利用する場合には、通信に支障が出た際にも「まぁ、いーさ」で済ます度量の大きさが必要である。

## ホームサーバ
**【ほーむさーば】**

家庭内LANの普及、そしてブロードバンド時代を迎え注目されているのが「一家に1台ホームサーバ」である。成長が見込まれる、この未来の巨大市場には各社とも続々と新製品を送り出し続けている。第1次ブームとなったのが、猛暑商戦の追い風を受けての2000年夏。業界トップ企業が投入したホームサーバは消費者の圧倒的支持を受け、社会現象にまでなった。このころのシステムは消耗品としてガスボンベが必要だったため継続利用に難点があったが、続く第2次ブームの2001年新春には、圧縮ポンプ使用によるボンベレス化が実現。より一層の普及に拍車をかけた。なお、ホームサーバの使いすぎは家庭内LAN、もとい「家庭内酒乱」の原因となるので、乱用に注意されたし。

## Samba
**【さんば】**

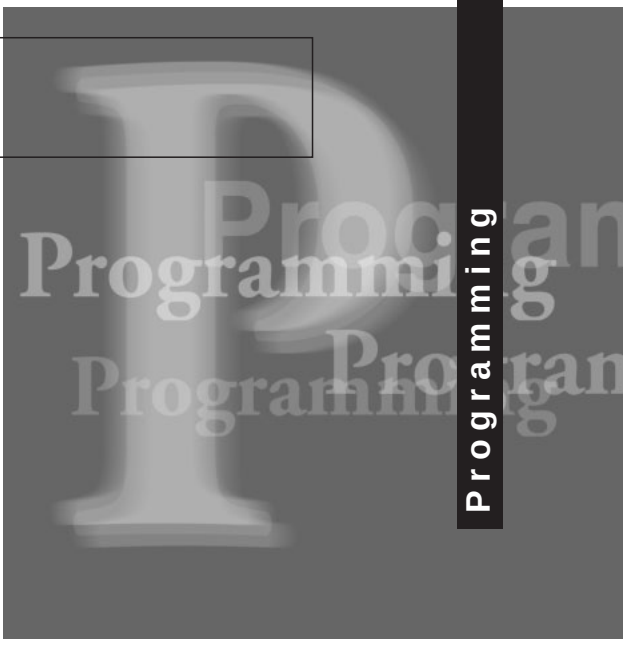

やっぱりセットで一家に一台でしょう

編者もオススメ。これからの家庭内LAN時代、一家にひとつ常備しておきたいアプリケーションである。なにも発売元が弊誌発行会社のグループ企業だからと手心を加えるわけではない。ひとたびマラカスを手に持ち画面にしたがって踊り出せば、ふしぎな爽快感とともに癒しが得られることを保証しよう。我が家では愚妻のダイエットに、締切からの現実逃避にと、さまざまな分野で活躍している。しかし、通販でマラカスコントローラを注文した翌日にDreamcast生産中止の報が流れたときはさしもの筆者も驚きを禁じ得なかった。DC退役後はNetBSDをインストール、Windowsネットワーク向けファイルサーバとして余生を過ごさせようと考えているところである。はて、この雑誌はBSD Magazineだったか……。なお、Samba動作マシンにDDR SDRAMを組み込むと大手ゲームソフトメーカーとの音ゲー特許論争に火がつく可能性があるため、初心者は注意が必要である。DDRはJava対応iモードなどで楽しんでいただきたい。

## Netatalk
**【ねたとーく】**

Macを導入している家庭でのLAN構築に不可欠なサーバアプリケーション。あの藤田 田もハンバーグラーも、もちろん愛用していると思う。ありがちなネタで読者に殴られそうだが、あと1行で終わるので今月もなんとか逃げ切った。

# プログラミング工房




**先月号では、文字列処理プログラミングの例題として、等距離暗号文字列検索プログラムを作成し、コマンドラインオプション解析部分について解説した。今月号では、プログラムの中核部分である検索部分について解説してみる。**

## 第16回　文字列処理プログラム（2）

文：藤沢敏喜
*Text : Toshiki Fujisawa*

### 文字列データの取り扱い

先月述べたように、今回対象とするデータは、ヘブライ語で書かれた旧約聖書の先頭から304805文字分である。このデータは前回解説した方法により、bible.txtという304805バイトのファイルとして作成することが可能である。検索処理を行うためには、まず最初にこのファイルのデータをメモリ中に読み込む必要がある。そのためには、たとえば、

```
char *buff;
FILE *fp;
buff = malloc(304805);
fp = fopen("bible.txt", "r");
fread(buff, sizeof(char), 304805, fp);
```

というような、プログラムを作成すればよいことになる（実際にはメモリ確保が失敗したときの処理や、ファイルが開けなかったときの処理などが必要である）。

この方法は、複数のファイルを切り替えたい場合には都合がよいが、今回のように取り扱うべき対象データが固定化している場合には毎回読み込むのが面倒である。また、プログラムと別にデータファイルを用意する必要があるため、インストール時にはデータファイルを適当なディレクトリに置き、そのディレクトリの位置をプログラムに指定する必要がある点もまたやっかいである。

この問題を解消するためには、実行ファイル自体にデータを埋め込むという方法がある。C言語では、変数の初期値を指定することが可能なので、たとえば4バイトからなる文字配列の初期データを指定するには、次のように記述することができる。

```
char text[4] = { 65, 66, 67, 0 };
```

ここで、65、66、67はそれぞれ、ASCIIコードでA、B、Cに対応するので、printf（text）;を実行すると、「ABC」という文字列が表示されることになる。

さて、上記のように、初期値が指定された変数を含んだソースコードを作成するにはどうしたらよいだろうか？bible.txtは300Kバイトもあるので、人間が手で書き上げるのはもちろん不可能である。したがって、なんらかのプログラムで変換することになるが、今回はこの変換をするのに、**リスト1**に示すmk-dateというperlスクリプトを用いている。このスクリプトでは最初に、

```
unsigned char bible_data[304805*2] = {
```

という文字列をプリントし、それから標準入力から1バイトずつ、文字変数$cに読み込んだバイナリデータを、

unpack関数を用いて数値に変換している。

そして、それを10進数の文字列としてカンマで区切りながらプリントし、1行が長くなりすぎないように、16個ごとに改行を出力している。このスクリプトを、

```
$ perl mk-data < bible.txt > bible_data.c
```

として用いることにより、初期値データを持つ変数を含んだ、bible_data.cというプログラムソースコードができあがる。あとはこのファイルをコンパイルして、プログラムとリンクすることにより、データを実行ファイルの中に埋め込むことができる。

## 等距離文字列暗号

データが用意できたところで、次は今回の例題プログラミングにおいて主題となる検索についての話である。詳細なプログラムソースコードを解説する前に、どのようなパターンの文字列を検索するかをまず最初に解説してみたい。

xbibleで検索する文字列は「等距離文字暗号」と呼ばれるもので、対象の文書を特定の文字数で折り返して（改行して）、2次元空間に並べ、単語を構成する文字を等距離で配置する暗号方式である。

ここで、等距離文字暗号の具体的な例を示すために、次のような35文字からなる1次元文字列を考えてみる。

```
ALL AI STUDY IS NULL. CANSEL SEND USA BY UNIX.
```

この文字列からすべてのスペースを取り除くと、

```
ALLAISTUDYISNULLCANSELSENDUSABYUNIX
```

となる。この1次元文字列を7文字ごとに改行し、2次元空間に表示すると、次のように、「L、I、N、U、X」、「A、S、C、I、I」という2つの文字列が浮かび上がる。

```
 A  L (L) A [I] S  T
 U  D  Y {I} S  N  U
 L  L [C] A (N) S  E
 L [S] E  N  D (U) S
[A] B  Y  U  N  I (X)
```

1次元の文字列上では、L、I、N、U、Xの文字列は、

```
ALLAISTUDYISNULLCANSELSENDUSABYUNIX
--L-------I-------N-------U-------X
  →       →       →       →       →
  1234567812345678123456781234567 8
```

▼リスト1　ソースコードを生成するperlスクリプト

```
#!/usr/bin/perl

printf("unsigned char bible_data[304805*2] = {\n");

$sep = "";
$i = 0;
for(;;){
    if( ($c = getc) eq "" ){
        last;
    }
    printf($sep);
    printf("%3d", unpack("C", $c )) ;
    if( ( $i++ % 16 ) == 15 ){
        $i = 0;
        printf(",\n");
        $sep = "";
    }else{
        $sep = ",";
    }
}
printf(",0\n");
printf("};\n");
printf("long sizeof_bible_data = sizeof(bible_data);\n");
```

というように、距離がプラス8文字である等距離文字列として現れる。また、A、S、C、I、Iは、

```
ALLAISTUDYISNULLCANSELSENDUSABYUNIX
----I-----I-----C-----S-----A------
   ←     ←     ←     ←     ←
     654321654321654321654321
```

というように、距離がマイナス6文字である等距離文字列として出現する。

ちなみに、上記の「ALL AI STUDY IS NULL CANSEL SEND USA BY UNIX」という文字列は筆者が作成したものである。この文を作るためには、まず最初に2次元の配置を考えながら、次のような1次元文字列を作った。

```
＋＋L＋I＋＋＋＋＋I＋＋＋＋＋C＋N＋＋＋S＋＋＋U＋A＋＋＋＋X
```

そして、「＋」の場所に意味がありそうな文になるように文字を埋めていったのである。しかしながら、文を意味あるものにするのは難しいため、文法や内容が滅茶苦茶な文章しか思い付くことができなかった。

もし聖書の暗号というものが存在し、誰かがそれを作成したと仮定すると、聖書の本文を意味のある文章にしながら、数千文字間隔での暗号を多数埋め込むのはたいへん困難な作業だったと推察される。

しかし、同じ意味で長さが違う単語や文の巨大データベースを用意し、コンピュータですべての場合の組み合わせを力まかせに演算することができれば、特定の単語を暗号として自動的に埋め込むことは、理論的には可能であると思う。並列計算可能なコンピュータの進歩を考えると、10年後にはそんなに難しいことではなくなるかもしれない。

ちなみに10年前には、リニアなメモリ空間として、6万文字分しか確保できない非力なOSが主流で、CPUの演算速度も今からは信じられないくらい遅かった。したがって、今回の例題プログラムのように、30万文字もある文書から実用的な速度で検索するプログラムを書くことは、きわめて困難だったのである。

## 文字列の一致

さて、上記のような等距離文字列を検索するためには、単語の一致を調べる関数が必要になるが、最初はもっと簡単な、文字間距離が1である文字列比較関数から考えてみよう。

### ■距離が1である文字列の比較

説明の都合上、ここではbuffという名称の文字列へのポインタで示されるメモリ領域に、「ABCDEF」という文字列があるものとしよう。これはたとえば、



Column

### ヘブライ語の歴史

この記事を書くために、ヘブライ語と日本語の相互変換辞書や文法書がほしくなったが、近所の書店では手に入れることができなかった。そのため、東京で最大の本屋街へ行って7冊ほど仕入れてきたのだが、ヘブライ語というのは、日本ではかなりマイナーな存在であるようだ。

『ヘブライ語のすすめ』（ミルトス発行、ISBN4-89586-019-1）によると、ヘブライ語の歴史は3000年を超えるものの、紀元前2世紀ごろから日常的な話し言葉としては使われなくなったそうである。

そして、紀元前73年のマサダの陥落から、ユダヤ人は祖国とともに言語を失い、19世紀の終わりごろまではヘブライ語を母国語として使う人はほとんど皆無であったという。そのため、世界中に散らばったユダヤ人は、それぞれの国でさまざまな言語を話すようになった。

しかし、20世紀初めになってパレスチナに集結するようになると、集まったユダヤ人の間で共通に話せる言語が存在しないことが問題となる。そこで、死語となっていたヘブライ語を公用語として用いるという、たいへん困難な作業が進められることになった。このようにして、現在のイスラエルでは、生まれたときからヘブライ語を母国語とするようになった人が多数派となっている。

現代の辞書によって、古代の単語を調べることができる言語はたいへん珍しいが、これは2000年ものあいだ冷凍保存されていた言語が解凍されたためであるといえよう。

「日本国」に「日本人」が住み、「日本語」を使用しているのに対して、「イスラエル」には「ユダヤ人」が住み、「ヘブライ語」を話している。国名と民族名、そして言語名がすべて違うところに、イスラエルの複雑な歴史がうかがえる。

なお、ヘブライ語やイスラエルに関する書籍の多くは、ミルトスから発行されていて、Webページでのオンライン販売も行われている（http://www.myrtos.co.jp/）。また、キリスト教徒が多い国では、ヘブライ語入門用Webページなどの情報もいろいろあるので、検索エンジンで探してみるとよいだろう。

```
buff[0] = 'A';
buff[1] = 'B';
buff[2] = 'C';
buff[3] = 'D';
buff[4] = 'E';
buff[5] = 'F';
```

というように初期化することにより生成される。そして、次のように文字列の終わりがゼロで終端された「ABC」という文字列のアドレスが、wordというポインタ変数に入っているものとする。

```
word[0] = 'A';
word[1] = 'B';
word[2] = 'C';
word[3] = '\0';
```

このような状況では、buffというメモリ領域にwordで示される単語が存在するかどうかを調べる関数は、次のように書くことができる。

```
typedef enum { false=0, true=1 } bool_t;

bool_t
```

## Column

### X Window Systemでのヘブライ語の表示

ヘブライ語は22文字の子音と1文字の母音からなるが、文末などで形が変化することもあり、フォントとしては27個ある。

多くの読者のマシンには、X Window Systemのフォントとして、ヘブライ文字がすでにインストールされているのではないかと思う。ヘブライフォントが存在するかどうかを調べるには、

```
$ xlsfont | grep heb
```

とすればよい。ヘブライフォントがあれば、

```
heb6x13
heb8x13
```

というように表示されるはずである。このフォント名を使い、

```
$ xfd -fn heb8x13
```

というコマンドを実行すると、フォントの一覧を表示することが可能である（**画面1**）。また、

```
$ xterm -fn heb8x13
```

を実行すれば、ヘブライフォントが使えるxtermを起動することができる。このxterm内で、CD-ROMに収録してある、asc2heb.shという変換スクリプトを使うと、暗号表をヘブライ文字で表示することが可能である。たとえば、

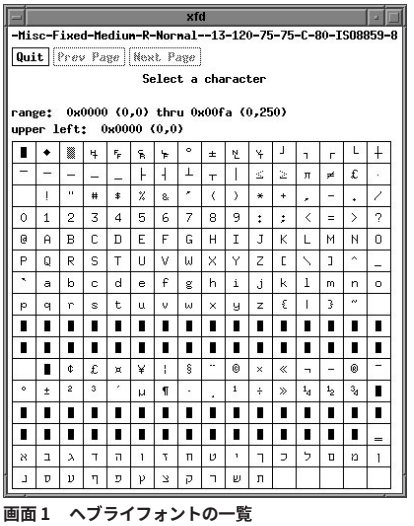
```
$ xbible 引数と検索文字列 | sh asc2heb.sh
```

というように使用し、xbibleの出力をパイプで受けるようにすると、**画面2**のように暗号表をヘブライフォントで表示することができる。

なお、先月号ではオリジナルの聖書データにおいて、「+」および「$」とエンコードされているテットとサメフと呼ばれる文字を、「t」および「s」として変換した。しかし、このマッピング方法では文字列検索時に問題があったため、今月号のCD-ROMに収録してあるxbible-1.12では、大文字の「T」と「S」を割り当てることにした。また、先月号ではマップし忘れていた「#」も、「S」に割り当てるように修正してある。つまり、xbible-1.12でのヘブライ文字とASCII文字とのマッピング関係は、CD-ROMにあるconv-table.shで生成される**画面3**のようになる。

最初のころ、ミミズのようなヘブライ文字は、非常に覚えにくいように見えた。しかし、いろいろな単語を検索しようとすると、辞書を何回も引かなければならないため、この対応関係はすぐに頭の中に入ってしまった。

ちなみに、ヘブライ語から日本語への辞書としては、キリスト聖書塾発行の『現代ヘブライ語辞典』（ISBN4-89606-103-9）が便利であった。




画面1　ヘブライフォントの一覧


画面2　暗号表のヘブライ文字表示


画面3　ASCII文字とヘブライ文字の対応

```
compare(char *buff, char *word)
{
    for(;;){
        if( *word == '\0' ){
            return true;  /* OK */
        }
        if( *buff != *word ){
            return false; /* NG */
        }
        buff++;
        word++;
    }
}
```

このforループでは、1回のループでbuff＋＋、word＋＋が行われるので、1文字ずつ比較が行われることになる。

ここでは、ループ中のif（＊buff != ＊word）という文が真の場合は、文字列が一致しなかったことになるので、return falseを実行する。

そして、ループしているうちに＊word == '\0'つまり、現在着目しているwordの文字が文字終端を意味するヌル文字まで行き着いた場合は、wordの内容が見つかったことになるのでreturn trueを実行する。

### ■距離がsである等距離文字列の比較

さて、先ほど説明したように、等距離文字暗号ではbuffにある文字列は、特定の文字数ごとに離れて（スキップして）並んでいる。

上記で説明したbuffメモリに保存してある文字列が、s文字間隔の等距離文字暗号であった場合、その検索を行うにはどうしたらよいだろうか？

答えは簡単で、buff＋＋としてbuffのアドレスを1文字増加させている部分をbuff ＋= sとして、s文字分増加させればよい。このようにすれば、スキップ数sを指定できる、compare（buff, s, word）という関数を簡単に作成することができる。

またこの関数を使えば、聖書の最初から最後（304805文字）まで、すべてのスキップ文字数において、wordで示される文字列が発見されるかどうかを調べることは非常

## Column

### ヘブライ文字による カタカナ表記

たとえば、「LINUX」という単語は、カタカナで「リナックス」と書いたり、「リヌクス」と書いたりされる。このように、ある言語を別の言語で表記する場合には、いくつかの表現が存在することがある。

したがって、英語や日本語などの固有名詞をヘブライ文字で表記する方法も複数存在する。また、ヘブライ文字は22文字しかなく、母音も1つだけであるため、ヘブライ文字への変換にはいろいろと難しい面がある。

日本語のカタカナをヘブライ文字に変換する方法はいくつか存在すると思われるが、『日本語ヘブライ語小辞典』（ミルトス発行、ISBN4-89586-010-8）では、カタカナからヘブライ文字への変換が定義されている。この定義を、今回のプログラムで使用したヘブライ文字からASCII文字へのマッピングを用いて表記したものが、右の表である。

なお、ヘブライ語は右から左へと表記するが、右の表では左から右へ表記してある。また、今回のマッピングの場合、TとSには大文字と小文字の区別があることにも注意してほしい。

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ア | a | イ | ay | ウ | av | エ | a | オ | av |
| カ | q | キ | qy | ク | qv | ケ | q | コ | qv |
| サ | S | シ | sy | ス | Sv | セ | S | ソ | Sv |
| タ | T | チ | cy | ツ | cv | テ | T | ト | Tv |
| ナ | n | ニ | ny | ヌ | nv | ネ | n | ノ | nv |
| ハ | h | ヒ | hy | フ | pv | ヘ | h | ホ | hv |
| マ | m | ミ | my | ム | mv | メ | m | モ | mv |
| ヤ | y | | | ユ | yv | | | ヨ | yv |
| ラ | r | リ | ry | ル | rv | レ | r | ロ | rv |
| ワ | vv | | | | | | | | |
| ン | n | | | | | | | | |
| ガ | g | ギ | gy | グ | gv | ゲ | g | ゴ | gv |
| ザ | z | ジ | gy | ズ | zv | ゼ | z | ゾ | zv |
| ダ | d | ジ | dy | | | デ | d | ド | dv |
| バ | b | ビ | by | ブ | bv | ベ | b | ボ | bv |
| パ | p | ピ | py | プ | pv | ペ | p | ポ | pv |

| | | | | | |
|---|---|---|---|---|---|
| キャ | qy | キュ | qyv | キョ | qyv |
| シャ | s | シュ | sv | ショ | sv |
| チャ | c | チュ | cv | チョ | cv |
| ニャ | ny | ニュ | nyv | ニョ | nyv |
| ヒャ | hy | ヒュ | hyv | ヒョ | hyv |
| ミャ | my | ミュ | myv | ミョ | myv |
| リャ | ry | リュ | ryv | リョ | ryv |
| ギャ | gy | ギュ | gyv | ギョ | gyv |
| ジャ | gy | ジュ | gv | ジョ | gyv |
| ビャ | by | ビュ | byv | ビョ | byv |
| ピャ | py | ピュ | pyv | ピョ | pyv |

表1　ヘブライ文字とASCII文字のマッピング

にやさしい。

つまり、聖書の全テキストの先頭アドレスがtopで、聖書の長さがLENであるとすると、次のように全文書をしらみつぶしに検索するプログラムを書くことができる。

```
#define   LEN 304805
char  top[LEN] = { 聖書の全文; }
char *buff;
int   s;

for(s=1; s<LEN; s++){
    for(buff=top; buff<top+LEN; buff++){
        if( compare(buff, s, word) ){
            printf("found !\n");
        }
    }
}
```

なお、等距離文字暗号においては、最小スキップの単語を見つけることが重要であるため、スキップ数を変化させるループを外側に配置してある。

## 問題点の解決

基本的なアルゴリズムとしては、上で述べた方法が使えるが、このプログラムにはいくつかの問題が存在する。

まず最初の問題は、topで確保したメモリ領域を超えるメモリアクセスが発生することである。topのメモリ領域は304805バイトであるので、304800文字目から、スキップ数6で文字を比較しようとすると、top[304806]をアクセスしてしまう。この領域にどんな文字が格納されているかはまったくわからないし、OSによってメモリ保護違反となってプログラムが異常終了してしまうこともある。

これを解決するためには、wordの文字数にスキップ数sを乗じたものを文書の最後（top＋LEN）から引いた場所までに検索を制限して、compareを呼ぶようにしておけばよい。つまり、

```
for(s=1; s<LEN; s++){
    limit = top+LEN - s*(strlen(word)-1);
    for(buff=top; buff<limit; buff++){
        if( compare(buff, word, s) ){
            printf("found !\n");
        }
    }
}
```

というように変更するとよいだろう。

ちなみに、for（s=1; s<LEN; s＋＋）の部分、つまりスキップ数を増やしていくループも、同じように制限して効率化することができる。詳しくはCD-ROMに収録されたソースコードを参照してみてほしい。

## compare関数の高速化

プログラムを高速化するためには、一番多く実行される部分の最適化を考えることがポイントである。もちろん、このプログラムにおいて重要なのは、compare関数の高速化である。この関数では、

```
for(;;){
   if( *word == '\0'  )  return true;  /* OK */
   if( *buff != *word )  return false; /* NG */
   buff +=s;
   word++;
}
```

というような処理がなされる。しかしながら、ちょっと工夫をすることにより、次のように2回の比較を1回で済ませるように書くことができる。

```
for(;;){
    if( *buff != *word ){
        return ( *word == '\0' ) ? true : false;
    }
    buff +=s;
    word++;
}
```

ここで、buffに存在するすべての文字がゼロを含まないと仮定しよう。すると、buffのメモリ内容にwordに示されるのと同じ文字列が存在した場合、wordを増加させると必ず終端のゼロに行き着いてから、trueでリターンすることになる。

一方、存在しない場合はwordが示す文字がゼロにならないので、falseでリターンすることになる。

ただし、このプログラムではbuffメモリの領域を超えてアクセスすることがあるため、2倍の量のbuffメモリ領域を確保し、その後ろの半分をたとえば0xFFなどの文字で埋める必要がある。このようなアルゴリズムを採用すると必要となるメモリは増えるが、現在のPC環境では問題にならない量である。

なお、xbibleでは、このようなアルゴリズムによる高速化のほかにも、「特別講座　ステップアップC言語」で解説するinline宣言やregister宣言を使うことによるスピードアップも行っている。

## マイナスのスキップ数

以上で、基本的な検索アルゴリズムの解説は終わったが、もう1つ考えなければならないことがある。先ほどの等距離暗号の解説では、「A、S、C、I、I」という文字列は、距離がマイナス6文字であった。この距離がマイナスとなる等距離文字列も、うまく検索しなければならないのである。

このためには、先ほど定義したcompare関数を改造するのが簡単である。つまり、この関数内で、距離がマイナスとなる文字も同時に検索してしまうようにするのである。ただし、上記で行った高速化を行うためには、wordで示される文字列の先頭より、1文字前にヌル文字（ゼロ）が必要になる。したがって、compare関数に渡すwordは、単なる文字列でなく、次のような構造体にするとよい。

```
typedef struct {
    char null[1];
    char str[MAX_WORD+1];
} word_t;
```

上記のメンバであるnull[0]には常にゼロを入れておき、word->strに検索すべき文字列（ゼロ終端）を入れておく。そして、compare関数には、この構造体を渡すようにすると都合がよい。

このようにして、マイナススキップも検索できるようにしたcompare関数は、結局**リスト2**に示すようなプログラムとなる。

なお、先月のCD-ROMに収録したxbible-1.00では、上記のような高速化は行っていないが、今月号のCD-ROMに収録したxbible-1.12では、今月説明したような検索アルゴリズムを採用している。

また、xbible-1.12では、コマンドラインに指定したすべての単語が見つかった場合のみ表示するなど、外部仕様が大きく変更され、いくつかのバグも修正してある。

## 次号について

今月は等距離文字暗号を題材として、文字列検索プログラミングについて解説した。等距離文字暗号を検索するプログラミングは特殊ではあるが、文字を扱うプログラムを学習するためには、好適な題材ではないかと思う。

現在の検索アルゴリズムはまだかなり遅く、検索には相当な時間がかかってしまう。compare関数のループをやめたり、あらかじめ検索しておいた文字列データベースを参照して高速化するなど、改良の余地はたくさんあるので、ぜひいろいろな実験をしてみてほしい。

なお、来月号では、検索した文字列をディスプレイ上に表示を行う部分について解説する予定である。

**▼リスト2　compare関数**

```
inline bool_t
compare( char *buf, register int skip, word_t *word )
{
    register char   *p, *w;
    int             len = strlen(word->str);

    /* プラススキップ文字列の検索 */
    for( p=buf, w=word->str; *p == *w  ; p+=skip, w++ );
    if( *w == '\0' ){ return true; }

    /* マイナススキップ文字列の検索 */
    for( p=buf, w=word->nul + len; *p == *w ; p+=skip, w-- );
    if( *w == '\0' ){ return true; }

    return false;
}
```

特別講座

# ステップアップC言語

## register宣言とinline宣言

### ■register変数

多くのCPUには、レジスタと呼ばれる記憶装置が存在する。レジスタはCPUの内部にあるため、メモリよりも高速にアクセスできることが特徴である。このため、頻繁に用いるデータを、メモリではなく、レジスタに割り当てれば、非常に高速に動作させることができる。たとえば、

```
int func(void)
{
    int i;

    for(i=1234; i<=5678; i++){

    }
}
```

というプログラムをprog.cとして保存し、

```
$ cc -S prog1.c
```

というコマンドを実行すると、prog.sというファイルができあがる。そして、x86系のコンパイラの場合、次のようなアセンブラコードが出力される。

```
        movl $1234,-4(%ebp)
.L2:    cmpl $5678,-4(%ebp)
        jle .L5
        jmp .L3
.L5:    incl -4(%ebp)
        jmp .L2
.L3:    ret
```

上記では－4（%ebp）、つまりEBPレジスタの内容から4を引いた番地のメモリに、int iの値を保存していることがわかる。

つまり、movel（move long）で1234をiに保存し、cmpl（compare long）で5678とiを比較して、jle（jump less equal）でループ脱出の判断をしているのである。

これに対して、

```
int func(void)
{
    register int i;

    for(i=1234; i<=5678; i++){

    }
}
```

というように、registerというキーワードを付けてiを宣言すると、次のようなアセンブラコードが生成される。

```
        movl $1234,%eax
.L2:    cmpl $5678,%eax
        jle .L5
        jmp .L3
.L5:    incl %eax
        jmp .L2
.L3:    ret
```

この場合、メモリの代わりにEAXというレジスタが使用されていることがわかる。

このように、レジスタ変数を使用することにより、高速なコードを生成できるが、CPUに内蔵されているレジスタはあまり多くない。特にx86系のCPUはレジスタが少ないため、どの変数をレジスタに割り当てばよいかをよく吟味する必要がある。

### ■inline宣言

頻繁に使う関数を、いちいちサブルーチンとして呼び出し、そのリターン処理を行うと時間がかかる。その時間を節約するために、サブルーチンを呼び出すすべての個所で、そのサブルーチンの処理を展開してしまう指定がinlineである。ここで、

```
static int disp(char *s)
{
    printf(s);
}

void main(void)
{
    disp("abc");
}
```

というプログラムを、「$ cc -S -O」でコンパイルすると、

```
disp:   ...
    call printf
    ret

main:   ....
    pushl "abc"のアドレス
    call disp
    ret
```

というようなアセンブラソースが得られる。一方、上記のdisp関数を、

```
static inline int disp(char *s)
```

と宣言すると、

```
main:   ....
    pushl "abc"のアドレス
    call printf
    ret
```

というように、disp関数の呼び出しの代わりに、disp関数の中身がそのままインライン展開されることになる。

上記のように、register宣言やinline宣言をすることで高速化をはかることができる。しかし、明示的に宣言を行わなくても、コマンドラインのオプティマイズオプションによって、同様の効果を得られることもある。gccのマニュアルを読みながらいろいろと試してみると、理解が深まるだろう。

# Perlスクリプト入門
## すぐにできる実践Perl

**今回はくり返し処理を行うために必要なループと、テキストファイルからデータを読み込むための方法を解説します。また、同じ処理を実現するための方法がいくつもあることを理解していただきたいと思います。**


## 第2回　ループとファイル入力

文：おもてじゅんいち／かざぐるま
*Text : Junich Omote/Kazaguruma*

Perlというスクリプト言語の特徴のひとつに、「ある目的を行うためのスクリプトの書き方は1つではない」というものがあります。これは時として、「どう書けばいいんだ」という混乱を招くかもしれませんが、「どう書くのが正しいんだ」という問いかけには、「どれでも正しい」と答えることができます。つまり、どのように書いたとしても、目的を達成できればよいわけで、ちょっと何かを処理したいときに、思いついたようにスクリプトを書くことができます。これがまさにPerlのP、「実践的（Practical）」といえる特徴なのでしょう。

今回は、Perlの「書き方は1つではない」面をかいま見ていただくことにしましょう。

### ループ（くり返し）

前回、スクリプトというものは、文字を表示したり条件を判断するといった、一連の処理を実行するものだということが理解できたと思います。

スクリプトやプログラムは、いわばコンピュータに対する手順書ですから、前回のように、行うべき処理を順にスクリプトとして書いていけば、コンピュータはそれに従って順次実行してくれます。

また、「このファイルを1行ずつ読み込んで、〜せよ」というように、同じような処理を何度もくり返して実行させたいことがよくあります。同じ処理のくり返しというのは、長くなればなるほど人間にとっては苦痛になりますが、コンピュータにとっては得意とするところなので、まさにくり返し処理こそコンピュータに押しつけたいものです。

100回も1000回ものくり返し処理を、その回数分だけスクリプトとして書くのは大変なので、Perlにはwhile文というくり返しのための命令が用意されています。

while文の書式は、

```
while(条件式) {
  実行文;
  実行文;
}
```

となります。(条件式)の部分には、if文と同じような条件式を書きます。while文はその条件式が真である（成り立つ）あいだ、{ }内の実行文をくり返し実行します。このようなくり返し処理部分を「ループ」と呼びます。

たとえば、「1から10までの数をprintせよ」という場合のスクリプトは、

```
$a = 1;
while( $a <= 10 ) {
  print $a;
  $a++;
}
```

となります。今まで学習したことだけで理解できるはずですから、まずは自分でこのスクリプトの動きをじっくり考えてみてください。

まず1行目で、変数$aの値を1にします。次にwhileブロック（whileから}まで）がありますが、このwhileブロックの条件式は、

```
$a <= 10
```

ですから、$aの値が10以下であるあいだ、{ }内をくり返し実行するということになります。最初は$aが1ですから、この条件式が真になり{ }内を実行しますが、{ }内では、

```
print $a;
```

で$aを出力したあと、

```
$a++;
```

で、$aの値を1つ増やしています。{ }内をすべて実行したあとは、再びwhile文の条件式を評価することになりますが、このときは$aが2になっています。今度もまだ条件式の「$a <= 10」が真になりますから、また{ }内を実行します。

こうして、{ }内を実行するたびに$aの値は1ずつ増えて行きます。そして、{ }内にあるprint文によって、ループを繰り返しているあいだに、

```
$a = 1 のとき    print $a
$a = 2 のとき    print $a
$a = 3 のとき    print $a
       :
       :
```

ということを行っています。そして、ついには$aが11となり、while文の条件式の「$a <= 10」が偽になります。すると、もう{ }内を実行することはやめて、{ }のあとに進みます。これを「ループから抜ける」といいます。

以上のことを行っていますから、このスクリプトの実行結果は、

```
12345678910
```

と画面に出力されます。

これで、$aの値を出力するという処理を10回くりかえすために、同じスクリプトを10回も書かずにすんだということになります。もちろん、100回でも1000回でも同じことで、条件式の「$a <= 10」という部分の数値を変えてやれば、何回でも好きなくり返し回数にすることができるのです。

ところで、今のスクリプトは「数字を1から順番に表示する」スクリプトなのですが、数字と数字がくっついているため、見にくいものになっています。これを、

```
1
2
3
:
:
10
```

と、改行しながら表示するにはどうすればよいでしょうか。まずは自分でどのように改良すればよいかを考えてみてください（解答は**リスト1**）。

**▼リスト1　1から10まで改行しながら表示**

```
$a = 1;
while( $a <= 10 ) {
  print "$a\n";
  $a++;
}
```

**■特殊な条件式**

先ほどの例とは逆に、「10から1までの数をprintせよ」という処理をさせたい場合は、

```
$a = 10;
while( $a >= 1 ) {
  print $a;
  $a--;
}
```

というように、$aの最初の値を10にして、条件式を、

```
$a >= 1
```

にしてもよいのですが、$a >= 1 は $a > 0 と同じ意味で

すから、

```
while( $a > 0 ) { ……
```

と書いてもよいことになります。しかしここで、

・条件式は値が0のときは偽として判断される
・while( )は、( )内が真であるあいだループする（偽になるとループを抜ける）

という2点を思い出してください。すると、わざわざ「$a > 0」と書かなくても、条件式として$aだけを書いておけば、$aが10から1である間は、真（0以外の値）ですし、$aが0になったときは偽と判断されます。

つまり、$a自体が条件式として使えるので、このスクリプトは、

```
$a = 10;
while( $a ) {
  print $a;
  $a--;
}
```

とも書けるのです。スクリプトがどう動くのかを自分でも考えてみてください。

ただし、先ほどの1から10まで出力するほうは、いつまでたっても$aが0にならないのでこの方法は使えません。あくまでも、$aが0に向かって行く場合だけです。

このように、最終的に0になる変数があり、その変数が0になったときループをやめて次に進むという場合には、条件式に比較演算子（＝＝、＜＝など）を使わずに、その変数だけを書いておくことができます。この方法は、後述するファイルからのデータ読み込みによく使われますので覚えておきましょう。

とはいえ、自分で書くときはかえってわかりにくい場合もあるので、最初のうちは比較演算子を使ってわかりやすく書くほうがよいでしょう。本に掲載されているスクリプトや、フリーのサンプルスクリプトなどを利用するときに、そのスクリプトを正しく理解するために知っておいてください。

このようにwhileの動きは、

**・( )内の条件式が偽になったらループをやめて次に進む**

**・( )内の値が0になったらループをやめて次に進む**

と覚えておきましょう。

**■無限ループ**

逆に、whileの( )内の条件式が、いつまでたっても0（偽）にならなければどうなるでしょう。

```
while(1) { …… }
```

は( )内が数字の1ですから、まさにいつまでたっても0（偽）になりません。

ようするに、このループはいつまでたっても終わらないことになります。プログラムを終了させることすらできません。これを無限ループといいます。よく、ソフトが「止まった」「フリーズした」などといいますが、実はこの無限ループに陥ってしまって、外見上はウンともスンともいわなくなった状態が多いのです。マウスやキーボードからの信号を受け付けなくなっただけで、内部ではグルグル動いているのですね。

しかし、このwhile(1) というのは、場合によっては便利に使えることもあるのです。たとえば、後述するファイルからのデータ読み込みなどの場合です。

通常Perlでは、ファイルからデータを読み込んで処理を行うときには、

**①ファイルの内容を1行ずつ読み込んで**
**②その1行のデータに対して処理を行い**
**③処理が終わったらまた次の行を読み込んで同様の処理を行う**

というように、ファイルのデータを1行ごとにくり返し処理を行うことになります。

このとき、読み込むファイルによって処理しなければならない回数、つまりファイルの行数はあらかじめ決まっていませんから、ファイルの最終行を処理したら、そこでループから抜けなければなりません。

このような場合、条件式の部分はとりあえずwhile(1)と書いておいて、{ }内の処理の中で、「もしファイルのデータが終わったらループから抜ける」という手順にするのです。本当に無限ループになってしまっては大変ですから、while(1) と書きながら、本当の無限ループにはならない方法を学びましょう。

■ループから抜ける

whileループから抜けるための命令は、

```
last;
```

です。while文の{ }内にこの命令があると、ループの途中であってもループから抜けて、{ }のあとに進みます。while(1)を使う場合は、必ず{ }内にlast;が書かれている必要があります。そうでないと、本当の無限ループになってしまいます。

while(1)とlast;を使ったスクリプトの例は次のようになります。

```
$a = 1;
while(1) {
  print $a;
  $a++;
  if ( $a > 10 ) { last; } # ループを抜ける
}
```

while( )内の条件式は1、つまり常に真になり、ループから抜けることはありません。しかし、その代わり{ }ブロックの最後の行に、if文と、last;があります。ループを抜ける条件は、while文のところには書かずに、ループブロック内に書いておくという方法を採っています。このif文が真になったら、last;を実行する、つまりループから抜けます。

このスクリプトも、1から10までを出力するスクリプトのもうひとつの書き方ですね。

## ファイルからのデータ入力

PerlはC言語など、ほかのプログラミング言語に比べてテキストファイルの読み書きを得意とするため、非常に簡単にテキストファイルの処理を行うことができます。

■ファイルのオープンとクローズ

ファイルを扱うためには、まず、どのファイルを読み書きするのかをファイル名で指定しなければなりません。そのために行うのが「ファイルのオープン」です。また、すべての読み書きが終了したあと、ほかのソフトやOSに対して、「もう終わりました」とファイルを解放することを伝えます。これが「ファイルのクローズ」です。たとえば、ファイル名が「test1.txt」の場合、

```
open(IN,"test1.txt");  # ファイルオープン
     :
     :
close(IN);             # ファイルクローズ
```

と書くだけです。open()とclose()の間で、ファイルを読み書きします（INについては後述）。ファイルをオープンしたときは、必ずクローズするとおぼえておきましょう。

■ファイルから1行読み込む

ファイルから1行読み込むのはもっと簡単です。オープンされたファイルから1行読み込んで、その文字データを$aに格納する場合は、

```
$a = <IN>;
```

でOKです。

まとめると、ファイル「test1.txt」の最初の1行のデータを表示するスクリプトは、

```
open(IN,"test1.txt");
$a = <IN>;
print $a;
close(IN);
```

となります。簡単でしょう。

■ファイルハンドル

ファイルを読み書きするとき、1行読み書きするたびにいちいちファイル名を指定するのは面倒ですし、間違いの元にもなりますから、ファイル名を指定するのは、open()の時だけにして、以後、読み書きやクローズするときは、そのファイルを表すファイルハンドルで指定することができます。先ほどのスクリプトでは、INがファイルハンドルです。

ファイルハンドルは自由な名前にしてもかまいません。しかし習慣的に、変数名やコマンドと見分けるために大文字が使われます。また、読み込みのためのファイルにはINを、書き込みのためにはOUTを使うのが一般的です。

2つ以上のファイルを同時に読み書きしたい場合は、必ずファイルハンドルを変えるようにしてください、そうし

ないと2つのファイルの区別がつかなくなってしまいます。

```
open(KIN,"kingaku.txt"); #1つめのファイルオープン
open(KOSU,"kosu.txt");   #2つめのファイルオープン
          :
          :
$a = <KIN>;   # kingaku.txtから1行読み込み
$b = <KOSU>;  # kosu.txtから1行読み込み
          :
          :
close(KOSU);  # 2つめのファイルクローズ
close(KIN);   # 1つめのファイルクローズ
```

close()の順番はどちらが先でもかまいません。使わなくなった順にクローズするように心がけましょう。

### ■ファイル読み込みとwhile

テキストファイルから1行のデータを読み込む方法は学びましたが、「$a=<IN>」を一度処理しただけでは、最初の1行だけしか読み込めません。

テキストファイルは通常、複数行のデータを持っているため、全部の行を読み込んで処理するためにwhileループと組み合わせて使うことが多いのです。

では、ファイル「test1.txt」の全行を読み込んで、画面に表示するスクリプトを考えてみましょう。

**図1**はその手順を考えたものです。この手順に従って、スクリプトを書いてみます（**リスト2**）。

ここで、ファイルのデータが終わったかどうかを、読み込んだ行の内容が" "のとき、すなわち何もないときをデータの終わりとして判断しています。ファイルを最初から1行ずつ読んでいった場合、文字がまったくなくなる行は、それ以上読み込む文字のない、ファイルの終わりということになるのです。

テキストエディタなどでファイルの内容を見たときには、ファイルの途中にも何もない行が存在するかのように思えます。しかし、ファイルの途中では、文字がまったく存在しない行でも必ず改行が含まれています。この行を読み込んだ場合は、"\n"というデータになります。行を読み込んで" "になるのは、それ以上ファイルに行が存在しない場合だけです。

**図1**の手順それぞれが、**リスト2**の各行に対応していることがわかると思います。whileの行には番号が付いていませんが、これは**図1**上の④から①と②のあいだに伸びている矢印にあたると考えてください。つまり、この矢印がループという動きを表すもので、②、③、④がループブロックの{ }内の処理になります。このループはwhile(1)の無限ループですから、④のif文で条件式を判断して、条件が真の場合はlastでループを抜けます。

さて、**リスト2**のスクリプトを実行してみればわかりますが、実行結果は当初の「ファイルtest1.txtの全行を読み込んで画面に表示する」という目的を満たしています。ですから、このスクリプトはこれで正解です。

しかし、もう少しこのスクリプトを調べてみると、ファイルのデータが終わったとき、②で何もないデータを読み込んで、④の条件式が真になりループを抜けるときに、③も実行されることになります。つまり、" "という何もないデータが画面に出力されます。

何もないデータを表示しても何も起こらないので、表面上の結果は問題ありませんが、" "というデータは何も表示しないとはいえ、本来は表示すべきものではないのですから、単に結果オーライだけでなく、正しく意図したとおり

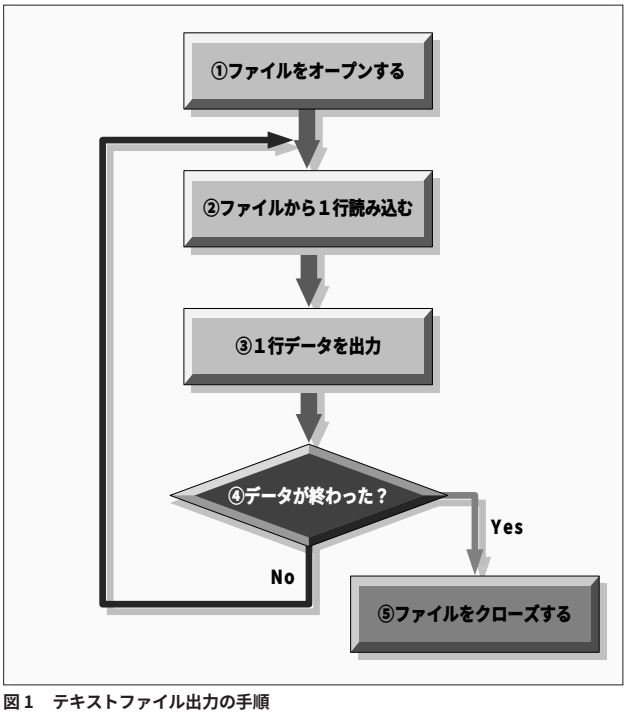



図1　テキストファイル出力の手順

▼リスト2　ファイルを読み込んで表示

```
open(IN,"test1.txt");                    ①
while(1) {
  $a = <IN>;                             ②
  print $a;                              ③
  if ( $a eq "" ) { last; }              ④
}
close(IN);                               ⑤
```

の働きになるように改良しておくことが大切です。

とりあえず今は問題になりませんが、後々このスクリプトをベースにして、色々な機能を付け加えていったときに、思わぬバグの原因になる可能性があります。

では、ファイルの終わりを判断するために読み込んだ、""というデータだけは出力しないようにするにはどうすればよいでしょうか。答は簡単で、printする前にループを抜けるようにすればよいのです（**リスト3**）。

**リスト2**との違いがわかりますか。③と④の順序が逆になっただけです。こうしておけば、データが""ではないとき、すなわちファイルの各行を読み込んだときは、④を通りすぎて、③のprintが実行されます。ファイルの終わりになって、読み込んだデータが""になると、④の条件は真になって、last;が実行されてループから抜けるため、③のprintは実行されません。

### ■空文字列は偽

前回にも、条件式の真偽の説明のところで触れましたが、＝＝やeqなどの比較演算子を使う場合以外に、変数の値自体で真偽を判定することができます。

数値の場合は値が0であれば偽、それ以外の値は真と判断されます。文字列の場合は、空（から）文字列、つまり""のときに偽、1文字でも文字があれば真と判断されると説明しました。

そうであれば、**リスト3**の④の条件式は、もう少し簡単に書けるのではないでしょうか。

```
$a eq ""
```

というのは、「$aが空文字列に等しいかどうか」であり、すなわち「$aが空文字列かどうか」ということですから、

```
if ( $a ) { last; }
```

と、書いてもよいのではないでしょうか。実際、Perlではこのような書き方もできるのです。この場合、$aの値が真か偽かを判断されるだけであり、$aが文字列であれば、空文字列か、文字が含まれているかが判断されます。書き方は問題ないとして、実行結果はどうなるでしょう。

このif文は、「もし$aが真ならループを抜ける」という意味になりますから、いい換えれば「もし$aに文字が含まれていたらループを抜ける」ということになり、期待している結果と逆の動きになってしまいます。こういうときは、真偽が逆になるようにして、

```
if ( !$a ) { last; }
```

と書くと説明しました（前回参照）。

これで、「もし$aに文字が含まれていなかったらループを抜ける」となります（**リスト4**）。

ただしこの書き方の場合、もし最終行が改行なしの"0"のみなら、$a＝＝"0"となり、$a＝＝0と同じように扱われてしまう場合があります。

### ■代入と条件判断を一度に行う

**リスト4**の②で、変数$aには、<IN>によってファイルの1行が読み込まれ、続いて④で真偽を判断されるのですが、**リスト5**のように記述すれば、この2つの処理は一度ですませることができます。

まず、変数$aに<IN>によってファイルの1行が読み込まれ、その読み込まれた値が、if文の判断の対象になりますので、動きとしてはリスト4のスクリプトとまったく同じことになります。

さて、**リスト5**をもう一度眺めてください。そもそもなぜこのループにはwhile(1)という無限ループを使っていたのでしょうか。

**図1**の手順を見てみると、ループを抜ける判断④の前に、②と③の処理を行う必要がありました。つまり、ループの先頭（while文の位置）で条件判断を行うことができなかったので、とりあえずwhile(1)で無限ループにしておいて、{ }内のif文で条件判断を行い、last;でループを抜けるという方法を採ったのでした。

**▼リスト3　""行を表示しないように改良**

```
open(IN,"test1.txt");                    ①
while(1) {
  $a = <IN>;                             ②
  if ( $a eq "" ) { last; }              ④
  print $a;                              ③
}
close(IN);                               ⑤
```

**▼リスト4　文字が含まれていなかったらループを抜ける**

```
open(IN,"test1.txt");                    ①
while(1) {
  $a = <IN>;                             ②
  if ( !$a ) { last; }                   ④
  print $a;                              ③
}
close(IN);                               ⑤
```

しかし、**リスト5**を見てみると、なんと条件判断の行がwhile文の直後までせり上がってきています。**リスト2**以降、何だかんだと工夫をしてきたおかげで、条件判断の部分が、{ }内の一番最初に書けるようになったのです。

**リスト5**の②④の位置に条件判断があるのだったら、これはもうwhile()に条件式を書いても同じことですから、条件式もlast;も含めてwhile文を書きかえてみましょう。

```
while( !($a = <IN>) ) {
```

と書いてしまいそうですが、気をつけてください。よく考えてみると、先ほどのif文では、「真ならばループを抜ける」でしたが、whileの場合は、「真ならばループする」になるのです。このままでは、$aに文字が含まれているときに($a = <IN>)が真になるため、!($a = <IN>)は偽になってしまい、データのあるときにループを抜けてしまいます。正しいwhileの条件式にするには、もう一度真偽を逆にするために!を付けないで、

```
while( $a = <IN> ) {
```

と書くのが正解です。( )も1組不要になります（**リスト6**）。

あらためて**リスト6**の動きを考えてみると、while文のところで、「ファイルから1行を$aに読み込み、それが文字を含む行であればループを続行し、""（ファイルの終わり）であればループを抜ける」という処理になります。

前回、条件式で＝＝のところをうっかり＝と書かないようにという注意をしましたが、**リスト6**の「$a = <IN>」は、条件式のなかで意識的に代入を行っているものですから、これで正しいのです。

### ■デフォルト変数 $_

**リスト2**から刻々と姿を変えてきたスクリプトも、いよいよ大詰めです。まず**リスト7**を見てください。

whileの()内が<IN>だけになり、printのあとの$aがなくなっていることがわかると思います。つまり、スクリプトから$aという変数が消えてしまいました。

では、ファイルから読み込んだ1行のデータはどこに記憶されているのでしょう。また、print文は何をprintするのでしょう。できれば解説を読む前に、このスクリプトを実行して結果を見てください。

実行結果は、$aを使ったスクリプトとまったく同じになります。

whileの()内に<IN>などのファイル読み込みだけを書くと、読み込まれた1行の文字データは、$_という変数に格納されるということが、暗黙のうちに決まっているのです。また、print文においても、ただ、print;だけを書くと、それは「$_をprintする」ということも同様に決まっています。つまり、

```
print;
```

と、

```
print $_;
```

は、同じ意味となるのです。

この$_をデフォルト変数（暗黙変数）といい、whileの()内や、print文、そのほか次回以降で解説するいくつかの場面で変数を省略した場合に、暗黙のうちに使われます。

ただし、あまり暗黙にしすぎるとあまりにも省略部分が多くなりすぎて、かえって混乱することになるかもしれません。最初のうちはきっちりと変数名を書いておいたほうがよいかもしれません。

このスクリプトの例のように、デフォルト変数はただファイルの各行を読んでくるだけといった単純な処理の部分と、次回で説明するパターンマッチの部分ぐらいで使うよ

**▼リスト5　ファイルの読み込みと終わりの判断を同時に行う**

```
open(IN,"test1.txt");                          ①
while(1) {
  if ( !($a = <IN>) ) { last; }                ②④
  print $a;                                    ③
}
close(IN);                                     ⑤
```

**▼リスト6　while内で読み込みと判断を行う**

```
open(IN,"test1.txt");                          ①
while($a = <IN>) {                             ②④
  print $a;                                    ③
}
close(IN);                                     ⑤
```

**▼リスト7　デフォルト変数を使う方法**

```
open(IN,"test1.txt");                          ①
while(<IN>) {                                  ②④
  print;                                       ③
}
close(IN);                                     ⑤
```

うにしましょう。

### ■6つのスクリプト

これで、「ファイルtest1.txtの全行を読み込んで画面に表示するスクリプト」が全部で6つできました。結局最後には、if文もlast;も使わない、非常に単純なスクリプトに行き着いたのです。とにかく手軽に書けて、すぐに実行できるところがPerlの長所ですから、そういう点では最後の**リスト7**のスクリプトが洗練されているといえます。みなさんも、慣れていくにつれて**リスト7**のようなスクリプトが書けるようになるでしょう。

しかし、先に述べたように**リスト7**だけが正解というわけではありません。**リスト2**から**リスト7**まですべて正解です。実行結果はすべて同じとなるからです。自分にとっての正解は、自分が十分理解できていて、いつでも書けるスクリプトの書き方であるということです。

## 応用スクリプト

ここまでの知識で、実用的なスクリプトを書くことができます。

**リスト8**は、「テキストファイルの行数を調べるスクリプト」です。たまに、そんなツールがあったらと思うことはないでしょうか。

**図2**に、このスクリプトがどのように処理しているかの手順を示しておきます。

このスクリプトを実際に動かして、いろいろなテキストファイルの行数を調べてみてください。なお、このスクリプトではファイル名は自由に変えられますが、フォルダ名を書いていないので、調べたいファイルをあらかじめスクリプトと同じフォルダに入れておいてください。

さて、ここまで読んでくださった読者の皆さんは、「ファイルの各行を読み込んで表示」したり、「ファイルの行数を数える」スクリプトぐらいは書ける実力が備わっています。これだけでも、ちょっとしたツールを作ることができますから、簡単なスクリプトではありますが、必ず実際にスクリプトを書いて実行して、また自分なりにいろいろ修正してみてその動きを実感してください。

ここまでの内容は、当然基本的な部分ですから、きっちりとマスターしておかなければなりませんが、基本的なだけにどんなスクリプトを書いていても必要となる部分でもあるのです。ですから、スクリプトを書く経験を積むことによって、自然と身体がおぼえてくれることでしょう。

説明を何度読み返してもよく理解できないことが、実際にスクリプトを書いてみればいっぺんに理解できたということも少なくありません。そして、それをいろいろ変えてみるうちに、応用や禁止事項も自分で発見できるものです。プログラムは動いてナンボのものですから、動かさなければ絶対上達しません。知識として置いておくだけでなく、ぜひ実際に動かしてみましょう。

また、書籍や雑誌に掲載されているPerlスクリプトなどを参考にしながら、どのような処理を行っているのかを考えてみるのも上達の秘訣です。すぐにはすべてを理解するのは難しいかもしれませんが、スクリプトを少しずつ分解していって、それぞれの動きを確認することで、必ずどのような処理を行っているのかがわかるはずです。これも、すぐに実行できるPerlの特徴です。

▼リスト8　行数をカウントするスクリプト

```
$count = 0;
open(IN,"test1.txt"); # ファイル名は自由に
while($line = <IN>) {
  $count++;
}
close(IN);
print "$count行です\n";
```

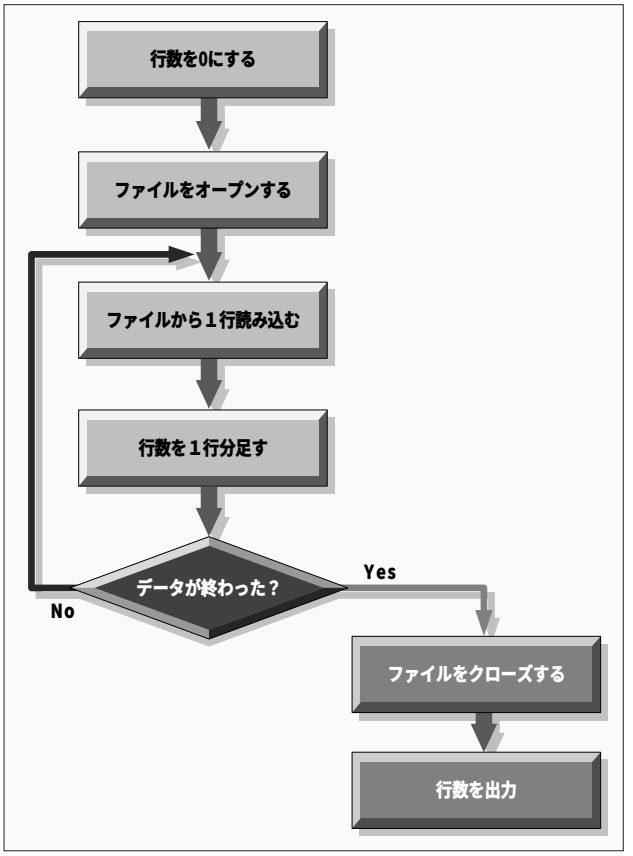



図2　ファイルの行数カウントの手順

# Rubyで行こう

**皆さんの周りの人はRuby使ってますか？　あなた自身がまだ使ってない？　それはいけません。今回は、皆さんと周りの人にどうしてもRubyを使っていただくための戦略について考えます。**


## 第15回　チェーンリアクション

文：赤松智也
*Text : Tomoya Akamatsu*

### あともう一歩のRuby

Rubyって話題になってます。でも、周囲を見まわすと、話題になったり、書籍がベストセラー入りしているわりには、使っている人が少ないような気がします。むしろ、「良さそうなんだけど」とか、「今度使おうと思って」とか言うような人のほうが多いようです。これだけ話題になっているのですから、連鎖反応的にRubyが使われるには、もう一歩だと思います。プログラミング言語は使ってナンボですから、ここはもう一押しのプロモーションが必要そうです。

### ホワイ・ミー

世の中には2種類の人間がいます。それは、「新しいプログラミング言語を学ぶのが苦にならない人」と「そうでない人」です。

前者にRubyを勧めるのは簡単です。そういう人はおおむねプログラミング言語に関心がありますから、Ruby本（**注1**）やハチドリ本（**注2**）を始めとするRubyに関する書籍や記事を見せれば、あとは自分で学んでくれます。本連載などは、結構効果的ではないでしょうか。

結果としてその人がRubyを選ぶという保証はないですが、それはその人の選ぶことなのでかまわないでしょう。このタイプの人は新しもの好きのわりには、他人から強要されることが嫌いな人が多いようですから、強く勧めすぎると人間関係を壊す恐れがあります。

一方、「新しい言語を学ぶのが比較的おっくうな人」たちに、プログラミング言語を勧めるのはそれなりに工夫が必要です。このタイプの人には、新しいプログラミング言語を学ぶための動機、あるいは理由が必要なのです。

とは言うものの、そもそもプログラミングに関心のない人に、プログラミング言語を勧めるのは無理があります。本誌はLinux magazineですから、一般誌よりはプログラミング好きの割合は多いでしょうが。

ここでは対象を「プログラミングに関心がある。好きである。だけどこれから新しい言語をわざわざ学ぶには少々抵抗がある」人に絞りましょう。対象を限定せず、プログラミングの楽しさについて語り出すと、何の連載だかわからなくなってしまいますから。

### 言語を選ぶ・選ばれる

人はどのようにしてプログラミング言語を選ぶのでしょうか。アプリケーション組み込みの言語とか、目的限定の

注1　『オブジェクト指向スクリプト言語Ruby』まつもとゆきひろ・石塚圭樹共著　アスキー　ISBN4-7561-0276-X
注2　『Rubyプログラミング入門』原信一郎著　まつもとゆきひろ監修　オーム社　ISBN4-274-06358-2

言語（レポート用言語など）については、選択の余地が非常に少ないでしょう。そのアプリケーションを使いたかったり、あるいはその目的を果たしたければ、その言語を使うことがほぼ自動的に決まります。

ここでは、そういう限定された条件は考えないことにしましょう。そうすると、理論だけから言えば、プログラミング言語はあらゆるアルゴリズムを記述できるので、極端な話、どのプログラミング言語を使っても、違うのは手間だけと言うこともできるでしょう。

しかし、その「手間だけ」が大問題です。人間の能力には限りがあるので、「手間の差」によって「現実的に不可能」ということも起こり得ます。

では、現実的な観点からは、プログラミング言語を選択する理由には、どのようなものがあるでしょう。ざっと考えると、「技術的な理由」と「非技術的な理由」があるでしょう。「技術的な理由」とは客観的に決定できるような理由であり、「非技術的な理由」とは、それ以外の要因によって決まる理由です。それぞれについて、もう少し詳しく見てみましょう。

## 技術的な理由

プログラミング言語を選択する場合の技術的理由には、以下のようなものが考えられます。

- **開発効率（言語そのものの素性）**
  **その言語でのプログラムの開発が、どれくらい楽で、手早くできるか。**
- **ツールの品質（開発環境、コンパイラ、インタプリタ）**
  **開発効率は、言語そのものの素性のほかに、その言語での開発を支援するツールの品質にも依存する。**
- **インターフェイス・ライブラリ**
  **その言語から使えるライブラリがどのくらいあるか。多くの場合は言語を選択する最大の理由になる。**
- **速度（実行性能）**
  **その言語で記述されたプログラムがどの程度高速に実行できるか。その言語（の処理系）がインタプリタ型かコンパイラ型か。**
- **移植性**
  **その言語で書かれたプログラムをほかのプラットフォームで動作させることがどれだけ簡単か。**

## 非技術的な理由

プログラミング言語の選択理由は、客観的なものばかりではありません。どちらかというと非技術的な理由を考えてみると、たとえば以下のような言語を選択することが考えられるでしょう。

- **最初に勉強した言語**
  **人間は不精なので、新しいものを学ぶのがおっくうなときが多い。**
- **身近に詳しい人がいる言語**
  **やはり、教えてくれる人が近くにいるのはありがたい。**
- **かつて仕事で必要だった言語**
  **仕事は学ぶ動機になることが多い。お金をもらえないのに勉強するのは変人か。**
- **メンテするプログラムを記述している言語**
  **プログラムを読む必要性は、プログラム言語を学ぶ必要性となる。**

これらをまとめると、要するに「人間は怠け者なので、言語を選ぶときも楽したい」ということと、「適切な動機があれば学ぶ」ということが言えそうです。考えてみると、プログラミング言語の切り替えということは、人間の言語ほどではないにしても、それなりに脳に負担がかかることのような気がします。一度学んだ言語に執着したがるのは、人間の怠け者としての性質が、無意識のうちに出ているのではないでしょうか。私自身も、すっかりRubyに執着してますし。

要するに「Rubyを使えばもっと怠けられる」ことを実証すればよいわけです。さらに「Rubyを学ぶのはたいへんじゃない」ことも示せればバッチリです。自信はありませんが、やってみましょう。

## Linuxでお勧めのプログラミング言語

その前に、これからプログラムを始める、またはこれからプログラミング言語を選ぶ方のために、Linuxでお勧めのプログラミング言語をまとめておきましょう。

### ■C

やはり、Cは押さえておきたい言語です。なぜかというと、Linuxカーネルそのものを含め、多くのソフトウェア

がCで記述されているからです。私自身は、プログラミングを学ぶ最も良い方法は、プログラムを読むことだと信じています。ですから、最も多くの「ソースが手に入る」言語のひとつであるCを押さえておくことは重要です。

Cは最良の言語ではないような気がしますが、世の中にある言語の中ではマシなほうでしょう。プログラミングを始めるなら、いろいろな意味で勉強して損はありません。

### ■Perl

結構いろいろなツールがPerlで書かれていますから、一応Perlも挙げておきます。ただし、Perlの場合、読めるプログラムは、さして勉強もせずにすらすら読めます。基本的なPerlは、BASICみたいなものですから。しかし、読めないプログラムは、たとえどんなに勉強しても決して読めませんから、一生懸命勉強する必要はないでしょう。

世の中にはRubyがありますから、今さら一生懸命Perlに力を入れても報われません（と、私は信じてます）。

### ■sh

コマンドラインで指定することをただ繰り返したり、簡単な条件判断を行うだけならば、sh（Bシェル）は有効な手段です。bashとかzshという名前の場合もありますが、ほとんど一緒です。csh（Cシェル）と呼ばれるシェルもありますが、使ってはいけません。あれは、化石世代のためのもので、これから学ぶ人が選ぶものではありません。

shである程度以上の規模のプログラムを書こうとしてはいけません。shは、ある種の目的限定プログラミング言語ですし、そういう冒険は、これからの人がするもんじゃありません。年寄りに任せておきましょう（わしも若いころはムチャをしたもんだが……）。

### ■Ruby

そして、われらがRubyです。Rubyがなぜ良いかという話は、これからたっぷりします。

お勧めプログラミング言語の次点としては、以下のものを挙げておきます。

### ■C++

最近はC++で書かれたソフトウェアもそれなりにありますから、C++を勉強するのにも意味はあるでしょう。もっとも、C++の複雑さに頭を悩ませるくらいなら、ほかにすることがあるだろう、という気もします。

### ■Java

最近の（というかちょっと前の）話題の言語といえば、Javaでしょう。が、私個人はJavaには魅力を感じなくて使っていません。したがって、特にコメントはできません。

### ■Python

Rubyに似た対象領域を持つ言語で、Rubyよりもかっちりした言語という印象があります。私自身はよく知らないのですが、Rubyより前からあって、海外ではRubyよりもはるかに広く使われているのだそうです。

## 開発効率

人により基準は違うでしょうが、私がプログラムを作る

Column

### Rubyを256倍使う本　極道編

うーむ、「邪道編」の次は「極道（ごくどう）編」ですか……。と思ったら、「きわめみち編」なんだそうで。

内容はというと、最近話題のXPこと「eXtreme Programming（エクストリームプログラミング）」の話題です。いや、XPには本当はいろんな要素があるんですが、そのうちのユニットテストについて特に解説した書籍です。ユニットテストというのは「初めにテストありき」というのが標語なのだそうです。つまり、「クラスを書く前に、そのクラスのテストプログラムを書く」という一種の逆転の発想に基づいたプログラミングです。これなら、いつでもテストできて安心ですね。目からウロコでした。

『Rubyを256倍使う本』といいつつ、Rubyに絡めて違うことを解説する本というのは、「邪道編」の路線を継承しているわけですが、個人的にはこの路線は成功していると思います。どうやら『Rubyを256倍使う本』シリーズは、噂の「王道編」を含めてまだまだ企画されているようです。今後にも期待したいものです。


Rubyを256倍使う本　極道編
助田雅紀著
アスキー
本体価格1200円
ISBN4-7561-3687-7

ときに一番重視するのは、開発効率です。というのも、私がふだん対象とするようなプログラムでは、プログラムの実行が少々遅くても、ただ待てばよいからです。そんなことは、プログラムを作るためにコンパイル、リンク、テスト、デバッグを行う時間に比べたら、（待ちながらほかのこともできるし）微々たるものです。

とりあえず、実例を見てみましょう。少々人工的な例ですが、ファイル内容をそのまま標準出力にプリントするcatプログラム（のサブセット）を、CとRubyの双方で記述します。

**リスト1**の処理内容については、自分で読んでいただくこととしましょう。基本的には引数がなければ標準入力から、引数があればそのファイルをオープンして、1文字ずつ読み込んで出力してます。

**リスト2**のRubyプログラムのほうは少々解説しておきましょう。といっても1行ですけど。ARGFは引数をつなげた仮想ファイルです。これは、Rubyが勝手に用意してくれます。この仮想ファイルのeachメソッドにより、1行ずつ読み込んでます。{と}の間が、各行に対して実行される部分です。そして、読み込んだ行（lineに代入される）をprintしています。これで、catの基本的動作である「引数がなければ標準入力から、引数があればそのファイルをオープンして、その内容をすべて標準出力に出力する」処理が行われます。

Rubyの記述力の高さを感じませんか？　この例題はRubyの得意分野なので、評価はやや不公平かもしれませんが、開発効率におけるRubyの優位性の一端は明らかになったでしょう。なお、このプログラムは、NHKの『クローズアップ現代』でも放映された由緒ある（笑）プログラムです。

Cなどと比較して、Rubyのほうが開発効率が高いというのには理由があります。

▼リスト1　catプログラム（C）

```
#include <stdio.h>

int
main(int argc, char **argv)
{
    int c, i;
    FILE *fp;

    if (argc == 1) {
        while ((c = getchar()) != EOF) {
            putchar(c);
        }
    }
    else {
        for (i=1; i<argc; i++) {
            fp = fopen(argv[i], "r");
            if (!fp) {
                perror(argv[0]);
                continue;
            }
            while ((c = fgetc(fp)) != EOF) {
                putchar(c);
            }
        }
    }
}
```

▼リスト2　catプログラム（Ruby）

```
ARGF.each{|line|print line}
```

### ■Rubyにはコンパイルが不要

インタプリタ型のRubyでは、CやC＋＋、Javaなどで必要なコンパイルとリンクのステップが不要です。「プログラムの実行は待てばよい」と述べましたが、コンパイルとリンクを待つ時間は妙にイライラするものです。実行して結果を確かめたいとあせる気持ちが、イライラを刺激するようです。インタプリタなら、エディタでプログラムを編集したら、すぐに実行することが可能です。開発サイクルが早まるだけでなく、精神衛生にも良さそうです。

### ■Rubyには高度なライブラリが標準添付

Cの場合、ANSI Cの範囲では、入出力と文字列操作のライブラリが少々提供されるだけです。POSIXを含めても、提供されるものには限界があります。一方、Rubyなら文字列、正規表現、配列やハッシュといった、高度かつ高機能なデータ構造が無料で付いてきます。

Cプログラマーなら、リンクリストを構造体を使って何度も書いた経験があると思います。Rubyなら、同様のことが、配列を使って一瞬で記述できます。大きさを途中で変えたり、要素を自由に追加することのできないCの配列ではこうはいきません。

また、上記例題であれほど差がついたのも、ファイルの読み込みなどに関する機能が、Rubyのほうが圧倒的に強力だからです。

### ■Rubyではメモリ管理が不要

メモリ管理で苦しんだことはありませんか？　まだ使っているヒープ領域をfree（）したり、free（）を忘れてメモリが足りなくなったことはありませんか？　私はありま

す。メモリ関係のバグは見つけにくいので、非常に苦労します。

しかし、Rubyならメモリ管理はインタプリタがやってくれます。ガベージコレクタ（GC）と呼ばれる機能が、使われていない領域を勝手に見つけて解放してくれます。プログラマーは、ただオブジェクトを割り当てるだけでよいのです。後片付けは全自動で行われます。

また、Rubyを使っていて嬉しいことは、小さなプログラムから大きなプログラムまで、Rubyで楽に書けるということです。私、そしてたぶんほかの人々も、いつも同じ言語で仕事したいというのが本音です。Rubyならそれができるのです。私は、日常のプログラミングは、シェルのほうが明らかに楽な数行程度のプログラムや、既存のCプログラムのメンテナンスをするなどの例外を除いて、すべてのプログラムをRubyで作成するようになりました。

私の書くプログラムは、せいぜい数行程度の小さいプログラム（ワンライナー。これが一番多い）から、数百行程度の中規模プログラムがほとんどです。大きいプログラムはさっぱり書かなくなりましたが、それは歳を取って大きなプログラムを書くための気力が減ったためではなく、Cで書けば数千行になりそうなプログラムも、Rubyで書けば数百行で収まってしまうからです。

## 実行性能

気になる実行性能のほうはどうでしょうか？　**リスト1**のCプログラムをコンパイルしたものと、**リスト2**のRubyプログラムを実行して比較してみました。Rubyの全ソースコード（65052行）を読み込ませた実行結果は、Pentium III 650MHzのPCで動作するLinux上で以下のとおりでした。

```
Cプログラム      0.384秒
Rubyプログラム   0.897秒
```

この結果をどう考えますか？　たしかにRubyのほうが倍以上遅いのですが、たった倍とも言えます。しかも、その差はわずか0.513秒。ほとんど一瞬です。**リスト1**と**リスト2**のプログラムを開発する手間を考えると、0.513秒差では割に合わないと思います。**リスト1**のプログラムをバグなしで一気に書き上げる自信は、私にはありません。タイプミスや勘違いで、何度もコンパイルとリンクを繰り返しそうです。しかし、**リスト2**のプログラムなら、ほぼ間違いなく1回で実行できそうです。開発時間の差は、数倍では収まらないでしょう。

これは、小さいプログラムの極端な例なのでしょうか？　実際には、Cプログラムは規模が大きくなるにつれ、複雑さがどんどん増します。個人的には、差はますます開くように思います。

「ムーアの法則」をご存じですか？　「コンピュータの性能は18か月で2倍になる」という内容なのですが、ここ30年間この法則は当てはまり続けているのだそうです。3年で4倍、6年で16倍、9年で64倍、30年では、えーと、約100万倍（1024×1024）ですね。

LSIのパターン間隔が原子サイズに近づいたり、熱密度が核反応炉に近づいたりと、「ムーアの法則」は物理的限界に近づいているようですが、ハードウェアがどんどん高速で安価になってきているのは事実です。すでにパソコンショップに並ぶコンピュータは、一昔前のスーパーコンピュータを超えそうな性能のものばかりです。そろそろ、このありあまるパワーを、人間に優しくするために使ってもよいのではないでしょうか？　スーパーコンピュータによる、巨大データを扱う数値計算などの限られた分野を除けば、プログラムの実行効率よりも、開発効率のほうを重視しても良い時代がきていると思います。

しかも、あとで述べる「ある方法」を使って、Rubyで巨大データを扱う例も出てきています。Rubyの適用分野はますます広がるわけですね。これで「どこでも同じ言語」という「怠け者プログラマー」の理想にまた一歩近づいたわけです。

## インターフェイス・ライブラリ

おもちゃのように小さいプログラムでない限り、自分ですべてを作り上げることはまれで、だれかが以前に用意してくれたライブラリを利用することで、楽をしようとするはずです。プログラミングのコストから考えても、人間の怠けたという性質から考えても、これは現実的です。

ということは、目的とする機能を持つライブラリが提供されている言語を選択するということは、ほとんどの場合妥当な判断です。実際、プログラミング言語を選択する技術的な理由で最大のものはこれです。なかには、ライブラリを作ることを趣味とする人もいるのでしょうけれども、そういう人は例外と言ってよいでしょう。

そこで、ライブラリが提供されている言語を考えてみると、ほとんどのライブラリはC向けに提供されています。

まれにC＋＋向けのものもありますが。しまった。このままではRubyは圧倒的に不利です。どうしましょう。

そんなあなたに朗報が。実は、Rubyは、CやC＋＋で書かれたライブラリによって機能を追加することができるのです。このようなライブラリを「拡張ライブラリ」と呼んでいます。拡張ライブラリは、CやC＋＋で記述されていますから、それらの言語の機能をすべて利用することができます。Ruby用の拡張ライブラリを書くのはそんなに難しくありません。利用したい関数やメソッドごとに、引数と戻り値に対して、RubyとCの相互型変換を行う関数を書けばよいのです。

たとえ、全部Rubyで行うには実行速度が問題になるような局面でも、本当に速度が要求されるのはごく一部です。ですから、そのごく一部だけこのような拡張ライブラリで実現すれば、問題は解決です。それが前に述べた巨大データなどに対応する「ある方法」だったのです。

拡張ライブラリは、既存のCまたはC＋＋向けのライブラリをアクセスすることもできます。Rubyから既存のライブラリをアクセスさせるための拡張ライブラリは「ラッパー」とも呼ばれます。慣れれば数時間の作業で、そこそこのラッパーが書けます。Rubyのソースコードの中にあるREADME.EXT.jpファイルには、ラッパーの作り方が解説されています。

また、ラッパー作成をさらに楽にするためのツールもあります。SWIG（Simplified Wrapper and Interface Generator）という、インターフェイス記述（Cのヘッダファイルに近い内容）を与えると、ラッパーのソースコードを出力してくれるツールがあるのですが、このツールのバージョン1.3a4以降では、Perl、Python、Tclなどとともに、Rubyにも対応しているのです。ただし、Ruby対応は最新リリースの1.3a5でもまだバグがあったのですが、「CVS版」ではちゃんと使えました。

SWIGのCVS版は、CVSを使ってネットワーク経由で入手できます。入手とコンパイル、インストールは以下の手順で行います。

```
$ cvs -d :pserver:cvs@swig.cs.uchicago.edu:/cvsroot
login
(Logging in to cvs@swig.cs.uchicago.edu)
CVS password: cvs ←パスワードは表示されない
$ cvs -d :pserver:cvs@swig.cs.uchicago.edu:/cvsroot
co SWIG
――ソースコードがSWIGディレクトリに展開される――
$ cd SWIG
$ autoconf
$ ./configure
$ make
$ su
# make install
```

たとえば、簡単な例では、**リスト3**のようなインターフェイス記述から、libmの関数のためのラッパーを出力できます。このラッパーは、libmの定義している機能を提供するLibmモジュールを定義するものです。Libmモジュールは、定数PIとE、それからcosからldexpまでのモジュール関数を提供します。戻り値や引数の型変換などは、SWIGが自動的に行ってくれます。

実際にラッパーを生成するには、このファイルに対して以下のようにSWIGを起動し、ソースコードを生成します。

```
$ swig -ruby -feature libm -o libm_wrap.c libm.i
```

それからMakefile自動生成用のextconf.rbファイルを用意します（**リスト4**）。extconf.rbの内容は簡単です。先

**▼リスト3　libm.iインターフェイス記述**

```
%module Libm
%{
#include <math.h>
%}

%include typemaps.i

const double PI = M_PI;
const double E = M_E;

double cos(double);
double sin(double);
double tan(double);
double exp(double);
double log(double);
double log10(double);
double sqrt(double);
double frexp(double x, int *OUTPUT);
double ldexp(double,int);
```

**▼リスト4　extconf.rb**

```
require 'mkmf'

if have_header('math.h') and have_library('m', 'fmod')
  create_makefile('libm')
end
```

頭でmkmfライブラリをrequireし、あとは、

```
have_header(ヘッダ名)
have_library(ライブラリ名, ライブラリ中の関数名)
have_func(関数名)
```

などで、ラッパーを用意するライブラリがインストールされているかどうかをチェックします。これらの関数は、それぞれ対象が見つかれば真を返します。そして、目的のライブラリがインストールされていれば、

```
create_makefile(ラッパー名)
```

を呼び出せば、Makefileが自動生成されます。ラッパー名は、そのラッパーをRubyから呼び出すときのrequireする名前になります。

extconf.rbが用意できれば、それを実行してMakefileを生成します。

```
$ ruby extconf.rb
checking for math.h... yes
checking for fmod() in -lm... yes
creating Makefile
```

あとはmakeを実行すれば、コンパイルまで行ってくれます。インストールにはmake installすれば完了です。ね、簡単でしょう？

SWIGのより詳しい使い方については、http://www.swig.org/を参照してください（英語ですけど）。

## Rubyを使わない理由・使う理由

さて、Rubyを使わない理由と使う理由を簡単にまとめてみましょう。「Rubyを使わない理由」は、

- **知らない**
- **慣れ親しんだ言語を変えたくない**
- **遅い**
- **ライブラリが使えない**

というようなものです。しかし、最初の2つの「非技術的理由」は単なる無知であり、また、あとの2つが「誤解」か、あるいは間違いではないとしても、多くの場合問題ではないということも示しました。

逆に「Rubyを使う理由」として宣伝したいのは、

- **生産性の高さ**
- **ライブラリ利用の簡単さ**

です。これらを十分宣伝できれば、Rubyに対する抵抗も少なくなるのではないでしょうか？

現在、マシンパワーの向上などで、Rubyで対応できない領域はしだいに減り続けています。仕事などの都合でプログラミング言語を選択できない場合を除けば、Rubyに移行した人はもっと幸せになれる可能性が高いのです。

## 自分のためのプログラミング

「Rubyを使わない人たち」に共通しそうな点について考えてみましょう。私は職業プログラマーですが、私の周りでも「仕事のためのプログラムは作るが、自分のためのプログラムは作らない」という人がたくさんいます。

自分が使うプログラムは、ほかのだれかが仕事で作ったプログラムで、使いにくければそのプログラムが許す範囲内で細々とカスタマイズするか、我慢するかしかありません。なかにはEmacsのように究極のカスタマイズを許すソフトウェアもありますが。

なんともったいない。もちろん、ほかのだれかが作ったソフトウェアを否定するつもりはありませんが、自分のために、自分の好みに合わせたツールを作る快適さは格別です。気に入らなければ、気に入るように作り変えればよいのですから。

「自分のためのプログラミング」は、自分の環境を快適にするだけでなく、自分のプログラミングスキルを向上させる効果もあります。しかし、このようなプログラミングを行うためには、生産性の高いプログラミング言語を用いることが必要です。簡単なことをするだけなのに、開発の手間がかかってしまっては、何のためにプログラミングするのかわかりませんし、そもそも開発を続けるためのやる気も維持できないでしょう。

こういう目的にこそ、Rubyはぴったりです。すばやい開発サイクル、高い生産性、そしてオブジェクト指向による再利用性（これはちょっと疑わしい）。たとえば、本連載の第12回（2000年12月号の『ポストマン』）で紹介したPOPへアクセスするツールをCで開発したら、完成するまで何週間もかかるでしょう。Rubyなら数時間です。

## 得意言語別アプローチ法

では、もっと具体的に、それぞれの人の得意言語別に独断と偏見でアプローチ法を考えてみましょう。

### ■Cな人に

「何をするにもCで」という人は、やはり実行効率を重要視するタイプでしょう。マシンパワーを最大限に使わないと気がすまないのです。たしかに、コンパイル型言語としてCは高速です。

この場合には、上でも紹介したような実行速度と開発効率のトレードオフを見せるとよいでしょう。自虐的な人以外は、速度がさほど重要でない局面では、開発効率を選ぶようになるかもしれません。

### ■C＋＋な人に

C＋＋な人への対応も、「Cな人」に対する対応とさほど変わらないでしょう。ただ、C＋＋は一応オブジェクト指向言語ですから、それに加えて、オブジェクト指向言語としてのC＋＋とRubyの比較を示すことができるでしょう。

Rubyのオブジェクト指向機能をC＋＋のそれと比較す

▼リスト5　list.cc

```
#include <iostream.h>

class MyListData {
public:
    virtual void print(void);
};

void
MyListData::print(void)
{
    cout << "No print method\n";
}

class MyListElem {
    MyListData *data;
    MyListElem *next;
public:
    MyListElem(MyListData *initdata) {
          data = initdata;
          next = 0;
    }
    friend class MyList;
};

class MyList : public MyListData {
    MyListElem *head;
    MyListElem *tail;
public:
    MyList(void) {
          head = tail = 0;
    }
    void add_to_list(MyListData *data);
    void print(void);
};

void
MyList::add_to_list(MyListData *data) {
    MyListElem *newelem = new MyListElem(data);
    if (head == 0)
          head = newelem;
    else {
          tail->next = newelem;
    }
    tail = newelem;
}

void
MyList::print(void)
{
    cout << "<MyList:\n";
    for (MyListElem *e = head; e; e = e->next) {
          e->data->print();
    }
    cout << ">\n";
}

class MyInt : public MyListData {
    int i;
public:
    MyInt(int init) {
          i = init;
    }
    void print(void);
};

void
MyInt::print(void)
{
    cout << i << "\n";
}

class MyPoint : public MyListData {
    int x, y;
public:
    MyPoint(int initx, int inity) {
          x = initx;
          y = inity;
    }
    void print(void);
};

void
MyPoint::print(void)
{
    cout << x << "@" << y << "\n";
}

void
main(void)
{
    MyList *list1 = new MyList;

    list1->add_to_list(new MyInt(10));
    list1->add_to_list(new MyInt(20));
    list1->add_to_list(new MyPoint(2, 3));
    list1->add_to_list(new MyPoint(4, 5));

    MyList *list2 = new MyList;
    list2->add_to_list(new MyInt(20));
    list2->add_to_list(new MyPoint(4, 5));
    list2->add_to_list(list1);

    cout << "list1:\n";
    list1->print();

    cout << "list2:\n";
    list2->print();
}
```

ると、次のような利点（と相違点）があります。

**・Rubyではすべてのデータがオブジェクトである**

C＋＋では、Cに由来する基本的データ型（int、char＊、floatなど）は、オブジェクト指向的な意味でのオブジェクトではありません。一方、Rubyでは統一的にオブジェクトを取り扱うことができます。

**・Rubyではすべてのオブジェクトがヒープに置かれる**

C＋＋では、オブジェクトはスタック上にも、ヒープ上にも置くことができます。これは、ときとして混乱を招きます。

**・Rubyではオブジェクトのメモリ管理が不要**

C＋＋は、オブジェクトのメモリ領域を解放するためには、deleteオペレータで明示的に指定する必要があります。ということは、プログラマーが、いつまでオブジェクトが有効かどうかを完全に把握している必要があるということです。Rubyでは、使われなくなったオブジェクトはGCが改宗してくれますから、極端な話、忘れてしまっても大丈夫です。

**・Rubyではvirtual宣言が不要**

C＋＋では、サブクラスで再定義されるメソッドはあらかじめvirtual宣言しておく必要がありますが、Rubyではすべてのメソッドがvirtual相当です。

**・Rubyのほうが簡潔**

簡単なリスト操作のプログラムをC＋＋（**リスト5**）とRuby（**リスト6**）とで記述してみました。それぞれのプログラムはまったく同じ動作をします。しかし、プログラムのわかりやすさや簡潔さには、かなり差があるように感じます。

**■Perlな人に**

Perlな人は、「自分のためのプログラミング」のすばらしさは理解しているのではないでしょうか。Perlの問題点は、プログラムが暗号になりやすい点ではないかと思います。もともとプログラムというものは、規模が大きくなれば、全容が把握しにくく読み取りにくくなりますが、Perlではその傾向がいっそう強いように思います。おそらくは、プログラムの挙動が、ひと目ではわからないような構文が多いことと関係があるのではないかと思います。

**▼リスト6　list.rb**

```
class MyElem
  def initialize(item)
    @data = item
    @succ = nil
  end

  def data
    @data
  end

  def succ
    @succ
  end

  def succ=(new)
    @succ = new
  end
end

class MyList
  def add_to_list(obj)
    elt = MyElem.new(obj)
    if @head
      @tail.succ = elt
    else
      @head = elt
    end
    @tail = elt
  end

  def each
    elt = @head
    while elt
      yield elt
      elt = elt.succ
    end
  end

  def to_s
    str = "<MyList:\n";
    for elt in self
      str += elt.data.to_s + "\n"
    end
    str += ">"
    str
  end
end

class Point
  def initialize(x, y)
    @x = x; @y = y
    self
  end

  def to_s
    sprintf("%d@%d", @x, @y)
  end
end

list1 = MyList.new
list1.add_to_list(10)
list1.add_to_list(20)
list1.add_to_list(Point.new(2, 3))
list1.add_to_list(Point.new(4, 5))
list2 = MyList.new
list2.add_to_list(20)
list2.add_to_list(Point.new(4, 5))
list2.add_to_list(list1)

print "list1:\n", list1, "\n"
print "list2:\n", list2, "\n"
```

もうひとつの問題は、Perlのオブジェクト指向機能です。設計者のLarry Wall氏自身が認めているように、Perlのオブジェクト指向機能は「やればできる」というレベルで、とても「簡単に使える」というレベルではありません。くどくど述べるよりも、プログラムを見てもらいましょう。

C++やRubyと同じオブジェクト指向サンプルのPerl版を**リスト7**に載せておきます。比較してみてください。

## まとめ

というわけで、今回は基本に帰って、Rubyをプロモーションすることについて考えてみました。

新しい言語を学ぶことは、ときとしておっくうなものですが、実際にRubyを見てもらえれば誤解も解けて、その良さが伝わると思います。皆さんの周りでも、状況が改善することを願っています。

ところで、『ペイ・フォワード 可能の王国』をご覧になりました？ 「だれかに救われた者は、別なだれか3人を救う」。うーむ。そこで、どうでしょう？ 「オープンソースに助けられた人は、なにかオープンソースプロジェクトに貢献する」というのは。あるいは、「Rubyが役に立った人は、別のだれか3人にRubyを紹介する」ってのもありえますね。今回の記事で、私自身は自分の分を果たせそうです。

**▼リスト7 list.pl**

```
package MyElem;

sub new {
    bless my $self = {}, shift;
    $self->{data} = shift;
    $self
}

sub data {
    my $self = shift;
    $self->{data};
}

sub succ {
    my $self = shift;
    my $succ = shift;
    if (defined $succ) {
         $self->{succ} = $succ;
    }
    else {
         $self->{succ};
    }
}

sub to_s {
    my $self = shift;
    my $data = $self->{data};

    if (!ref($data)) {
         sprintf("%s", $data);
    }
    elsif ($data->can(to_s)) {
         $self->{data}->to_s();
    }
}

package MyList;

sub new {
    bless my $self = {}, shift;
    $self
}

sub add_to_list {
    my $self = shift;
    my $elt = new MyElem(shift);

    if ($self->{head}) {
         $self->{tail}->succ($elt);
    }
    else {
         $self->{head} = $elt;
    }
    $self->{tail} = $elt;
}

sub each {
    my $self = shift;
    my $func = shift;
    my $elt = $self->{head};

    while ($elt) {
         $func->($elt);
         $elt = $elt->succ();
    }
};

sub to_s {
    my $self = shift;
    my $str = "<MyList:\n";

    $self->each(sub{my $e = shift; $str .= $e->to_s()."\n";});
    $str .= ">";
    $str;
}


package Point;

sub new {
    bless my $self = {}, shift;
    $self->{x} = shift;
    $self->{y} = shift;
    $self
}

sub to_s {
    my $self = shift;
    sprintf("%d@%d", $self->{x}, $self->{y})
}

package MAIN;

$list1 = new MyList;
$list1->add_to_list(10);
$list1->add_to_list(20);
$list1->add_to_list(new Point(2, 3));
$list1->add_to_list(new Point(4, 5));
$list2 = new MyList();
$list2->add_to_list(20);
$list2->add_to_list(new Point(4, 5));
$list2->add_to_list($list1);

print "list1:\n", $list1->to_s(), "\n";
print "list2:\n", $list2->to_s(), "\n";
```

# [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、シェルスクリプト内で加減乗除などの数値演算を行うコマンドexprや、bash独自の$((…))構文などの書き方やさまざまな演算子、条件式での数値の比較方法などについて説明したのち、複数のファイルに連番を付けるシェルスクリプトを作成する。**

## 第6回　シェルスクリプトでの数値処理

文：大池浩一
*Text:Koichi Oike*

シェルスクリプトで扱うデータは、ファイル名やテキストなど文字列を対象とすることが多いので、シェル変数は文字列ベースの処理が基本となっている。しかし、中には何らかの数値演算が必要なケースもある。

たとえば、「ファイルサイズの合計を求めたい」とか、「元旦まであと何週間あるか調べる」、あるいは「複数のファイルに連続する番号のファイル名を付ける」といった場合、シェル変数の内容を数値と見なして、足し算や引き算などの数値演算を行わなくてはならない。

そこで、今回はシェルスクリプトで数値演算を行う方法について説明する。四則演算（加減乗除）や剰余、論理演算、ビット演算などが可能だ。対象は整数に限られるため、実数演算が必要な場合は、AWKやPerl、Rubyなどのスクリプト言語に任せよう。

ただし、数値演算を行うシェルスクリプトを書くには、ちょっとしたコツを学ぶ必要がある。最初に述べたように、シェル変数の処理は基本的に文字列ベースだからだ。このことを忘れてスクリプトを書くと、予想しない結果に驚くはめになる。特に、文字列と数値を混ぜ書きできるAWKやPerlの経験がある人は気をつけよう。

なお、Linuxの標準シェルであるbashと、そのベースとなったBourneシェルとでは、数値演算に関する機能に天と地ほどの違いがある。そこで、まずはBourneシェルでも通用する古典的な方法を取り上げ、その後でbash独自の機能について説明することにしよう。

### シェルスクリプトで数値処理を行う

bashのベースとなったBourneシェル（Bシェル）には、数値処理を行う機能が用意されていない。その代わり、外部コマンドexprで数値演算を行い、その結果をコマンド置換で取り込んで利用する。Linuxのディストリビューションでも、/usr/binにexprがあるはずだ。

一方、Linuxの標準シェルであるbashでは、数値演算に関するさまざまな機能がシェル内に取り入れられているため、Bourneシェルより高速な処理が可能だ。数値演算の結果で置換される$((…))構文、数値演算の結果をシェル変数に代入する組み込みコマンドlet、整数型のシェル変数を生成する組み込みコマンドdeclare（-iオプション）がこれに該当する。ただし、これらの機能を使ったスクリプトは、生粋のBourneシェルでは正常に動かない。

以下では、

- **exprとコマンド置換による古典的な数値演算の方法**
- **bashで利用できるさまざまな数値演算機能**
- **条件判断で数値の比較などを行う方法**

について説明した後、数値演算や数値の比較を利用したスクリプトの作成を行う。

### ●exprを利用した古典的な数値演算

exprの書式は以下のような単純なものだ。

```
expr 式
```

引数に式を指定して実行すると、演算結果が標準出力に表示される。式には、加減乗除などを行う演算子を記述できる（**表1**）。このほか、論理演算（AND、OR）や文字列演算（正規表現マッチなど）などの演算子も用意されているのだが、ここでは触れない。

数値と演算子はスペースで区切る必要がある。たとえば、「1＋1」を計算するなら、

```
$ expr 1␣+␣1
2
```

とする。もし、「1+1」とつなげて書くと、exprはこれらを数式ではなく1つの文字列として扱うため、「1+1」がそのまま演算結果として表示されてしまう。

また、表1の演算子のうち、「＊」は、シェルにとって特別な意味（ワイルドカード）を持つ特殊記号なので、そのままではカレントディレクトリの全ファイル（「.」で始まるものを除く）のリストに展開されてしまう。exprの演算子として「＊」を利用するには、両端を「'」で囲む（クォーティング）か、前に「¥」（英語端末では「＼」）を付ける（バックスラッシュエスケープ）必要がある。

たとえば、1日の秒数を計算するには、

```
$ expr 60 '*' 60 ¥ * 24
86400
```

とすればいい。最初の「＊」はクォーティング、次の「＊」はバックスラッシュエスケープにより、ワイルドカードとして扱われるのを防いでいる。これ以降は、見やすさの点からクォーティングを使用する。

複数の演算子を1つの式で使う場合は、演算子の優先順位に注意しよう。通常の数式と同様に、「＊」「／」「%」が「＋」「－」よりも優先される。優先順位が同じなら、式の左から右に向かって順番に評価される。

| 演算子 | 意味 |
|---|---|
| ＋ | 加算 |
| － | 減算 |
| ＊ | 乗算 |
| ／ | 除算（余りは切り捨て） |
| % | 剰余（余り） |

表1　exprで利用可能な演算子（一部抜粋）

演算順序を変更するには、式の一部をカッコで囲めばいい。ただし、「(」と「)」もシェルの特殊記号（サブシェルの実行）なのでクォーティングすることと、演算子や数値とカッコの間もスペースで区切ることに注意しよう。たとえば、「(1＋2)＊3」を計算するなら、

```
$ expr '(' 1 + 2 ')' '*' 3
9
```

とすればいい。

exprの演算結果をシェル変数に代入（格納）するには、シェルのコマンド置換機能を使う。たとえば、1日の秒数をシェル変数hogeに代入するには、

```
$ hoge=`expr 60 '*' 60 '*' 24`
```

とする。最初に「｀」（バッククォート）で囲まれた「expr 60 '＊' 60 '＊' 24」が実行され、続いて、その部分が演算結果で置換された「hoge＝86400」が実行されて、シェル変数hogeに86400が代入される。

また、コマンド置換「｀…｀」の内部では、「$変数名」による変数の参照を行えるから、シェル変数hogeの内容を1増やすには、

```
$ hoge=`expr $hoge + 1`
```

と書けばいい。現在のhogeの値に1足した値がexprで計算され、hogeに代入される（**図**）。

要点をまとめると、

- **exprの引数に式を指定する。**
- **数値や演算子はスペースで区切る。**
- **「＊」とカッコは「'」で囲むか「¥」を前に付ける。**
- **コマンド置換を使って演算結果をシェル変数に代入。**

ということになる。

### ●bashで利用できるさまざまな数値演算

Linuxの標準シェルであるbashでも、exprを使って数値演算をする方法はそのまま利用できる。一方、以下で説明するbashの数値演算機能を使うと、さらに高速な処理

が可能で、スクリプトの見た目もわかりやすくなる。

その代わり、bashの機能を利用したスクリプトは、Bourneシェルしか持たないOSでは動作しない。スクリプトが使われるOSを考慮して、汎用性のあるexprと高速で見やすいbash独自機能を使い分けるといいだろう。

### (1)数値の演算結果で置換される$((…))構文

まずは、$((…))構文から説明しよう。これは、「$((」と「))」で囲んだ式を計算し、その結果で$((…))全体が置換されるという、exprの演算機能とシェルのコマンド置換機能をあわせ持つ構文だ。

たとえば、「1＋1」を計算して、シェル変数hogeに結果を代入するには、

```
$ hoge=$((1+1))
```

とする。exprと違って、数値と演算子の間をスペースで区切る必要はない（区切っても構わない）。

演算結果を画面に表示するには、

```
$ echo $((1+1))
```

のように、echoの引数で$((…))構文を使えばいい。また、「”」でクォーティングした文字列の中でも$((…))構文による数値変換は有効なので、

```
$ echo "1足す1は$((1+1))です"
1足す1は2です
```

といった使い方も可能だ。

利用できる数値演算子は、exprと同じ四則演算に加え、ビット演算が追加されている（**表2**）。数値の比較や論理演算を行う関係演算子も用意されているが、それらについてはのちほど説明しよう。

なお、$((…))構文の中の内容はシェルにより特別扱いされるので、exprの式とは異なる点がいくつかある。たとえば、「＊」や「>」などの特殊文字をクォーティングする必要はない。そのまま、

```
$ hoge=$((60*60*24))
```

のように書くことができるわけだ。

また、$((…))構文でシェル変数の内容を参照する場合は、変数名の前に「$」を付けなくてもいい。たとえば、シェル変数hogeの値を1増やすには、

```
$ hoge=$((hoge+1))
```

とすればいい。$((…))構文は数式であることが決まっているので、「hoge」は文字列ではなく、変数展開すべきシェル変数名として扱われるのだ。

### (2)数値演算の結果をシェル変数に代入するlet

シェル変数に数値演算の結果を代入する場合には、

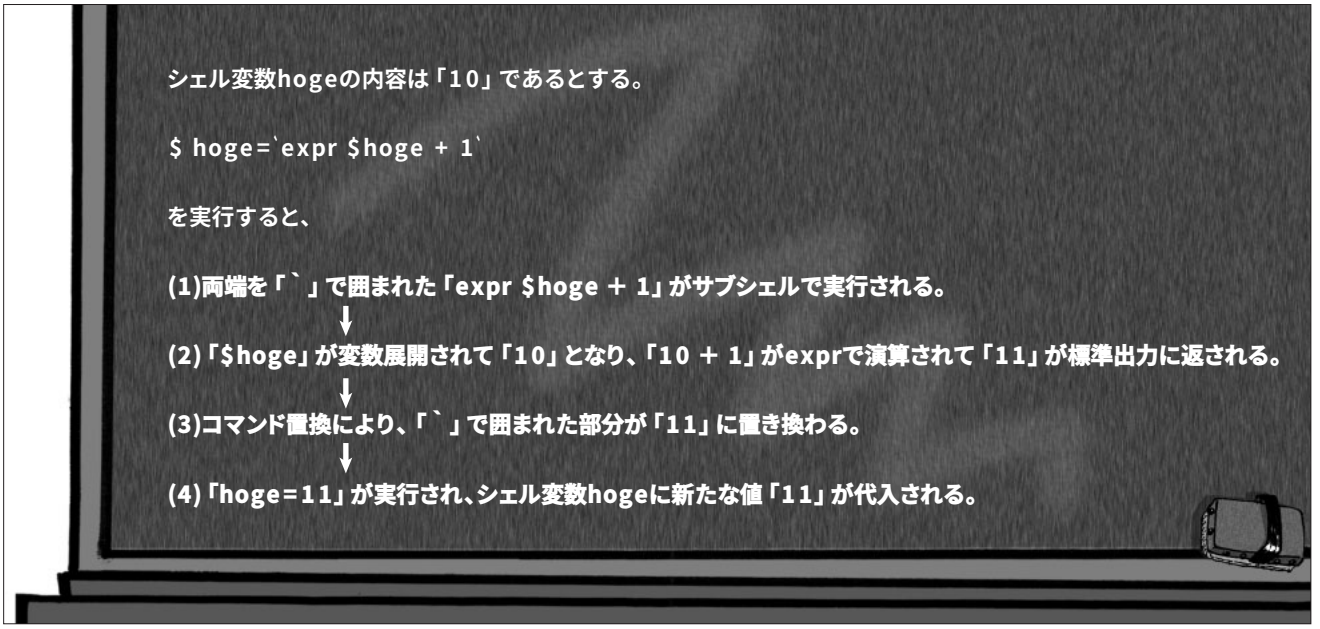



図　exprを使ってシェル変数hogeの内容を1増やす

$((…))構文を使うよりも、組み込みコマンドletを使ったほうがすっきりと書ける。使い方は簡単で、

```
let 変数名=式
```

とすると、式の内容を数値演算した結果をシェル変数に代入する。

たとえば、式「1＋1」を数値演算した結果をシェル変数hogeに代入するには、

```
$ let hoge=1+1
```

とすればいい。また、hogeの内容を1増やすには、

```
$ let hoge=hoge+1
```

とする。$((…))構文と同様、式の中でのシェル変数の参照に「$」を付ける必要はない。

なお、変数名や式と「＝」の間にスペースを入れるとエラーになるので注意されたい。また、「＊」などシェルの特殊文字を式の中で使う場合は、次の例のように式全体を「 ’」でクォーティングする必要がある。

```
$ let hoge='60*60*24'
```

letを使う方法では、式の周囲に余計なものがないため、exprとコマンド置換を組み合わせる方法や、$((…))構文を使う方法よりも式が見やすくなる。

### (3)整数型のシェル変数を生成するdeclare -i

組み込みコマンドletは、式の内容を数値演算してシェル変数に代入するだけなので、文字列ベースというシェル変数の性質が変わるわけではない。

| 演算子 | 意味 |
|---|---|
| ＋ | 加算 |
| － | 減算 |
| ＊ | 乗算 |
| ／ | 除算（余りは切り捨て） |
| % | 剰余（余り） |
| << | 左ビットシフト |
| >> | 右ビットシフト |
| & | ビットごとの論理積（AND） |
| \| | ビットごとの論理和（OR） |
| ~ | ビットごとの否定（NOT） |
| ^ | ビットごとの排他的論理和（XOR） |

表2　$((…))構文で利用可能な数値演算子

一方、組み込みコマンドdeclareの-iオプションを使うと、常に整数しか代入されない「整数型」のシェル変数を定義することができる。

```
declare -i 変数名
```

のように整数型として扱うシェル変数名を指定すればいい（同時に値を指定することも可能）。

たとえば、

```
$ declare -i hoge
```

として整数型のシェル変数hogeを定義すると、これ以後のhogeに対する代入は、$((…))構文や組み込みコマンドletを使わなくても、自動的に式と見なされて数値変換の対象となる。つまり、

```
$ hoge=1+1
$ hoge=hoge+1
```

などと書くだけで、hogeに「1＋1」の演算結果「2」を代入したり、hogeの値を1増やしたりできる。「＝」の前後にスペースを入れてはいけないことや、特殊文字のクォーティングに関してはletと同じだ。

この節での要点をまとめると、

- **$((…))構文がexpr＋コマンド置換の代わりになる。**
- **数値や演算子をスペースで区切る必要はない。**
- **シェル変数の参照に「$」は必要ない。**
- **シェル変数に数値演算の結果だけを代入する場合は、組み込みコマンドletを使うといい。**
- **declare -iで、数値型のシェル変数を定義できる。**

ということになる。

#### ●数値演算を条件判断で利用するには

続いて、数値演算の結果を代入したシェル変数の内容を別の数値と比較し、その結果で分岐処理を行うことを考えてみよう。こうした場合は、if制御構造の条件部で実行されるtest（あるいは[）の引数で、2つの数値の大小関係などを比較する「評価関係演算子」を用いる（**表3**）。たとえば、

```
if [ $hoge -gt 10 ]; then
  echo OK
fi
```

とすると、シェル変数hogeの内容が10より大きい場合にだけechoが実行され、画面に「OK」と表示される。

前回説明したように、test（あるいは[）の条件式では、複数の条件を-a演算子（AND）や-o演算子（OR）で組み合わせることができる。たとえば、

```
if [ $hoge -gt 10 -a $hoge -lt 20 ]; then
  echo OK
fi
```

とすると、シェル変数$hogeの値が10より大きく、かつ20より小さい場合にだけ「OK」と表示される。

ところで、両側の内容が「等しい」かどうか判断する演算子としては、前回紹介した＝演算子がすでに用意されている。今回登場した-eq演算子との違いはいったいどこにあるのだろうか。

＝演算子は、両側の内容を文字列として等しいかどうか比較するのに対し、-eq演算子では数値として等しいかどうか比較する。たとえば、「03」と「3」を比較した場合、長さが違うので文字列としては等しくないが、数値としてはどちらも3なので等しい。「0003」や「＋3」などについても同様だ。

具体的な例として、現在何月かを調べて、月ごとの和名（睦月、如月、弥生、卯月、皐月、水無月、文月、葉月、長月、神無月、霜月、師走）を表示するスクリプトを考えよう。現在の月の数字は、外部コマンドdateの引数に「＋%m」を指定して実行すると得られる。

```
$ date +%m
03
```

スクリプトでは、コマンド置換を使ってこの数字をシェル変数monthに代入し、1から12の数値と順に比較して、それぞれの月の和名を表示すればいい。具体的には以下のようになる。

```
#/bin/sh
month=`date +%m`
if [ $month -eq 1 ]; then
  echo "睦月（むつき）"
elif [ $month -eq 2 ]; then
  echo "如月（きさらぎ）"
elif [ $month -eq 3 ]; then
  echo "弥生（やよい）"
elif [ $month -eq 4 ]; then
  echo "卯月（うづき）"
elif [ $month -eq 5 ]; then
  echo "皐月（さつき）"
elif [ $month -eq 6 ]; then
  echo "水無月（みなづき）"
elif [ $month -eq 7 ]; then
  echo "文月（ふみづき）"
elif [ $month -eq 8 ]; then
  echo "葉月（はづき）"
elif [ $month -eq 9 ]; then
  echo "長月（ながつき）"
elif [ $month -eq 10 ]; then
  echo "神無月（かんなづき）"
elif [ $month -eq 11 ]; then
  echo "霜月（しもつき）"
elif [ $month -eq 12 ]; then
  echo "師走（しわす）"
fi
```

なお、dateが出力する月の数字は2桁に固定されているので、1月から9月までは先頭に「0」が付く（01、02、…）ことに注意しよう。もし、このシェルスクリプトで使われる条件式を、文字列として比較する＝演算子を使って、

```
if [ $month = 1 ]; then
```

といった具合に書いてしまうと、「01」と「1」は文字列としては等しくないので、現在が1月でも「睦月（むつき）」と表示されなくなってしまうのだ。

bash独自の拡張である$((…))構文には、「>」や「＝＝」

| 演算子 | 意味 |
|---|---|
| 数値1 -lt 数値2 | 数値1が数値2より小さい場合に真 |
| 数値1 -gt 数値2 | 数値1が数値2より大きい場合に真 |
| 数値1 -le 数値2 | 数値1が数値2以下の場合に真 |
| 数値1 -ge 数値2 | 数値1が数値2以上の場合に真 |
| 数値1 -eq 数値2 | 数値1が数値2と等しい場合に真 |
| 数値1 -ne 数値2 | 数値1が数値2と等しくない場合に真 |

表3 test（あるいは[）の評価関係演算子

といったC言語風の関係演算子が用意されており（**表4**）、複雑な条件式をすっきりと記述できる。

たとえば、シェル変数hogeの内容が10より大きい場合に「OK」と表示するには、

```
if [ $((hoge > 10)) = 1 ]; then
  echo OK
fi
```

とすればいい。

まず、$((…))構文の中の「hoge > 10」という条件式が評価され、シェル変数hogeの内容が10より大きい（条件式が真の）場合は「1」、10以下の（条件式が偽の）場合は「0」に置換される。よって、[の条件式では、演算結果が1と等しいかどうかだけを調べればいい。

すでに述べたように、$((…))の中でシェル変数の内容を参照する場合は、変数名の前に「$」を付けなくてよい。また、$((…))の中で使われている「>」は、通常は標準出力のリダイレクトに使われる特殊文字だが、$((…))内部の文字列はシェルにより特別扱いされるため、「 ’」でクォーティングする必要はない。

もちろん、$((…))構文では通常の四則演算も可能なので、演算結果を直接数値と比較するような条件式も記述できる。たとえば、シェル変数hogeの内容が奇数の場合に「OK」と表示するスクリプトは、

```
if [ $((hoge % 2 == 1)) = 1 ]; then
  echo OK
fi
```

となる。＝＝演算子と＝演算子に注意しよう。

複数の条件を＆＆演算子や｜｜演算子と組み合わせることも可能だ。たとえば、シェル変数$hogeの値が10より大きく20より小さい場合に「OK」と表示するには、

| 演算子 | 意味 |
|---|---|
| 数値1 < 数値2 | 数値1が数値2より小さい場合に真 |
| 数値1 > 数値2 | 数値1が数値2より大きい場合に真 |
| 数値1 <= 数値2 | 数値1が数値2以下の場合に真 |
| 数値1 >= 数値2 | 数値1が数値2以上の場合に真 |
| 数値1 == 数値2 | 数値1が数値2と等しい場合に真 |
| 数値1 != 数値2 | 数値1が数値2と等しくない場合に真 |
| 条件1 && 条件2 | 条件1と条件2がどちらも成立する場合に真 |
| 条件1｜｜条件2 | 条件1と条件2のいずれか一方でも成立する場合に真 |
| ！条件1 | 条件1が成立しない場合に真 |

表4　$((…))構文で利用可能な関係演算子

```
if [ $((hoge > 10 && hoge < 20)) = 1 ]; then
  echo OK
fi
```

と書けばいい。

C言語の経験がある人は、なじみのある演算子を使えるので、シェルスクリプトを書いたり読んだりするのが楽になるのではないだろうか。

この方法を利用する場合、[の条件式で「＝1」として1と比較することを忘れてはいけない。もし、

```
if [ $((hoge > 10)) ]; then
  echo OK
fi
```

と書くと、$((…))構文の条件式が真の場合は「1」、偽の場合は「0」で置換されるので、[の条件式は「1」または「0」となる。

前回も説明したように、単独の文字列が条件式に指定されると、文字列の長さが評価の対象になる（空文字列だけ偽）。「0」「1」はいずれも空文字列ではないから、[の条件式は常に真となる。つまり、シェル変数hogeの内容に関係なく、画面に「OK」と表示されてしまうのだ。

この節の要点をまとめると、

- **数値演算の結果を条件判断で比較する場合は、test（あるいは[）の評価関係演算子を使う。**
- **bash独自の$((…))構文にも、数値の比較を行うための関係演算子が用意されている。**
- **$((…))構文の中に条件式を書く場合は、test（あるいは[）の条件式で1と比較するのを忘れないこと。**

ということになる。

## 今月のスクリプト

後半は、既存のスクリプトの内容を解説する「スクリプトを読む」と、与えられたテーマを実現するスクリプトの作成手順を説明する「スクリプトを書く」で構成する。今月は紙幅の都合上「スクリプトを読む」はお休みし、「スクリプトを書く」では、

- **指定したファイル（複数可）の名前を、古いものから順**

**に3桁の連番付きの名前に変更する「renum」**

を作成する。デジカメで撮影した画像ファイルなど、大量のファイルを一括リネームするのに役立つスクリプトになるはずだ。

スクリプトの構成は、

**・複数のファイル名を古い順に並び替える**
**・番号を1ずつ増やしながら処理を繰り返す**
**・新しいファイル名を生成して変更する**

という3つの部分に分かれている。以下では、それぞれの処理について見ていこう。

**●ファイル名を古い順に並び替えるには**

実行時のコマンドライン引数のリストを「"$@"」で参照できることは、前回までに説明した。ただし、引数のファイル名はコマンドラインで指定した順番で並んでいるため、作成（更新）時刻の古い順に並び替える必要がある。

ファイル一覧を表示するコマンドlsには、ファイルの作成（更新）時刻順に表示する-tオプションが用意されている。そのままだと新しいファイルが先頭になるため、表示順を逆にする-rオプションと組み合わせよう。

たとえば、並び替えた後のファイル名のリストをシェル変数filesに格納するには、コマンド置換を利用して、

```
files=`ls -tr "$@"`
```

とすればいい。

ユーザーによっては、シェルのエイリアス（別名）機能を使ってlsを再定義していることがある。こうしたエイリアスが今回のスクリプトから実行されることを避けるには、次のようにlsをフルパスで指定すればいい。

```
files=`/bin/ls -tr "$@"` 2> /dev/null
```

末尾の「2> /dev/null」は、標準エラー出力を/dev/nullにリダイレクトするおまじない。これで、lsのエラーメッセージがいっさい画面に表示されなくなる。

**●ファイル名を取り出して繰り返し処理を行う**

シェル変数filesに格納されたファイル名のリストから、ファイル名を1つずつ取り出して処理を行うには、本連載の第2回で説明したfor制御構造が最適だ。

新しいファイル名には、1から始まる番号を付けるので、現在の番号を格納するシェル変数numを用意し、繰り返し処理のたびにその内容を1ずつ増やす必要がある（numの初期値は1とする）。外部コマンドexprとコマンド置換を組み合わせる方法を使うと、以下のように書ける。

```
num=1
for f in $files; do
  ファイル名を変更する処理
  num=`expr $num + 1`
done
```

もし、スクリプトをbashだけで動かすなら、exprとコマンド置換の部分を、「num=$((num＋1))」や「let num=num＋1」と書いてもいい。

**●新しいファイル名を生成して変更する**

最後に、連番を含む新しいファイル名を生成し、元のファイル名から変更する処理を加えてスクリプトは完成だ。新しいファイル名は次の3要素で構成される。以下では、それぞれの処理を行うスクリプトを考えていこう。

**・固定文字列…ファイル名の先頭に共通する文字列**
**・連番…「001」から始まる3桁の番号**
**・拡張子…元のファイル名の拡張子（ピリオド含む）**

(1)固定文字列をコマンドラインで指定

固定文字列は、スクリプト実行時のオプションで「-b 固定文字列」として指定し、シェル変数baseに格納する。そのためには、スクリプトの最初で、

```
if [ "$1" = -b ]; then
  base="$2"
  shift 2
fi
```

と書けばいい。最初のコマンドライン引数（$1で参照）が「-b」の場合に、シェル変数baseに2番目のコマンドライン引数（$2で参照）を格納する。なお、-bオプションを省略した場合は、then以下の処理がスキップされるため、baseの内容は空文字列になる。

また、固定文字列をbaseに格納した後、「shift 2」で

引数の内容を2つ前にずらしていることに注意しよう。これを忘れると、ファイル名のリストを作る際に、「-b」や固定文字列がファイル名として含まれてしまう。

### (2)番号から3桁の文字列を生成

連番の処理では、「1」や「12」といった数値を「001」や「012」のように3桁の文字列に変換する処理が必要だ。そのために、exprの文字列演算子（今回は説明していない）のひとつ、：（コロン）演算子を使う。

：演算子は、左側に指定された文字列を、右側に指定された「正規表現」に基づいて加工した結果を返す（正規表現については回を改めて説明しよう）。今回の目的を達成するには、次のように書けばいい。

```
num=`expr "00$num" : ".*\(...\)$"`
```

これで、シェル変数numの内容（1から始まる数字）の前に「00」を付けた文字列から、末尾の3文字だけを取り出せる。なお、実際のスクリプトでは、numが1000より小さい場合に限ってこの処理を行っている。

### (3)元のファイルから拡張子を取り出す

元のファイル名から拡張子（ピリオドを含む）の部分だけを取り出す処理にも、(2)と同様にexprの：演算子を利用する。具体的には、次のように書ける。

```
ext=`expr \`basename "$f"\` : ".*\(\.[^.]*\)$"`
```

ここでは、2重のコマンド置換が使われている（内部のコマンド置換は「￥｀」で囲む必要がある）。まず、「basename "$f"」により、ループ変数fの内容からディレクトリを除いたファイル名が得られる。続いて、exprの：演算子により、最初の置換結果中の「.」からファイル名末尾まで続く文字列を取り出し、シェル変数extに格納する。

(1)〜(3)の処理を行えば、新しいファイル名を構成する固定文字列（base）、3桁の連番（num）、拡張子（ext）をすべて用意できる。

あとは、mvを使って、

```
mv "$f" "$base$num$ext"
```

とすれば、新しいファイル名に変更される。

なお、実際のスクリプト（**リスト**）では、ファイルが指定されなかった場合のエラー処理（6〜9行目）と、ファイルが通常ファイルかどうかのチェック（13行目）も行っている。このスクリプトを~/binなどにコピーし、ファイル属性を実行可能に変更する。

```
$ cp renum ~/bin
$ chmod +x renum
```

それでは実行してみよう。

```
$ renum -b img *.jpg
```

すると、「＊.jpg」にマッチするファイルが、作成（更新）時刻の古いものから順に、

```
img001.jpg
img002.jpg、
  :
```

というファイル名に変更されるはずだ。

**▼リスト　スクリプトrenum**

```
 1: #!/bin/sh
 2: if [ "$1" = -b ]; then
 3:   base="$2"
 4:   shift 2
 5: fi
 6: if [ $# = 0 ]; then
 7:   echo "usage: rename [-b BASE] files..." >
/dev/stderr
 8:   exit 1
 9: fi
10: files=`/bin/ls -tr "$@" 2> /dev/null`
11: num=1
12: for f in $files; do
13:   if [ -f "$f" ]; then
14:     if [ $num -lt 1000 ]; then
15:       num=`expr "00$num" : ".*\(...\)$"`
16:     fi
17:     ext=`expr \`basename "$f"\` :
".*\(\.[^.]*\)$"`
18:     mv "$f" "$base$num$ext"
19:     num=`expr $num + 1`
20:   fi
21: done
```

目指せEmacsの達人

# Emacsはじめました

## 最終回　プログラミングと言語モード

**UNIXコンピュータの使い道は数々あれど、王道はなんといってもプログラミングでしょう。Emacsは、書いているプログラムの種類に合わせて気働きをしてくれます。というわけで、この連載のトリを飾るテーマは、プログラミングを助けてくれる言語モードについてです。**

*Illustration : Manami Kato*

文：佐々木太良
*Text : Taroh Sasaki*

### プログラミング言語とLinuxユーザー

最終回は言語とモードのお話です。最後だからたいして重要じゃないのかな、なんて思わないでください。これまでにも、Emacsは編集しているファイルの内容によって振る舞いを変える、ということを学んできました。同様に、プログラムを編集しているときにもプログラミング言語によって動作が変わります。そんなわけで、モードという概念は、これまでの回でも知らず知らずのうちに登場していたことになります。

ところでプログラムは作りますか？　UNIXのインストールといえば、昔ならシステム管理者の離れ業だったものですが、今では本誌の付録のCD-ROMでちょちょいのちょいです。反対に、初心者でもすなるものだったプログラミングは、上級者の技と思われるようになったのではないでしょうか。

UNIXの熟達度を測るおもしろい指標が、1986年にネットニュースのnet.unixに流れたことがあります。これを翻訳した「これがUNIX社会の階級だ！！」がfj.jokesに投稿されてアーカイブに残っているので、このあたりの雰囲気を感じ取ってみてください。翻訳とコメントはかの！齊藤明紀先生です。（オリジナル：http://galaxy.rwcp.or.jp/text/cgi-bin/newsarticle2?ng=fj.jokes&nb=14、邦訳：http://galaxy.rwcp.or.jp/text/cgi-bin/newsarticle2?ng=fj.jokes&nb=15）

個人的には、真の意味で「コンピュータを使う」とは、なんらかのプログラミングにほかならないと思っています。アプリケーションを使うだけなら、たとえばワープロソフトであればワープロ専用機や和文タイプライター、メールソフトであればポケットボードといった専用機器の代用をしているだけです。やはり、プログラミングこそ、コンピュータをコンピュータとして使う醍醐味でしょう。

この記事でなにかのプログラム言語自体を説明することはできませんが、教科書を買ってきて始めたとしてもきっとEmacsが助けてくれるので、ぜひ試してください。

そうはいいながら、モードの説明のために実例を出さないといけません。ここではおもに、C言語を使ってみることにしましょう。

### 言語モードの特徴

Emacsにはさまざまなメジャーモードがありますが、ここではプログラミング言語を編集しているときのモードを言語モードと呼びましょう。

まずはC-x C-fでfind-fileしたとき、拡張子によって、c-modeやperl-modeなどに入れます（**表1**）。もちろん、**M-x c-mode**などの操作によって意識的にこれらのモードに入ることもできます。

表1を見ると、C言語やPascalといった伝統的なプログ

ラミング言語のほかに、Javaなどの新しい言語、さらにPerlやシェルスクリプトなどのスクリプト言語まで、Emacsで編集しそうなものが幅広くサポートされていることがわかります。面白いのは、MakefileやChangelog（バージョンアップしたときにどこがどう変わったかをメモしておくドキュメント）といったプログラム開発環境もトータルでサポートしようとしていることです。

最後のTeXやHTMLは、プログラミングというには当たらないかもしれませんが、言語の形をしていればEmacsの得意分野です。YaTeXは第10回（2001年1月号）で出てきましたが、このほかにもUNIXプログラマーには必須のtexinfo、nroff（manで出力されるマニュアルは、nroff -man形式で書かれています）なども簡単に記述できるようになっています。

言語によってそれぞれのモードの動作は異なりますが（だからモードっていうんですね）、大ざっぱにいうとどのモードでも以下のような機能は備えています。

### ■フェイス（見栄え）の変更

一般的に、キーワードや変数などは、文字のフォント（書体）や色などを変えて表示してくれます。これは各モードを起動してみれば一目瞭然ですね。ただし、Emacsはなるべく軽く編集が行えるように設計されているので、挿入や削除を繰り返していると、これが狂ってくることがあります。そんなときには**C-l**を押せば、カレントのバッファのフェイスをもう一度適切に付け直してくれます。

| 拡張子 | 言　語 | 適用されるモード |
|---|---|---|
| .c、.h、.y、.lex | C | c-mode |
| .c++、.h++<br>.cc、.hh、.C、.H<br>.cpp、.cxx、.hxx | C++ | c++-mode |
| .el | elisp | emacs-lisp-mode |
| .java | Java | java-mode |
| .f、.F、.for | Fortran | fortran-mode |
| .p、.pas | PASCAL | pascal-mode |
| .l、.lisp | Lisp | lisp-mode |
| .ada、.adb、.ads | Ada | ada-mode |
| .oak、.scm | Scheme | scheme-mode |
| .prolog | Prolog | prolog-mode |
| .s、.S、.asm | アセンブラ | asm-mode |
| .pl、.pm | Perl | perl-mode |
| .awk | AWK | awk-mode |
| .sh、.sharほか | Shell | sh-mode |
| .tcl、.expほか | Tcl | tcl-mode |
| .mk | Makefileほか | （Makefile） |
| changelogほか | （Changelog） | change-log-mode |
| .texinfo、.texi | texinfo | texinfo-mode |
| .tex | TeX | tex-mode, yatex-mode* |
| .ltx、.sty、.clsほか | LaTeX | latex-mode, yatex-mode* |
| .html | HTML | html-mode, yahtml-mode* |
| .sgml、.dtd | SGML | sgml-mode |
| .mm、.me、.ms、.man | nroff | nroff-mode |
| .text、.articleほか | テキスト | text-mode |

表1　拡張子と主要な言語モード

### ■〈TAB〉を押したときの挙動

viなどの古典的なエディタでは、〈TAB〉キーにはTAB文字を1個挿入するだけの役割しかありません。スクリーンエディタなら端末の動作定義にしたがって、次のタブストップ（8文字おきなど）から次の文字が表示されますが、Emacsでの動作はこれよりもっとインテリジェントです。c-modeでは、ポイントがある行のインデント量を調整せよ、という命令になっています。ポイントが行のどこにあろうと関係ありません。本来TABが入るべきではない場所にポイントがあったとしても、〈TAB〉キーを押したところで空白は空きません。

あまり好きな機能ではないのですが、デフォルトではTAB文字の代わりにスペースが挿入されます。使用している環境によって見栄えが変わることを防いでいるのです。

c-modeでタブストップの幅を変えたければ、.emacs に、

```
(setq-default c-basic-offset 5)
```

のように書いておきます。

TAB文字を強制的に挿入したいときのために、**M-i**（tab-to-tab-stop）という操作がありますが、これとてほかのエディタで〈TAB〉を押すのとは異なります。適切なTAB位置までポイントを進め、それまでの間に可能ならTAB文字、半端が出たらスペースを挿入します。このため、タブストップを8文字ごとに設定してある場合、挿入したTAB文字を後ろから削除しようとしても、**図1**のようにスペース7個に変換されてしまいます。

これが嫌な場合、旧来のエディタと同じようにTABを取り扱ういちばん手軽な方法は、結局c-modeのインテリジェンスを犠牲にしてテキストモードに切り替えることでしょう。**M-x text-mode**と操作してください。.emacsに書く場合は、

```
(setq auto-mode-alist
  (cons (cons "￥￥.c$" 'text-mode) auto-mode-alist))
```

のようにします。

■自動インデント

これも〈TAB〉キーを押したときの挙動と関係があるのですが、c-modeなど大半の言語モードでは、インデント（行頭字下げ）が必要なカッコなどのキーを押したとき、インテリジェントにインデント幅を調整してくれます。

**図2**を見てください。1行目と2行目は、通常のtext-modeと同様にタイプします。

3行目で〈TAB〉をタイプした瞬間、Emacsはこの行が“{……}”のブロックの内側だということを理解しているので、1段すなわち2文字ぶん字下げしてくれます。4行目も同様です。

さて、5行目で〈}〉をタイプした瞬間、Emacsは面白い挙動をします。Emacs内部では、〈}〉は自動字下げ機能と結び付いているため、“}”をいきなり3行目と同じ位置まで揃えてくれます。ちなみに5行目を入力する前に〈TAB〉をタイプしているとすると、ポイントは4行目と同じ位置（2段字下げ）に揃えられますが、その場合でも〈}〉をタイプしたとたんに字下げ量が自動的に1段分減ります。

何はともあれ、実際に**図2**のとおりタイプしてみると、c-mode（そしてほかの言語モード）がなぜ役立つのかよく理解できるでしょう。

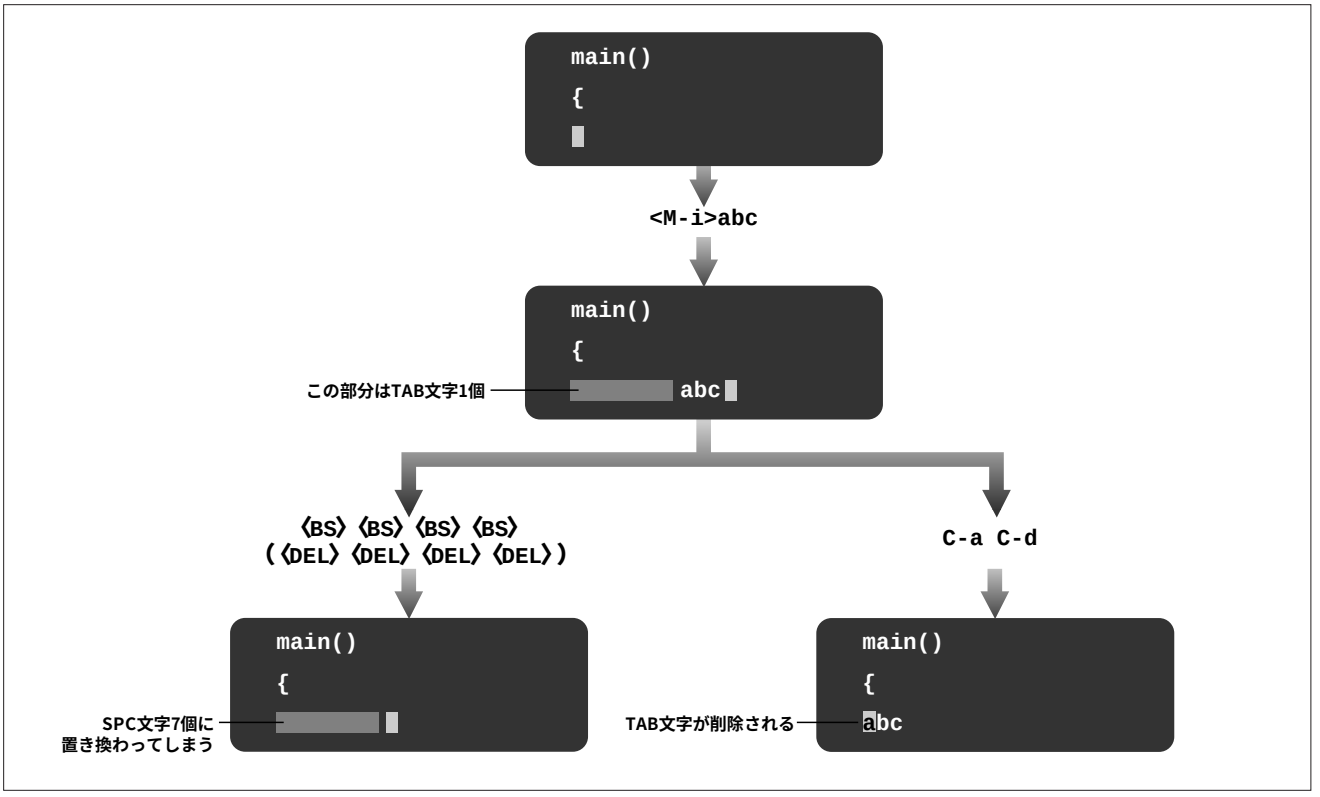



図1 c-modeにおけるTAB文字の働き

## Column

### grepの活用

どんな作業も、grepを知っていると知らないでは効率が大違い。Emacsではこれを内部から利用できます。

**M-x grep**を実行すると、どういうgrepコマンドを実行するか尋ねてきます。プロンプトに“grep -n”と例示されているので、続けて“grep -n hoge ＊.c”と入力してみましょう。hogeは探すキーワード、＊.cは検索対象のファイルです。もうひとつのバッファに結果が表示されるはずです。

これならシェルからgrepを実行したほうが早いじゃないかって？

でもこの方法だと、grepはカレントのバッファと同じディレクトリで実行されるので、まずそのディレクトリに行く手間が省けます。そしてもうひとつ、**C-x`**を使って次に見つかったキーワードに進めるときも、ファイルがオープンしていなければ、言語モードのデバッグよろしく適当にオープンして、該当する行にジャンプしてくれるのです。

このためにファイル名と行番号情報が必要なので、最初に提示される“grep -n”の-nオプションを消去してはいけません。

grepコマンドは、テキストファイルを対象として使えますが、なんといってもプログラミングのときこそ威力を発揮します。

### ■コンパイル

Perlなどのインタープリタ言語はコンパイルの必要はありませんが、Cなどのコンパイラ言語にはコンパイルが必要です。Emacsでプログラムを作成した場合、コンパイルからデバッグまでをEmacsのなかで行うことができます。

**M-x compile**を実行すると、まずセーブしていないバッファをセーブするかどうか尋ねられます。次に、コンパイルコマンドに何を使うかが尋ねられます。デフォルトでは“ make -p ”が提示されますが、makeを使うほど大規模なプログラムでなければ、**C-a**や**C-k**を使って消し、“ cc -o hoge hoge.c ”などを入力してやればよいでしょう。

コンパイル中は、ソースが上のバッファに、コンパイラからのメッセージが下のバッファに表示されます。もしエラーがある場合（**画面1**）、**C-x `**でエラー箇所に次々と飛ぶことができます。下のバッファの最上行がエラーメッセージ、上のバッファのポイントが該当行に置かれるので、すぐに修正ができます。よくある話ですが、エラーはポイントが移動した行にあるとはかぎらないので、前の行のセミコロン、さらにもっと前のカッコやダブルクォートの閉じ忘れに注意してください。

エラーメッセージはc-modeの場合、gccの出力を仮定しているようです。Cの関数名を間違ったために、コンパイラではなくリンカでエラーが出た場合などは、エラー箇所にジャンプする機能は使えません。これは、リンカのエラーでは行番号が表示されないためで、上のバッファ（ソース）を使って自分で該当する関数を検索するなど、シェルからCコンパイルするときと同じ程度の知識は必要となります。でも楽ですよね。

トライ＆エラーでコンパイルを繰り返すときには、2回目以降のM-x compileでは、同じコンパイルコマンドがデフォルトで提示されるので、かなり楽ができます。

### ■実行〜デバッグ

さてCのプログラムができました。実行してみましょう。この場合も、シェルから実行してもよいのですが、せっかくシェルモード（第8回・2000年11月号）を習ったのですから、**M-x shell**を使ってその場で実行してみましょう（**画面2**）。

おや、作ったプログラムが即死（コアダンプ）してしまいましたね。う〜ん、デバッグが必要なようです。Emacsはデバッグ用にgudモードを備えていますが、これがなかなか強力で、シェルからgdbを実行してエディタで見比べるのに比べてかなり簡単な操作でデバッグができます。

gudモードのなかではgdbを使用しますので、オプション-gを付けて“ cc -g -o hoge hoge.c ”のようにコンパイルしなおしてください。次に**M-x gdb**を実行します。ここでどういうコマンドを実行するか尋ねられるので、“ gdb hoge ”をミニバッファに入力します。

バッファ＊gud-mode＊が開いてプロンプト“（gdb）”が出ました。ここから先はgdbを使いこなす知識が多少必要です。たとえば、何行目でコアダンプしたか見たければ、“ run ”と入力します。コアダンプした箇所でgdbは停止しますが、このときウィンドウは2つに割れて、新しく開いたバッファにはソースと停止した行が**=>**印で表示され

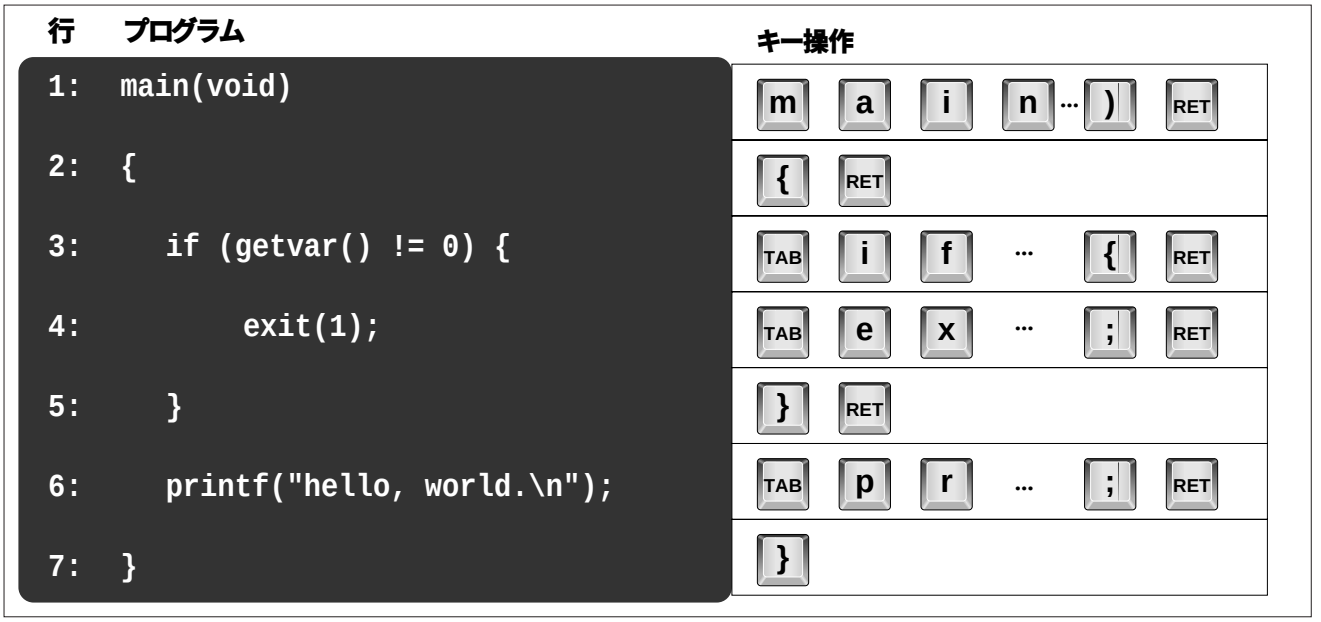
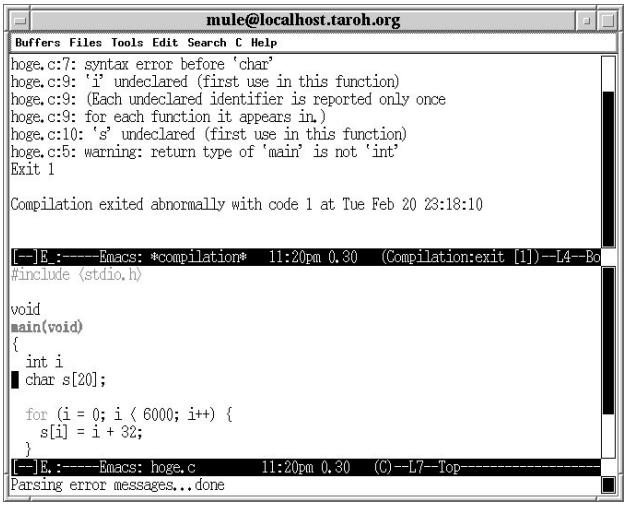



**図2 c-modeの自動インデント**




画面1　コンパイル中のエラー


画面2　作ったプログラムをshellバッファで実行

ます。この印は一時的に表示されているもので、バッファに書き込まれたわけではないのでご安心を。

**画面3**、**画面4**の例では、breakpoint（一時停止位置）の設定やステップ実行などもgud-modeのショートカットで行っていますが、これらは別に知らなくても＊gud-mode＊のバッファでbreak 10、stepなどのコマンドをタイプすることで同じように実行できます。

Perlのソースを編集している場合も、デバッガと連動させることができます。**M-x perldb**とすると、Perlのデバッガであるperldb（これはPerlで書かれているんですねぇー）を起動して、デバッグ対象となるプログラム名やオプションを与えることができます（**画面5**）。

あとはシングルステップ動作（s）や変数の表示など、あまり高度なことでなければgdbでCプログラムをデバッグするのと同様の感覚でできます。詳しくは、DB 〈1〉などと表示されているコマンドラインから、**h h**や**h デバッガコマンド**のようにして表示してみてください。

## アフターケア

さて1年以上にわたる連載になってしまいましたが、最後に連載中説明しきれなかったことや、最近の動向をまとめます。

### ■Emacsのバージョン

現在のGNU Emacsの最新バージョンは20.7で、ftp://ftp.gnu.org/gnu/emacs/からソースコードが入手できます。Emacs19系統は、連載開始当初にすでに開発が止まっていたので、依然19.34.1が最新です。

XEmacs系列は、さらに速いスピードで改良されているようです。この号が出るころには、最新バージョンが21.2に上がっているでしょう（ftp://ftp.jp.xemacs.org/pub/GNU/xemacs/）。21.2からはGTK XEmacsというクールな見栄えのXEmacsも利用可能で、今年の秋にリリース予定のXEmacs 22.0に統合されるようです。

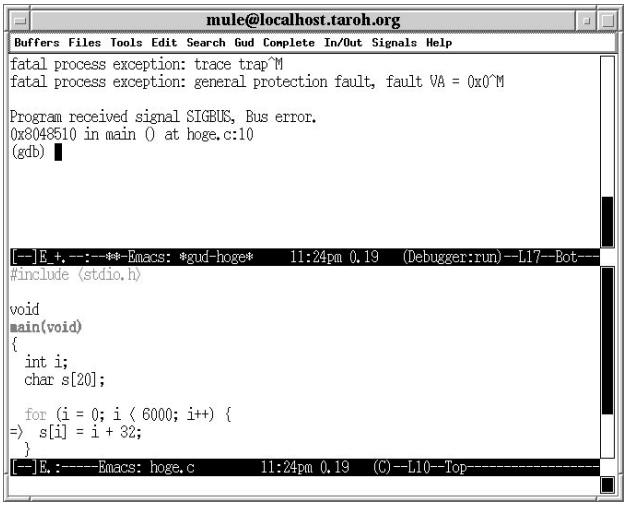
お使いのLinuxディストリビューションに手軽に利用できるパッケージがあるかどうかというのも導入の決め手でしょうが、最新版がほしければ、ソースから導入してみるのも面白いものです。

### ■日本語入力その後（第3回・2000年5月号）

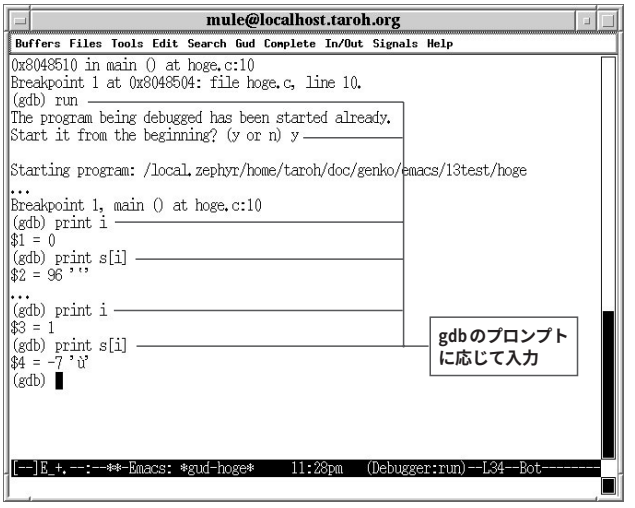
2001年2月号の「番外編」でもちょっと触れたとおり、商用Wnnの最新版Wnn7が5月に正式リリースされるようです（http://www.omronsoft.co.jp/press/wnn7cpt.html）。ベータ版リリースの話題もあったのですが、こちらは原稿執筆時点ではまだアナウンスされていません。

2月号の時点では、パッケージで導入できるフリーの日本語入力IMといえば、Wnn4かCannaが主流でしたが、最近になってフリー版Wnnの後継にあたるFree Wnnがさまざまなパッケージで入手可能になっていますし、ディストリビューションに付属するようになってきました。ただし、Free Wnnそのものは、大きなセキュリティホールに対する対策がなされたほかはあまり大きな変化はないようです。

このほか、A.I.softからWXG for Linux/FreeBSDというのが発表され、現時点ではなんと無料で利用できます（http://www.aisoft.co.jp/japanese/ainews/news/free




画面3　gud-mode




画面4　gudの操作（続き）


画面5　perldbの実行例

wxgl.asp)。WXシリーズは変換の賢さに定評のあるIMで、筆者はまだLinux版を試していませんが、DOS時代にはずいぶんお世話になりました。RPMなどのパッケージも用意されており、インターフェイスとしてCannaのプロトコルを使っているので、cannaserverを停止してwxgserverを起動すれば、簡単にEmacsから利用することができるようです。

### ■メールリーダその後（第4～6回・2000年6～8月号）

Mew（http://www.mew.org/）は最新バージョンが1.94.2です。きびきびと動作するようにしたり、GNUPGをサポートするなどの改良がされています。

Wanderlust （http://www.gohome.org/wl/）の最新バージョンは2.4.1です。LDAPサポート、Emacsバージョン21のサポートなどが加わったのが大きな変更でしょうか。また新着メールをチェックする機能（biff機能）は、とても便利に感じます。GUI系のメーラがあまり進歩しないのに比べて、MewやWanderlustのメーラの進歩の速さと斬新な試みは、さすがオープンソースの世界ならでは、とうなってしまいます。

### ■バイリンガルEmacspeakその後（第11回・2001年2月号）

ソフトウェアの一部として使用している日本語のLinux用音声合成エンジンをユーザーが入手できないことが、普及の障害となっていましたが、その後（株）クリエートシステム開発製の日本語のLinux用音声合成エンジンがVectorプロレジで購入できるようになり、ユーザーに入手の道が開けました。

http://www.argv.org/bep/Linux/ を参考にすれば、比較的容易にEmacsを喋らせることができるようになるはずです。ただしLinux版は、まだバイリンガル化が完全ではなく、カタカナ英語のようになってしまうのが残念です。

### ■多言語その後（第12回・2001年3月号）

「その後」というにはまだ記憶に新しいことですが、ページの都合などで紹介できなかった、Emacsのギリシャ語対応CGreekと、タイ語対応についてすこし補足します。

CGreek（http://www.m17n.org/cgreek/）は、Emacs20で古典ギリシャ語を扱えるようにするためのマクロとフォント集です。これを利用すると、Emacs上でほかの言語と古典ギリシャ語を取り混ぜて記述できるだけではなく、ギリシャ語TeXのソース（目で見るととてもわか

りにくい）を簡単に作成できるようになったり、TLG（Thesaurus Linguae Graecae――ギリシャ語シソーラス、カリフォルニア大学アーヴィン校などのプロジェクトが作成）という膨大な古典ギリシャ文学のデータベースをEmacs上から利用できるようになります。

CGreekは ftp://ftp.m17n.org/pub/cgreek/からダウンロードできます。elispを適当な場所に置いて.emacsを追加する程度でインストールはおしまいですが、多言語ネタではつきもののフォントが3つばかりありますので、先月号の手順を参考にしてXのフォントとして追加します。TLG CD-ROMを利用する場合には、TLCコンバータというのをコンパイルする必要があります（たいして難しくはありません）。

タイ語Emacsのthemacsは、タイ語対応Emacs20.4のバイナリ、フォントとマクロ集です。公式ページは現在見えなくなっているようですが、パッケージそのものはhttp://saikam.nii.ac.jp/ftp/thaisoft/nectec_links/themacs/ などから入手して、インストールできます。

## 目指せEmacsの達人

一部、最初の回のおさらいになりますが、最後にEmacsの達人はココが違う、という点を挙げてみましょう。

### ■困ったときのとコマンド中止とアンドゥ

どんなツールでも、使いはじめはパニックに陥りやすいものです。わけがわからない状況になったら、手遅れになる前に（?）コマンドを中止するC-gを押すこと。また編集誤りをしても、アンドゥ（C-_／C-x u）を使えばいくらでも前の状態に戻せるので慌てないこと。

### ■Emacsの気持ちになろう

Emacsが多機能なことは聞いて知ってはいるけれど、これまでのエディタに比べてEmacsはいささか難しい、と感じる人がいるようです。わけがわからなくなると、ニッチもサッチもいいかなくなって、あげくのはてに破滅的な挙動をするから、ということのようです。

ですが、使いこなしている人からみると、そんなことはありません。むしろ「やさしい」「初心者向け」といわれているエディタよりわかりやすい面もあります。

これはちょうど人付き合いと同じで、Emacsが今どういう動作をしているかをこちらが把握していれば、それに応じた操作ができるということなのではないでしょうか。Emacsには各種のモードがあって、モードが違えば同じ操作をしても挙動が違うことをたびたび述べてきました。しかし、こうした明示的なモードにかぎらず、「今find-file中である」「今C-cが押されていて、次の入力を待っている」というさまざまな細かい状態がEmacsに存在します。

初心者がよく陥るパニックに、find-file中にマウスに手を触れてしまってほかのバッファに移ってしまい、いくらC-gを連打してもfind-fileが取り消せない、ということがあります。

マウスにうっかり触れて、それに気づかなかったということが第一の問題点です。また、ミニバッファの“Find file: ”がC-gで取り消せないというのは、find-fileの最中にほかのバッファに移ってもかまわないけど、最後に必ずミニバッファに戻って操作の続き（か中止）をしないといけない、という「常識」が身に付けばどうということはありません。これは、Emacsがどんな状態にあるときにどういう挙動をするか、という観察を積み重ねて得られるものです。「Emacsの気持ちを知る気がない」と、観察不足でいつまでたってもとんちんかんな操作がなくならないのではないでしょうか。

### ■infoを活用せよ

ときどき、どうしてそんなにEmacsのことを知っているの？　と聞かれることがあります。でも実は、知識として持っている量はたいしたことがないのです。知識を増やすなら知識自体よりも知識へのポインタを増やせ、とはよく言われることですが、どうせ暗記するならマニュアルの中身を暗記するより、こないだ困ったときにどこを見たかという解決方法を憶えるのが得策でしょう。

Emacsのマニュアルであるinfoは全般によくできているので、活用しない手はありません。**C-H i**とタイプしてみましょう。もちろん、各種ツールや標準外のマクロのinfoは書き手もさまざまなので、出来不出来はあるでしょう。しかしEmacs標準のマクロのinfoは、辞書のようなたんなるリファレンスマニュアルではなく、比較的よく使う実例を挙げて書いてあるので、非常にプラクティカルです。また、ローカルマクロでもこの流儀を受け継いでいるものが少なくありません。

### ■メニューバーの活用

X上のEmacsであれば、モードに特有の項目がメニューバーに追加されます。筆者のようなショートカットキー派で、ほとんどメニューバーを操作することがない人でも、

使いこなしの大きなヒントになるでしょう。なにしろ、メニューバーに紹介されている機能は、設計者が「よく使うであろう」「主要な」機能と考えているものでしょうから。また、メニューバーから設定されているショートカット操作を知ることもできます。もっとも最近のEmacsは、おせっかいにも「次回からショートカットを使え」とミニバッファに表示してくれるので、見逃すことも少なくなってきました。

たとえば、今回紹介しなかったmakefile-modeがどんなものか探ってみたければ、Makefileというファイルをfind-fileしてみてください。メニューバーにMakefileという項目が新設されます。そこを探っていくと、**M-n**で次の依存関係に飛べることが発見できるわけです。

### ■helpを活用せよ

次に、「この機能がインストールされている（はずな）のにinfoで見つからない〜」というときのことです。ちょっと邪道かもしれませんが、キーバインド、elispの関数、変数をあてずっぽうの名称でサーチすることで見つかることがあります。

たとえば、今回取り上げたgudに関するキー操作にはどういうものがあるかな、というときは、gud-modeに入っておいて**C-H b**とすると、＊Help＊バッファには考えられるかぎりのキー操作が表示されるので、**C-o**でそのバッファに移り、**C-s gud**などとすれば、gud関連で可能な操作がリストアップできます。

gud-finish（C-c C-f）って何かなというときには、**C-H f gud-finish**というコマンドを実行すれば、簡単な説明が表示されます。**M-x hogemoge**という操作の“hogemoge”の部分はelisp関数になっていますから、これはコマンド名は知っているけど動作がわからないというときにも有効です。よく知っているつもりでいる関数でも、引数を付けると別の挙動をすることを発見したりします。

ほかに、**C-H v**で関係ありそうな変数の説明を表示させてみると、役立つこともあります。補完がきき、ある関数の挙動に関連ある変数名はその名前で始まっていることが多いので、たとえば**C-H vgud**〈SPC〉で説明を表示させてみることができます（説明がない場合もあります）。

### ■elispマクロ書きは他人のパクりから

Emacsは、どんなに複雑な挙動をするコマンドでも、誰かがelispで書いているので、すぐにそこにあるソースが見られる（/usr/local/share/emacs/バージョン/…/＊.elなど）という利点があります。したがって、わけはわからずともelispのソースを眺めていると、隠れた（って見つけられないだけですが（泣））コマンドやキー操作を発見して思わずうれしくなることがあります。

関数名や変数名がわかっているけど、helpに説明がないとき（あるいは意味不明）、どういう定義なのかなというとき、あるいはこんな動作があるはずなんだけど……というときも、上記のディレクトリを適当な英単語をキーワードにしてgrepしてみると、いろいろなヒントが得られることがあります。

さらにelispが多少でもわかってくると、.emacsを書くときに参考となる書き方に出会えるのも、大きな魅力です。

### ■困っている人はほかにもいるはず〜Webの活用

知りたい情報があったら検索エンジン、というのはもうすっかり定着している使い方かもしれません。フリー・オープンのコミュニティでは、オンラインの情報はペーパーメディアの雑誌や書籍より価値がある場合が多いのです（と雑誌で力説しても説得力がないですが（笑））。Emacsにかぎらず、フリー・オープンのソフトとして集団で開発しているApacheやPHPなどは、自前のドメインを持っていて有用な情報がちりばめられています。紙のマニュアル本が、オンラインにいられないときの、非常用の貧弱な環境に見えることもあるほどです。

また、Webページに困ったときの経験談が書いてあったり、メーリングリストでやりとりされた質問や意見が検索できたり、開発者がメーリングリストやWebで情報交換・情報公開をしている場合などが多いものです。あなたも困ったことがあって人に助けられたら、次はそれを公開することで他人のお役に立てる番……かもしれません。

## 最後に

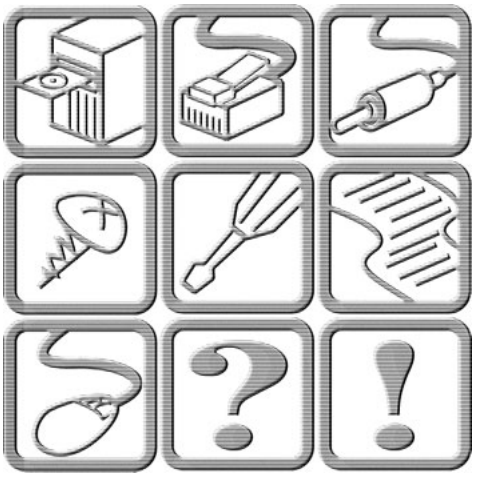
基本操作の紹介もそこそこに、つれづれなるまま面白いと思ったEmacsの使い方を連載してきましたが、いかがでしたか？　筆者のWebページ（http://www.taroh.org/Docs/linuxmag/）でも可能なかぎり落ち穂拾いを続けていきますので、おやっ？と思ったときには覗いてみてください。

では、またお会いする日まで。

◆

Happy hacking！！

Try & [trái] Try [trái]



# Devfs（デバイスファイルシステム）－2

文：政久忠由
*Text : Tadayoshi Masahisa*

首筋に嫌な汗をかいている。そして首と頭の付け根の後ろ側に痺れたような奇妙な違和感も同時に覚える。

これは僕が何かしら問題を抱えたまま睡眠をとった時の目覚めにいつも感じることである。怖い夢を見た時とは、ちょっと異なり、緊張は局所的だ。全身にわたる緊張や発汗はなく、ちょうど首筋だけが熱くなっている。また頭、つまり脳は、極めてクリアな状態で、けだるさはない。むしろ聡明な状態、そう台風一過のような状態だ。この時僕は、首は脳のヒートシンクなんだよなぁとつくづく思うのである。脳は血液によって冷やすに限る。発熱の際、どうして氷水枕とかで頭を直接冷やすことが一般化しているのか未だに理解に苦しむ。まぁ僕は、医学的な知識は皆無に近いので、真偽の程は定かではないけど、頚動脈／静脈を冷やすのが最も効率的だよなぁ。あっ、でも効果があり過ぎて、ショック状態になってしまう惧れがあるのかもしれない……

## ジャンクな一日

さて、こんな目覚めを迎えることになった理由は、僕のLinuxマシンにある。

前日（といっても僕の生活リズムの前日だから、単に寝る前ということなのだけれども）、僕は、常用しているAthlon搭載のLinuxマシンを再起動した。さまざまな変更を整理するためだ。カーネルやglibcを始め、本当にシステムのキーコンポーネントをすべて新しく作り直し導入した状態だった。

こういったことは、特に珍しいことでもなく、ベータコードが主要部分には含まれていないようだったので、格別注意を払うことなく、システムの再起動を行った。しかし、そこには苦難の道が待ち構えていた。

まず、システムにログインできないのである。ネットワークを介したリモート、さらにローカルコンソールの両方でログインできない。加えて、システムの動作が非常に遅いということも判明した。

最悪である。原因を探るにはあまりにも条件が悪い。再起動前と後では、システムの変更箇所が多岐にわたり過ぎる。先のソフトウェアコンポーネントに加え、実はシステムBIOSのアップグレードも行っていたのだった。

原因を探るにしても、とりあえずログインできなくては手の施しようがない。そこでとりあえず、liloに登録してあるいくつかのカーネルバージョンを切り替えて見ることにした。

でも結果的には、ログインすることはできなかった。ログイン関連のライブラリに問題があるようだ。glibcを2.2.1から2.2.2にしたことを真っ先に疑ったが、何にしても、とにかくファイルシステムにアクセスできる状況にならなければならない。

幸いこのシステムのハードディスクは、数週間前に移行したもので、元のディスクにはそれなりに安定動作していた時のイメージのすべてが完全な形で残っている。これに

アクセスできれば、目的のライブラリファイルをオーバーライドして手っ取り早く修復できそうだ。

問題はどうアクセスするかである。元のディスクに収められたシステムは、IDEインターフェイスのプライマリーマスタに接続して利用する設定がなされているので、そのように接続して、修復したいディスクを適当なインターフェイスに移動して修復する手もあるけれども、何となく面倒な気がした。リペア用のフロッピーディスクで完結したLinuxシステムも非常用にと用意していたので、それを利用しようかとも思ったのだけれど、最初に試すべき手を思い出した。INITに1をわたす、つまりシングルユーザーモードでの起動である。さらにINITでの初期化を行わず、/bin/shを直接実行する手段もある。

落ち着いて最善の方法を取ろうと思うのだが、結構頭に血が上っているようだ。このところ目立ったトラブルに見舞われていなかったので、かなり舞い上がっている。でも、ちょうどシステムの引越しをしたところで、元のディスクを潰していなかったのはラッキーであった。我が家で最も遅い部類のハードディスクだったので、使いまわしをする気が起きなかったのが幸いした。最悪ロールバックはこの状態で押さえられる。もしバックアップスナップショットがなく、再インストールが必要な状況だったとしたら…と思うと、背筋が寒いのだけれど（各種設定ってハッキリ言って一回決まればそれっきりってことが多いから、忘れちゃうし、元通りにするのってとっても面倒なんだよねぇ）、しかしまぁすぐに割り切っていたかもしれない。ディスクが壊れてデータが吹っ飛んだわけではないからね。

さて問題のシステムは、liloの各ラベルに引き続き、INIT番号1を指定することで、シングルユーザーモードでログインすることができた。ただ、新しいカーネル2.4.1では、システムが遅いという現象は引きずっている。そこで作業は、古い2.4.0で行うことにした。

その状態でいろいろチェックしたところ、glibcの問題ではないようだ。これまで完全に忘れ去っていたが、システム全体のファイルの更新日チェックから、PAMモジュールを新しいのにしていることが判明した。確かに作り変えたかもしれない。PAMの設定を見直そうかとも思ったのだけれど、ローカルのコンソールにログインしての作業は、1000円程度で購入した堕落したキーボードがあまりにも苦痛なので回避することにして、元のハードディスクから、PAM v0.74系の/lib/libpam＊と/lib/security/のモジュールを上書きし、ldconfigでバインド状況を更新した。

ビンゴ！ PAMモジュールを修復したことでログインできないという状況は克服できた（何が気に入らなかったのだろう？ 不明）。

しかし、依然としてカーネル2.4.1では、システムが遅いという現象が生じている。ただ、カーネル2.4.0では、この現象は生じていないので、他のライブラリの関与については、プライオリティを下げることができる。

はてさて、システムが遅いという現象は、カーネル2.4.0とカーネル2.4.1の違いに絞られたわけなんだけど、じゃぁ何が悪さをしている？ これが難題であった。

一応安定バージョンのカーネルである。実際、別のCerelonマシンでは同じバージョンのカーネルで何の問題もなく快適に動作する。けれども、その別マシンで問題なく動作するカーネルをフロッピーに導入し、当該のAthlonマシンで実行するとやっぱりダメなのだ。カーネル構成をあれこれしたが、一向に改善されない。何か祟られているような気さえした。

数時間の格闘の後、この状態を放置して僕は、睡眠に突入した。問題解決の糸口を掴めないまま、眠りにつくことになったのである。本当に散々なジャンクな一日だと思った。

## 問題はACPIのIdleモードとの相性

眠りというのは、僕にとってはよい結果を生むことが多い。学生の頃から試験の前日は、それなりに情報を詰め込むがきちんと睡眠をとることに重点を置いていた。そう、情報は整理されないと効率が悪いのである。

今回は目覚めた時、以前当該マシンのACPIでトラブったことを思い出した。信心深い人なら、神などからの啓示と思ってしまうかもしれないけれど、僕は神も仏もシャーマンも信仰していないので、単に記憶が整理されたとしか考えていない。

パワーマネージメントのACPIに関しては、前日、特に気にもかけていなかったので、カーネル構成でいじることをしなかった。不覚。

とりあえず、カーネル2.4.0と2.4.1のパッチファイルでACPI関連の変更箇所をチェックしてみる。そうすると、CPUのパワーマネージメントモードのenter／exitの判定などが変更されている。早速、ACPIを無効にしてカーネルを作成し、実行してみる。システムが遅い現象はあっさりと回避された。どうやら当該システムのACPI BIOSのC2／C3モードのレーテンシ既定値がカーネル2.4.1のACPIコードとは相性が悪いようだ。たぶんC3モードの判定から抜け出せないままの状態が続いていたらしい。原因

は本当に些細なことだった。/proc/sys/acpiでそれぞれの設定値を確認しても特に問題があるようには思えないが、はてさて何でや？ これはそのうち、設定をいろいろと変えてテストしてみることにしたい。

ちなみにLinuxカーネルは、現時点（2月中旬）で最新は2.4.2。このバージョンでもこの問題の根本的な対策はなされていないのだけれど、問題を回避するためのカーネルオプションが2つ用意されている。

```
acpi＝no-idle
acpi＝off
```

no-idleを指定すると、acpi_idleルーチンの実行が抑制される。またoffを指定するとACPI全体の処理を抑制することができる。

これらのオプションは、カーネルを指定する際の引数として指定するのだが、liloでは、/etc/lilo.confで、append＝“acpi＝no-idle”という行を追加することで設定できるので覚えておこう。

あと一応、問題のマシンと問題ないマシンのACPI設定は次のとおり。

```
ACPI: Core Subsystem version [20010208]
●問題の出るマシン
ACPI: System firmware supports: C2 C3
ACPI: plvl2lat=90 plvl3lat=900
ACPI: C2 enter=1288 C2 exit=322
ACPI: C3 enter=38653 C3 exit=3221
ACPI: Using ACPI idle
ACPI: System firmware supports: S0 S1 S4 S5
$ head /proc/sys/acpi/c*
==> c1_count <==                0x00000648
==> c2_count <==                0x00000f6a
==> c2_enter_latency <==        0x00000508
==> c2_exit_latency <==         0x00000142
==> c3_count <==                0x00000000
==> c3_enter_latency <==        0x000096fd
==> c3_exit_latency <==         0x00000c95
●何の問題もないマシン
ACPI: System firmware supports: C2
ACPI: plvl2lat=1 plvl3lat=1001
ACPI: C2 enter=14 C2 exit=3
ACPI: C3 enter=-1 C3 exit=-1
ACPI: Using ACPI idle
ACPI: System firmware supports: S0 S3 S4 S5
$ head /proc/sys/acpi/c*
==> c1_count <==                0x00000001
==> c2_count <==                0x000046fb
==> c2_enter_latency <==        0x0000000e
==> c2_exit_latency <==         0x00000003
==> c3_count <==                0x00000000
==> c3_enter_latency <==        0xffffffff
==> c3_exit_latency <==         0xffffffff
```

ACPIのidle機能が有効になることで遅くなるという問題の生じるマシンでは、C3モード（C2より深いパワーマネージメント）をサポートしているが、このモードに突入することで問題が生じているわけではないことは、/proc/sys/acpi/c3_countの値をチェックすることで確認できる。カウンタは0、つまり一度もこのモードに移行してはいない。

また、plvl2latやC2 enter／exitの設定値がかなり異なっているように表示されているが、これはスケールの違いで、C2 enter／exitをplvl2latで割ると前者でも約14と3になることが分かると思う。内部的にはこれらの値とACPIのタイマが比較されるので特にこれが問題というわけでもないと思う。実際、前者のC2 enter／exitの値を大きくしたり、小さくしたりしても問題は解決されなかったのである。

というわけで僕は現状ではちょっとお手上げ状態。結局acpi＝no-idleを指定して使用している。超下らない裏技としては、CPUに休み時間を与えないように無限ループプログラムを動かしておくことでも、それなりに実りというか、意味のある対策にはなる。単にwhile(1);を含むプログラムを実行するだけなのけれど、CPUがパワーマネージメント状態に移行するのを防ぐのだ。ただCPUパワーは多少浪費されることになる。最近のCPUは馬鹿ではないのでそれほど心配する必要はないと思うのだけれど、場合によってはCPUがひどく発熱することがある。少し気にはとめておきたい。

解決策ではないのだけれど、前者は、ACPIのidleが有効になった時点で、システムタイマの時の刻みも遅くなっている。だからそのシステムのすべての世界が遅くなってしまっている。外界の世界の時間流から隔離されてしまっている。まぁあくまで相対的になんだけど、それが問題なのだ。CPUのパワーマネージメントでCPUのタイマカウ

ンタは当然遅れるので、それをACPIのタイマか適当なクロックタイマで補正するハズなのだけれど、何だかなぁ。ACPIの場合、同じC2といっても、どの部分のパワーマネージメントを行うかは実装によってまちまちだからという問題もある。一方では、CPUのクロックカウンタがパワーマネージメントの対象で、もう一方ではそうではない実装という可能性だってあるし、単にAthlon CPUの癖かもしれないし、チップセットの気まぐれということだって考えられる。ただサンプルがたった2ケースだけなので、傾向もへったくれもなかったりする。とにかくACPIのファームウェア自体、デバイスドライバ、ハードウェア、どこに問題があってもおかしくないから、本当に困りものなのである。

でまぁ、そんなこんなで当面のトラブルは解決した（ことにした）のだけれど、前者のマシンは他にも得体の知れない問題が潜んでいることが判明してしまった。かなり怪しい挙動だ。割り込みやスケジューリングレベルの処理に問題があるような動作をするのである。これに関しては問題が根深そうなので、今回手をつけるのは止めておくことにしたい。ん〜、開発バージョンより、カーネル2.4.0以降で数多くトラブルってのは、ついてないよなぁ。しかも一方のシステムでは何の問題もなくて、もう一方のシステムだけで踏んだり蹴ったりなので、明らかにこの環境に起因するトラブルと言わざるを得ない。このシステム、ごく普通だと思うのだけれども。

あ〜ぁ、しばらく嫌な汗をかき痺れを感じることになりそうである。目覚めると7人の小人が直してくれているってことはないよなぁ、きっと。

## Devfsのしくみ

さて、嫌なことはこれぐらいにして今回の本題に移ろう。

Linuxに取り入れられているDevfs（デバイスファイルシステム）の実装は、メモリ上に仮想のファイルシステムを提供するだけの器（うつわ）としての存在であることは前回説明したとおりだ。

このネームスペース（名前空間）に、実際にシステムが稼動するのに必要となる、というか、システムがアプリケーションプロセスに対して提供することができるデバイスのみ存在するというのが、デバイスファイルシステムの特徴なんだけど、これは、各デバイスドライバレベルで、デバイスファイルシステムにそれぞれが提供するデバイスファイル（ノード）を追加するルーチンを装備することで実現している。

つまりデバイスファイルシステムを有効にした際に、そこに登録されるデバイスノード群は、基本的には、各デバイスドライバが登録するのである。デバイスファイルシステムは、単に空間を提供しているだけなのだ。RAMからメモリを切り出して、デバイスノードを格納できる適当なファイルシステムでフォーマットして、/devにマウントする。ベースの動作としてはこれだけなので、既存の機能（RAMDISKとExt2fsなど）を組み合わせて実現することもできなくはないのだけれど、デバイスファイルシステムはあくまでデバイスノードを格納するためのものなので、それに特化したメモリ効率のよいファイル構造体の配列（これがデバイスファイルシステムなんだけどね）で管理されるようになっている。

実際、/devにデバイスノード以外のファイルを作成しようとしてみると、その意味がよくわかると思う。一般的なExt2fs上の/devでは、当然通常のファイルも作成可能だ。/devだからといって特別なことはなく、ノードとブロックユニットで管理されている。あくまでデバイスノードの作成（格納）にも対応しているファイルシステムなのである。

しかし、デバイスファイルシステムをマウントした/devには、mknodコマンドなどで、デバイスノードは作成できるものの、通常のファイルは“cannot create regular file”、“Permission denied”といったメッセージが表示され、作成することはできない。内容の存在しないサイズ0のファイルであってもダメだ。つまり、デバイスファイルシステムは、ファイルのサイズに関わらず、通常のファイルは作成できない、特殊なファイルシステムなのだ。この特定のオブジェクトのみの格納に特化しているからこそ、いわゆるファイルとしての各デバイスノードオブジェクトや格納構造の維持に必要となるメモリをコンパクトなものにでき、結果、デバイスファイルシステムがシステム消費メモリに占める割合も非常に少なくて済むのである。実際、デバイスファイルシステム上での、コアとなるデバイスノードの構造体は、32バイト＋デバイス名×バイト分だ。Ext2fsだとInodeに加えてなんやかやのさまざまな構造体の複合体で表されるので、数倍のサイズが必要となってしまう。

## Devfsにおけるデバイスノードの登録

さて、つまるところデバイスファイルシステムは、デバ

イスノードを格納するためのさらのファイルシステムを作り出し、/devに自動的（カーネル構成やカーネルパラメータ次第）に、もしくは手動でマウントすることができる。しかし、そこに登録されるデバイスノードは、各デバイスドライバがそれぞれの初期化時に作成する必要がある。

そのため、このデバイスファイルシステムに対応していないデバイスドライバだと、自動的にデバイスノードが登録されることはない。必要に応じて、誰かがそのノードを作成しない限り、メモリ上でデバイスドライバが初期化され有効になったとしても、使いものにならない、まぁ使えないのである。この問題については後にまわすとして、各デバイスドライバが初期化の際、デバイスノードをデバイスファイルシステムに登録することについて、少し説明しておきたいと思う。

デバイスファイルシステムには、その操作のための関数を用意している。どのような関数があるかは、/usr/include/linux/devfs_fs_kernel.hファイルを見て欲しい。

登録を行うdevfs_register_＊系と登録を抹消するdevfs_unregister_＊、そしていくつかの操作系の関数が用意されていることがわかる。

実際にフロッピーディスクとRAMディスクのドライバを見てみることにしよう。ファイルは、/usr/src/linux/drivers/block/のfloppy.cとrd.cだ。

floppy.cでは、ドライバの初期化を行うfloppy_init()において、devfs_mk_dirでfloppyディレクトリを作成し、devfs_register_blkdevでデバイスノードの登録を行っている。これらの作業によって、/dev/floppyにノード名、0、0u1040、0u1440…といったフロッピーディスクデバイスノードが作成される。

RAMディスクドライバのrd.cでも同様に、初期化ルーチンのrd_init()において、devfs_mk_dirでrdディレクトリを作成し、devfs_register_seriesでノードの作成を行っている。この結果として、/dev/rdが作成され、ノード0～15が登録される。

またそれぞれdevfs_unregisterの実装も行われているので、ドライバが抹消される際にデバイスノードも抹消されるようになっている。

まず最初に/dev/の状態を確認しておき、それから/sbin/modprobe floppy（rd）でドライバを登録し、変化を見るとよいだろう。また/sbin/rmmod floppy（rd）でドライバをアンロードした時の変化も観察してみよう。

これらを見て何か感じるものがあった人もいると思う。

たとえば、デバイスファイルシステムでなくても、ドライバの初期化時に/dev/にデバイスノードを作成できるのではないかと。また、オンデマンドロードモジュールはどうなるのかと。

結果から言うと、デバイスファイルシステムでなくても、ドライバの初期化時に/dev/にデバイスノードを作成できる。実際、RAMディスクには、通常のファイルシステムのルーチンregister_blkdevで/dev/ramdiskというノードを作成するコードが含まれている（抹消のためのunregister_blkdevもある）。しかし、一般にはこれらの処理は実装されていないのが実情だ。なぜ？　僕にはわからない。個人的にはドライバレベルで処理してくれたらとっても便利だと思うんだけどねぇ。

必要に応じてカーネルモジュールデーモンがドライバモジュールをロードするオンデマンドロードモジュール機能に関しては、デバイスノードをドライバの初期化時に登録するデバイスファイルシステムとは相容れないものがあり、ハッキリ言って利用できない。

考えて見ればわかるが、オンデマンドロードモジュール機能は、/dev/のデバイスノードをアプリケーションプログラムなどがハンドルしたのを仮想ファイルシステムレイヤレベルで検出し、その要求するイベントを処理するためのデバイスドライバを検索して、自動的にドライバモジュールをロードして処理を続行するよう実装されている。

つまりオンデマンドロードモジュール機能は、/devにデバイスノードがなければ意味がないのだ。どのドライバモジュールが必要なのかの情報が、一切伝わらないのである。

たとえば、僕はフロッピーディスクやCD-ROMドライブのドライバは、めったに利用しないのでモジュール化しておき、先の機能を利用することで、ドライバの明示的なロードは行わず、/mnt/floppy（cdrom）のマウント操作だけで済ませていた。でもデバイスファイルシステムを利用するようになって、ドライバモジュールのロードを明示的に手動で行う必要が出てきた。

まさに一長一短である。

僕は、今のところデバイスファイルシステムを選択している。この先気が変わるかもしれないけれども。

一応、必要なドライバをモジュール化しないで、カーネルに直接組み込んでおくことでも対処は可能だ。基本的には、さほど利用しないドライバルーチンをカーネル本体に組み込んでしまうとメモリの無駄なので、なるべく避けたいところなんだけど、最近は本当にメモリが少なくて困っている人は希だと思う。だから、たまにでも使うかもしれ

ないドライバをカーネル本体に組み込んでしまっても、問題はほとんどない。1Mバイト以上浪費することはまずない。せいぜい数十から数百Kバイトの浪費だろう。

ちなみに取り違えないようにして欲しいことなんだけど、デバイスファイルシステムでドライバモジュールすべてが問題になるわけではない。問題なのは、デバイスノード越しに直接利用されるドライバだけだ。ファイルシステムなどの中間層のドライバは関係ない。実は、それほど大きな問題はなかったりする。僕はフロッピーとCD-ROMでちょっと困っただけ、かな。

## devfsのカーネルオプション

僕はデバイスファイルシステムをブート時に自動的に/devにマウントする設定で動作させているが、カーネルオプションでこれを操作できるようになっている。mount、unmount、only、show、加えてデバッグ用のdallなどのオプションもあるが、基本は最初の4つである。mountは自動マウントする、unmountはしない、onlyはデバイスファイルシステムに既定のデバイスノードを作成しない、showはノード抹消の際のメッセージを表示するためのオプションだ。ACPIと同様にdevfs=mountといった感じで指定する。

## devfsd

devfsdは、従来環境、つまり従来のデバイスノード名との互換性などを維持管理するためのデーモンサービスだ。必須のサービスではない（僕は使っていない）。このプログラムは、デバイスファイルシステムの作者のホームページhttp://www.atnf.csiro.au/~rgooch/linux/から入手できる。現時点（2月中旬）では、devfsd-v1.3.11がリリースされている。

devfsdをコンパイルして（make一発でOK）、make installを実行すれば、/sbinにdevfsdプログラム、/etcに設定ファイルのdevfsd.conf、あと必要に応じてmodules.devfsがコピー、作成される。設定ファイルは問題が生じない限り変更する必要はないだろう。

早速、devfsdを実行してみることにしよう。自動的にバックグラウンドに移行するので、プログラムの引数に管理するディレクトリ（通常/dev）を指定するだけでよい。いくつかオプションが用意されているが、デバッグメッセージのレベルを調整するためのものなので、man devfsdに目を通しておくだけでいいだろう。

```
# /sbin/devfsd /dev
Started device management daemon for /dev
```

さて、このデーモンによって何が提供されるのだろうか？ /devを見てみよう。

hda＊、pty＊、tty＊など従来のデバイスノード名のシンボリックリンクがデバイスファイルシステムで提供されるノード名に対して作成されているのがわかると思う。

```
hda -> ide/host0/bus0/target0/lun0/disc
ptyp0 -> pty/m0
```

つまり、このデーモンプログラムを初期化スクリプトの最初に実行するように設定すれば、既存の設定やプログラムのトラブルをある程度、回避できるというわけだ。

たとえば、X端末プログラムなどで仮想回線を構築する際、glibcのgetpt()／ptsname()を利用していれば、デバイスファイルシステムが提供する/dev/pts/＊だけで問題はないのだが、プログラムによっては、別の方法で処理している場合がある。xtermやktermのソースコードを見ると機種や環境によって実にさまざまな仮想回線の方法で処理していることに驚くと思う。

またdevfsdは/devの監視を行っていて、floppyドライバをロードするなどして、新しくデバイスノードが登録された場合など、その状況に応じて新規にシンボリックリンクを作成するようになっている。たとえばfloppyの場合は、/dev/fd0 -> floppy/0といった感じだ。

僕はサウンド機能をあまり利用していないから気にならなかったのだが、/dev/dspがデバイスファイルシステムでは、/dev/sound/dspだったりするので、できることならdevfsdを動作させておいたほうがいいかもしれない（設定ファイルの修正でどうにかなる場合はいいけど、デバイスノード名がハードコーディングされていると修正が面倒だからね）。

ちなみにRedHat 7.0のrc.sysinitには、initlogに続いて2番目にdevfsdの実行スクリプトが記述されているので、特に何もする必要はない。/dev/.devfsd（デバイスファイルシステムが有効になっているという証）と/sbin/devfsdが存在すれば、/sbin/devfsd /devが実行されるようになっている。

まぁとにかくやってみないことには、ね。ではでは。

# Linux日記

## 第19回　メール配送（6）

**sendmailの設定ファイルsendmail.cfは、その書式が難解なことで有名です。しかし、難しいからといって中を見ることもしないなんてもったいない。今回からはsendmail.cfの内容を解説します。**

文：榊 正憲
*Text : Masanori Sakaki*

*illustration ; Aki*

雑誌の原稿というのは、当然のことながら、発売日よりかなり前に書いている。また、雑誌の何月号という数字は、実発売日からおおよそ1カ月先行している。このため、この2001年4月号の原稿は、筆者サイドから見ると、2001年になってから最初に書いている原稿なのである。

さて、本当に21世紀になってしまった。子供のころに読んだ少年誌の巻頭特集にあった21世紀の予想図では、都会には70年万国博のパビリオンのようなビルが立ち並び、建物の間は透明チューブで結ばれ、エアカーが走り回ることになっていたのだが、このような状況には、残念ながらならなかったようだ。月面に基地もないし、衛星軌道に浮かぶドーナツ型の宇宙ステーションもないし、それに連絡するPAN AMのシャトルも運行していない。そもそも、PAN AM自体がなくなってしまったのだから仕方がない。HAL-9000はまだ実用化されていないようだ。

それでも、東京の湾岸線あたりを地上用自動車で走れば、かなり未来的風景が広がり（少なくとも「ソラリス」よりは未来的だろう）、2足歩行ロボットも登場した。DOCOMOのCMでやっていたように、携帯型通信機器は当時想像していた以上に活用されている。少なくとも、宇宙飛行士、正義の味方や悪の手先、あるいは金持ちの道楽の人命救助隊員ではなく、女子高生がこういった機器を縦横無尽に使っているというSFを筆者は読んだり見たりした記憶はない。20世紀の最後4分の1は、宇宙開発などの重厚長大プロジェクトはSFよりも大幅におくれてしまったが、情報系はSFの世界が予想した程度には進んだようだ（人工知能分野に頑張ってもらいたいところだ）。

まぁ、21世紀になったからこのようなことを思うのだが、よく考えてみれば、宇宙人の侵略に対抗するSHADOという組織がイギリスの映画会社の地下深くに作られていたのも、毎週日本が怪獣や宇宙人に襲われて、富士の裾野の山が割れて地球防衛軍極東基地のウルトラホークが発進していたのも1980年代の話なのだ。また別の大事件で、妖星ゴラスが地球との衝突コースに乗ったため、南極に巨大なロケットエンジンを建設して地球の軌道をずらすなんてことをやってのけたのも確か80年代前半である。エアカーやHALがないのは残念だが、怪獣の襲来やら太陽系規模の天変地異がなかったことに感謝すべきなのかもしれない。

### ■2001年問題？

1年前の今ごろは、2000年問題でワイワイいっていた（あるいは、思いのほか問題がでなくて喜んでいた）。家のコンピュータの1台は、リアルタイムクロックに問題があることがわかり、現役からの引退を余儀なくされた。今年は何事もなく過ぎるはずだったのだが、1月に入って数日してから、ちょっとした問題が起こった。コンピュータの日付がおかしいのである。曜日と日付が合っていない。

どうしたことかといろいろ調べてみ

たら、何のことはない、正しいと信じて自分が見ていた紙のカレンダーが去年の1月のものだったのである。12月の次のページが1月のものだったので、そのまま見ていたのだが、実は先頭に戻っていたのだ。間違えていたのはコンピュータではなく、自分の認識だったわけだ。なんとも情けない2001年問題であった。

### ■ウイルスメール

ある日、おかしなメールが来た。Subject:もFrom:もTo:もない。telnetでSMTPをしゃべるという記事が雑誌に載った直後だったので、読者のいたずらかなとも思ったのだが、中身はMIMEエンコードされたWindowsのEXEファイルだけである。調べてみたら、TROJなんとかというウイルスメールであった（筆者の周辺には、MicrosoftのOutlookを使っているユーザーがまるでいなかったので、今までウイルスメールは来たことがなかったのである）。

ウイルス関連ソフトのサイトの情報によると、このプログラムを実行すると、Windowsのソケットライブラリが書き換えられ、メールの送受信（SMTPとPOPのセッションだろう）がモニタされるようになる。そして、そのセッションで使われた送信元、宛て先のアドレスに自身のコピーを送るらしい。メールシステムのレベルではなく、ネットワークドライバのレベルで動作するため、Outlookを使っていなくても伝染していくというシステムだからたちが悪い。

このウイルスプログラムは一応多国語対応のようで、言語環境に応じて何カ国語かでSubject:や本文を用意するようなのだが、その対応言語に日本語が入っていなかったようだ。そのため、Subject:や本文のないメールになってしまうのだろう。sendmailの解説記事を連載していることでもあり、参考までに、うちに来たメールのヘッダを**リスト1**に示しておこう（内容は一部書き換えてある）。

さて、ここまでsendmailの連載を関心を持って読んできた読者なら、このヘッダからいろいろなことがわかるだろう。

最初のReceived:ヘッダは、あるプロバイダのメールサーバ（mf003）から我が家のメールサーバへのSMTP中継記録である。2番目は、最初のコンピュータからプロバイダのメールサーバにSMTPで中継した記録だが、実に、最初の発信元コンピュータの名前がoemcomputerである。これは、Windowsをプリインストールしてあるマシンのデフォルトの名前のはずだ。ドメイン名もへったくれもない。

Message-Id:とDate:を見ると、これらのヘッダは最初からあったのではなく、中継したプロバイダのメールサーバが生成したものであることがわかる。なので、最初にWindowsマシンから送られた時点では、ヘッダは何もなかったのであろう（Windowsマシンとメールサーバの時間が秒まで合っているとも思えない）。これらのヘッダは、なければMTAが生成するという性質のものなので、プロバイダの中継サーバが付加したのである。

これらのヘッダ以外には、WindowsのEXEファイルをエンコードするためのものしかなかった。Bcc:を使えばTo:やCc:のないメールを送ることができるが（宛て先はエンベロープ情報で指定されるということを思い出そう）、From:がないというのもすごい。メールシステムのセキュリティについては、この連載で数カ月後に説明するつもりだが、From:ヘッダはセキュリティ対策のチェック項目である。きちんとチェックしていれば、プロバイダのSMTPサーバはうちのサーバにリレーしなかったのではないかと思うのだが。

さて、セキュリティの話は暖かい季節になってから行うとして、自分の携帯端末から世界中と自由に通信できるという未来図の一部を担っているsendmailの話、それもsendmail.cfの話の続きである。

### ■マクロとクラス

マクロを使えば、sendmai.cf中で頻繁に参照される文字列（ホスト名やドメイン名、メールアドレスなど）をマクロとして定義し、さまざまな部分で参照することができる。これにより、

**▼リスト1　ウイルスメールのヘッダ**

```
Received: from mf003.XXXXX.ne.jp (mf003.XXXXX.ne.jp [210.XX.XX.XX])
        by mail.takobeya.com (8.9.3/3.7W-2.8compat) with ESMTP id SAA05598
        for <masa@takobeya.com>; Wed, 31 Jan 2001 18:06:10 +0900 (JST)
Received: from oemcomputer
        by mf003.XXXXX.ne.jp (8.9.3+3.2W/3.7W-10/13/99) with SMTP id SAA03637
        for <masa@takobeya.com>; Wed, 31 Jan 2001 18:06:28 +0900
Date: Wed, 31 Jan 2001 18:06:28 +0900
Message-Id: <200101310906.SAA03637@mf003.XXXXX.ne.jp>
```

定義している行だけを変更して、ファイル全体の参照内容を変えることができる。マクロの定義は、行頭にDを記した行で行われる。定義したマクロは、$で参照できる（**リスト2**）。

クラスはマクロに似ているが、配列変数のように、複数の文字列を定義することができる。また、定義する文字列を外部ファイル中で指定するファイルクラスもある。クラスはC、ファイルクラスはFで定義する（**リスト3**）。

ここではa、bといった1文字の名前で参照されるマクロ、クラスを定義しているが、複数文字の名前にすることもできる。この場合、名前を中カッコで括る。たとえば、D{abc}stringという文によって、abcという名前に文字列stringを定義することができる。参照する場合も同様で、${abc}と表記する。中カッコを忘れると、aという名前で参照されるので注意すること。

### ■予約名

多くのマクロ名、クラス名が、特定の用途に使うものとして予約されている。たとえば、$jはホスト名、$mはドメイン名といった具合である。基本的に1文字の小文字マクロ／クラス名は、実際に使われているか使われていないかは別として、すべてsendmailにより予約されている。何らかの理由で自分でマクロを定義する場合は、大文字か複数文字の名前を使うようにしよう（これらの名前にも予約されたものがあるので注意すること）。

予約マクロには、sendmail.cf中で明示的に定義しなくても、sendmailプログラムが自動的に初期値を設定するものもある。sendmail.cfを見ていて、どこにも定義されていないのに使われている名前があれば、この種のものと考えていいだろう。よく使われる予約名を**表1**、**表2**に示しておく。

### ■ヘッダの定義

配信中のメールに付加するヘッダの内容も、sendmail.cf中で設定する。これは行頭のHで定義される。ヘッダには、Received:のように必ず追加されるもの、Date:やMessage-ID:のように、なければ追加するものなどがある。ヘッダをどのように追加するか、ヘッダの内容をどのようなものにするかといったことは、基本的にHコマンドで定義される。Hコマンドは、文字列定数や各種マクロ参照により、これらのヘッダを作成し、メールに付加する（**リスト4**）。

### ■ルールとルールセット

ルールは、メールアドレスを評価、書き換える式である。ルールは行頭のRで定義される。ルールは次のように記述する

```
R<LHS>  <RHS>  コメント
```

<LHS>（Left Hand Side）は左辺という意味、<RHS>（Right Hand Side）は右辺という意味である。LHSとRHSの区切り、RHSとコメントの区切りはタブ文字でなければならない。スペースは使えない（LHSやRHSでスペースが文字として扱われるためだ）。

**▼リスト2　マクロの定義と参照**

```
Dastring ——— aという名前でstringという文字列を定義する
Db$a ——— bを定義しているが、その内容は$aなので、bはstringになる
```

**▼リスト3　クラスとファイルクラスの定義**

```
Cc aaa bbb ccc ——— cという名前で、文字列aaa、bbb、cccを含むクラスを定義する
Fd/etc/hoge ——— dという名前で、/etc/hogeの内容を含むクラスを定義する
```

| 名前 | 説明 |
|---|---|
| a | メールの発信日時 |
| f | メール発信者のアドレス |
| j | ローカルホストのドメイン名（ホスト名を含む） |
| m | ローカルホストのドメインサフィックス（ホスト名を含まない） |
| w | ローカルホスト名（ドメイン部を含まない） |

**表1　よく使われるマクロ**

**▼リスト4　ヘッダの定義（Received:ヘッダ）**

```
HReceived: $?sfrom $s $.$?_($?s$|from $.$_)
        $.by $j ($v/$Z)$?r with $r$. id $i$?u
        for $u; $|;
        $.$b
```

| 名前 | 例 | 説明 |
|---|---|---|
| m | | ローカルドメイン一覧（V8.8以降では使用されていない） |
| t | root MAILER-DAEMON | トラステッドユーザー（信頼できるユーザーアカウント） |
| w | mail.takobeya.com localhost | ローカルホスト名の別名一覧 |

**表2　よく使われるクラス**

また、コメントに使われる#も、LHS、RHS中ではデータとして使われる。ただし、RHSがタブ文字で終了した後、行の残りの部分はプログラムから無視されるので、この部分にコメントを記述することができる。

ルールの基本的な仕組みは簡単だ。与えられた文字列がLHSにマッチすれば、RHSを評価して、入力文字列を書き換える。そして、書き換えを行っても、行っていなくても次の行に進む。

ルールは、ルールセットという形にまとめられる。ルールセットは行頭のSで始まり、0から99の番号か名前を指定することができる。あるルールセットは、別のSコマンド（別のルールセットの開始）かファイルの末尾で終了する（**リスト5**）。

これだけの説明では何もわからないだろう。実は、ルールとルールセット、そして配信エージェントの部分が、sendmail.cfの要の部分なのだ。

### ■ルールの詳細

ルールセットは、列挙されたルールの集合体に番号や名前を付けたものである。しかしその働きを実際に理解するためには、ルールの動作をある程度は知っておかなければならない。ルールは、与えられた文字列（入力文字列）をLHSと比較し、一致していればそれをRHSにしたがって書き換えて出力文字列とする。一致していなければ書き換えは行われず、入力文字列がそのまま出力文字列となる。

少し詳しく説明しよう。LHSと入力文字列は、まずトークンという単位に分割される。連続した英数字文字列は1つのトークンになる。記号類の多くは、それ1文字でトークンになる。メールアドレスに使われる「.」ピリオドや「@」は1文字で1つのトークンになる。たとえば、“ user@takobeya.com ”というメールアドレスは、「user」、「@」、「takobeya」、「.」、「com」というようにトークンに分解される。

文字列中に適当な記号があることによって、文字列はその記号も含めてトークンに分割されるのだが、どの記号が分割文字として使われているのか？ sendmailのプログラムのレベルでは、カッコ（「(」と「)」）、不等号（「<」と「>」）、カンマ、セミコロン、バックスラッシュ（¥記号）、ダブルクォーテーション（「"」）、復帰、改行文字（\rと\n）である。さらに、sendmail.cf中でこれ以外の文字を分割文字として追加指定することができる。

```
O OperatorChars=.:%@!^=/[]|+
```

Oコマンド（オプションコマンド）で「OperatorChars」に指定した文字が、さらにトークン文字列として使われる。@やピリオドなど、メールアドレスでおなじみの文字が追加していさ

**▼リスト5　ルールセット3（抜粋）**

```
S3                                  ルールセット3
R$@                 $@<@>           いろいろな書き換えルールを記述する
R$*                 $:$1<@>
R$*<$*>$*<@>        $:$1<$2>$3
R:include:$*<@>     $::include:$1
R$*[$*:$*]<@>       $:$1[$2:$3]
R$*::$*<@>          $:$1::$2
  ⋮
  ⋮
```

れているのがわかる。

スペース（空白文字）は特殊な扱いを受ける。スペースは分割文字として働くが、ほかの分割文字と異なり、空白を収めたトークンは生成されない。@やピリオドなどは、前後の文字列をトークンに分割し、なおかつその分割文字自身もトークンになるが、空白はトークンにならないのである。

分割文字を含む文字列を1つのトークンとして扱いたい場合は、文字列をダブルクォートで括ればよい。

トークンに分解された入力文字列とLHSは、トークン単位で比較される。すべてのトークンが一致すれば、入力文字列とLHSが一致したことになり、入力文字列はRHSで指定された出力文字列に置き換えられる。一致しなかった場合は、入力文字列に対して何も書き換えを行わず、そのまま出力文字列とする。

```
Rroot@takobeya.com  user@localhost
```

たとえば上記のルールに、入力文字列として「root@takobeya.com」という文字列を与えると、出力文字列は「user@localhost」になるが、「user@takobeya.com」を与えた場合は、LHSにマッチしないので書き換えは行われず、出力文字列は「user@takobeya.com」のままである。

### ■LHSのワイルドカードオペレータ

実際のルールでは、このように定数文字列に対して比較を行うのではなく、各種文字列にマッチするワイルドカードオペレータと呼ばれるトークンをLHSに指定することが多い。また、RHSのほうにもワイルドカードオペレータを指定することができる。RHSで指定したワイルドカードオペレータは、

LHSのワイルドカードオペレータにマッチしたトークンを参照するものである。ワイルドカードオペレータを使うことにより、文字列の比較をより汎用的に行うことができる。

ワイルドカードオペレータは、シェルでファイル指定に使う＊文字、エディタなどの文字列検索に指定する正規表現などに似ている。ただし、ルールで使うワイルドカードオペレータは、トークン単位で一致、不一致となる点が異なっている。

ワイルドカードは、$に続けて適当な文字や文字列を指定した形で表記される。LHSに指定できるワイルドカードを**表3**にまとめておく。

次のルールは、ユーザー名@ドメイン名という形式の文字列が入力されるとマッチする。

```
R$-@$+  user@domain
```

たとえば、「masa@takobeya.com」という入力文字列を与えれば、「masa」が$－（任意の1個のトークン）にマッチし、「@」はそのままマッチする。「takobeya.com」は3つのトークンに分割されるが、これは$＋（1個以上の任意のトークン）にマッチする。その結果、出力文字列はuser@domainに書き換えられる。しかし、入力文字列が「masa@」、「masa.sakaki@takobeya.com」だったりするとマッチしない。「masa@」の場合は、@以降がないので、$＋の1個以上のトークンという条件を満たさない。「masa.sakaki@takobeya.com」の場合は、ユーザー名の部分が3個のトークンになるので、$－の1個のトークンに続けて@があるという条件を満たさないからだ。

### ■RHSのオペレータ

LHSにワイルドカードオペレータを使うことで、さまざまな条件の文字列を判定することができる。しかし、実際に入力文字列を有意義に書き換えるためには、RHSで記述する出力文字列中で、入力文字列を参照できなければならない。これを実現するのがRHSのワイルドカードオペレータだ。また、RHSのオペレータには、入力文字列のトークンを参照するもののほかに、ルールセットの制御などのためのオペレータも用意されている。

まずは、入力文字列について使用するオペレータ（**表4**）を説明しよう。

#### ●$n　LHSのトークンを参照

入力文字列のトークンの参照には、$n（nは1から9の数字）というオペレータを使う。1から9の数値は、入力文字列の個々のトークンを参照するものではなく、LHSで指定されたワイルドカードオペレータの位置に該当するトークンを参照するものである。最初のワイルドカードオペレータにマッチしたトークンは$1、次のワイルドカードオペレータにマッチしたトークンは$2というようになる。そして、そのワイルドカードオペレータとマッチしたトークンから構成される文字列を出力するのである。

```
R$-@$+  user=$1/domain=$2
```

この場合、$1はLHSの$－、$2は$＋に対応する。このルールに、「masa@takobeya.com」という入力文字列を与えると、$1は最初の$－にマッチした1個のトークン（「masa」）、$2は$＋にマッチした1個以上の任意のトークン（「takobeya.com」）を参照する。結果として、このルールの出力文字列は、「user=masa/domain=takobeya.com」となる。

$nの形で参照されるトークンは、LHSの記述と入力文字列に応じて、数が変わってくる。$－は常に1個のトークンとマッチするので、これを参照する$nは常に1個のトークンを表すことになるが、$＋の場合は1個以上の任意の数のトークンを表すことになる。また、0個以上のトークンとマッチする$＊の場合は、$nで参照されるトークンがないという場合もある。この場合、参照結果は空文字列となる。

#### ●$[と$]　DNSの参照

入力トークンの参照とは別に、トークンに対して特殊処理を行うオペレータもある。「$[ トークン $]」というオペレータは、$[と$]で囲まれたトークンからなる文字列をホスト名と想定し、DNSに問い合わせを行い、ドメイン名を正規化する。たとえば、省略された形式のホスト表記を正式なドメイン名

| オペレータ | 説明 |
|---|---|
| $＊ | 入力文字列中の任意の数のトークン（0個を含む）とマッチする |
| $＋ | 入力文字列中の1個以上の任意の数のトークンとマッチする |
| $@ | 0個のトークンとマッチする（該当する位置にトークンがない場合にマッチする） |
| $－ | 入力文字列中の1個の任意のトークンとマッチする |
| $=クラス | 入力文字列中のトークンとクラスに含まれるトークンを比較し、一致するものがあれば（複数可）マッチとする |
| $~クラス | 入力文字列中のトークンとクラスに含まれるトークンを比較し、一致するものがなければマッチとする |

表3　LHSで使えるワイルドカードオペレータ

| オペレータ | 説明 |
|---|---|
| $n | LHSのn番目のトークンを参照する |
| $[と$] | DNS検索を行う |
| $(と$) | データベース検索を行う |

表4　RHSのオペレータ

にする（mailをmail.takobeya.comにするなど）といった処理を行う。DNSへの問い合わせでは、MXレコードも調べられる。$[と$]に囲まれたトークンで表される文字列に対して行われる処理を以下に示す。

**・ドメイン名が省略表記だった場合**
**正規形式のドメイン名に書き換える。**

**・ドメイン名に対応するAレコードがあった場合**
**Aレコードで返されたドメイン名に書き換える。**

**・ドメイン名に対応するMXレコードがあった場合**
**正規形式のドメイン名に書き換える。**

**・[192.168.1.4]という形式のIPアドレスの場合**
**逆引きを行い、検索された正規形式のドメイン名に書き換える。**

場合によっては、指定した名前が見つからないなどの理由で、DNSで名前解決ができない場合がある。この場合は、入力文字列がそのまま出力文字列となるが、必要なら、デフォルト文字列を指定することもできる。

```
R$-@$+  $1@$[$2$:ERROR$]
```

このように記述することにより、検索に失敗した場合は、出力文字列はuser@ERRORになる。実はこのルールの記述にはちょっとした問題がある。これについては次の節で説明しよう。

**●$(と$) データベースの参照**

$(と$)は、外部のデータベースファイルにアクセスする機能を提供する。検索するデータベースと検索するキーを指定すると、そのキーを検索し、該当するデータを出力文字列とする（長くなるのでここでは説明しない）。

**■RHSの制御用オペレータ**

RHSには、LHSを参照するオペレータのほかに、ルール処理を制御したり、ほかの処理を呼び出したりするためのオペレータも用意されている。これを**表5**にまとめておく。

これらのオペレータについて簡単に説明しよう（この説明は、オペレータの機能を解説するものだが、すべての詳細を網羅しているわけではないことに注意。すべてを知りたければコウモリの本を読むべし）。

**●$: 1回だけ評価する**

$:は、ルールを1回だけ評価して次に進むことを示すプレフィックスである。

ここまでルールの説明は、入力文字列に対して1回だけ評価を行うものとして解説してきたが、実際のルール動作はちょっと違う。ルールは、入力文字列に対してLHSにマッチするかどうかを判定し、マッチしたらRHSにしたがって書き換えを行うが、この書き換え結果をさらにLHSと比較するのである。その結果、再度LHSとマッチすれば、再び書き換えを行う。この処理を、マッチしなくなるまで繰り返すのである。最後にマッチしなくなったら処理は完了だ。その結果、出力文字列は、最後にマッチした状態のまま残る。一方、最初からマッチしなかった場合は、入力文字列が出力文字列となり、そのまま次に送られる。

内部の動作について簡単に触れておこう。入力文字列は作業用バッファに書き込まれる。そして、そのバッファとLHSを比較し、一致していれば書き換えるというループ処理に入る。このループは、バッファとLHSが一致しなかったときに終了する。これにより、文字列がLHSに一致する限り、書き換えが繰り返される。一致しなくなったら、ループを終了する。もし、最初から一致していない場合は、ループを1回も通らずに、つまり書き換えを行わずに次の処理に進む。いずれにせよ、バッファに残った内容は、ルールの処理結果となるわけだ。

$:は、このLHSにマッチする限り、何度も書き換えを繰り返すという動作を抑止するものである。入力文字列がLHSと比較され、1回書き換えを行ったらそれで処理を終了する。例を考えてみよう。入力文字列の末尾に、文字列「@takobeya.com」という文字列を追加するマクロを考えてみよう。

```
R$+  $1@takobeya.com
```

$1はLHSの$＋を参照するので、たとえば入力文字列「masa」に「@takobeya.com」を追加した結果を出力文字列とする。しかし、このルールの評価はここで終わらない。「masa@takobeya.com」は$＋にマッチするので、再度「@takobeya.com」が追加されるのである。これが延々と繰り返され、エラーとなってしまう。これは意図した動作ではないだろう。

| オペレータ | 説明 |
|---|---|
| $: | ルールを1回だけ評価する |
| $@ | ルールを評価した後、ルールセットからリターンする |
| $>ルールセット | 指定したルールセットを使って書き換えを行う |
| $# | 配信エージェントを指定する |

表5 各種制御用のRHSオペレータ

正しく処理するには、次のように記述する必要がある。

```
R$-  $1@takobeya.com
```

$−は1個のトークンとマッチするだけなので、1回目の評価が終わった後の「masa@takobeya.com」とはマッチせず、意図した結果が得られる。しかし、入力文字列が「masa.sakaki」というように複数のトークンに分割される場合はマッチしないので、文字列の追加が行われない。入力文字列が複数のトークンから構成される場合は、$:を使えばいい。

```
R$+  $:$1@takobeya.com
```

$:により、ルールの評価が1回だけ行われることが保証され、文字列の追加が1回だけ行われる。

この例は、無限に文字列が長くなるというパターンのエラーだが、これとは異なるパターンもある。たとえば、少し前に説明した次のルールは無限ループに陥る。

```
R$-@$+  $1@$[$2$:ERROR$]
```

このルールにuser@takobeya.comという文字列を与えると、takobeya.comについてDNS参照が行われて、user@takobeya.comという出力文字列になる。RHSが評価されたにも関らず、実質的な書き換えは行われていないことに注意。当然この結果はLHSに再びマッチしてしまう。つまり無限ループになるわけだ。このような無限ループはsendmailによって検出される。

こういったルールの問題点を調べるには、しばらく後で解説するsendmailの-btオプションが便利に使える。

### ●$@　ルールセットからリターン

$@もプレフィックスで、ルールセットからのリターンを指定する。入力文字列がLHSにマッチすると、RHSにしたがった書き換えを行った後、そのルールセットを終了する（次のルールに進まない）。マッチしなかった場合は、書き換えを行わず、次のルールに進む。いわば、サブルーチンの途中で処理を終了するようなものである。

$@を使用した場合、書き換え結果を再度LHSと比較するという処理も抑止される。つまり、入力文字列とLHSがマッチした場合、書き換え処理を1回だけ行い、そのルールセットは終了するのである。

ルールセットは、sendmail内部から、あるいは別のルールセットから呼び出されるが、いずれの場合も、最後に実行したルールの出力文字列がそのルールセットの処理結果として呼び出し側に返される。$@で途中終了した場合は、その$@オペレータを含むルールによる処理結果が最終的な結果として渡される。

### ●$>　別のルールセットの呼び出し

$>は、別のルールセットの呼び出しである。$>に続けて、呼び出すルールセットの番号か名前を指定する。RHSの書き換え用の文字列は、ルールセットを示す数字や名前の後に空白を置いてから記述する。予約ルールセットはsendmailから呼び出されるが、それとは別に独自のルールセットを作成し、それをほかのルールセットから呼び出すことができる。

入力文字列がLHSにマッチすると、それがRHSにしたがって書き換えられる。そしてその書き換え結果が、$>で指定したルールセットに対する入力文字列となる。呼び出されたルールセットは必要に応じてこの入力文字列の書き換えを行う。呼び出されたルールセットが終了すると、そのルールセットが書き換えた（書き換えないかもしれない）結果がRHSの処理結果となる。1回だけ書き換えを行う$:プレフィックスが指定されていれば、これで次のルールに進むが、指定されていない場合は、ルールの評価規則にしたがって、この結果を再度LHSと比較し、マッチすれば、RHSにしたがった書き換え（そして別のルールセットの呼び出し）を繰り返すことになる。

### ●$#　配信エージェントの選択

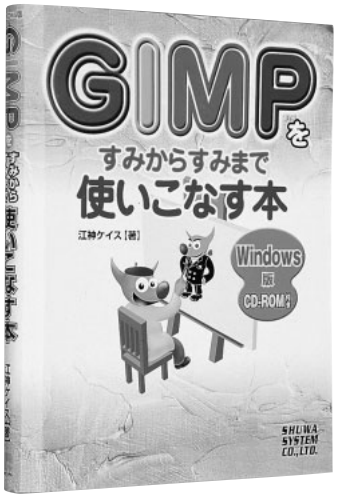
$#は、配信エージェントを選択する。入力文字列がLHSにマッチした場合、RHSで指定した書き換えを行った後、Mコマンドで定義した配信エージェントを呼び出す。$#を使った場合も、書き換え後のLHSとの再比較／書き換えというループは抑止される。これについては、またあとで解説する。

### ■さて次回は

今回はここで誌面が尽きた（担当編集によれば、それでもいつもより1ページ多いらしい）。

次回は、実際にsendmailがメールの配信を行う際のルールセットの流れ、ルールセットによるアドレスの処理、RHSの$#コマンド（配信エージェントの選択）とMコマンド（配信エージェントの定義）について解説する予定だ。また、sendmailの動作チェックを行う際に利用するテストモードの使い方などもあわせて説明するつもりだが、実際に何を書くかはそのときになってみないとわからない。

では、また来月。




## GIMPをすみからすみまで使いこなす本

**江神ケイス 著**
**秀和システム**
**B5変形判／214ページ／CD-ROM1枚付き　　本体価格 2400円**

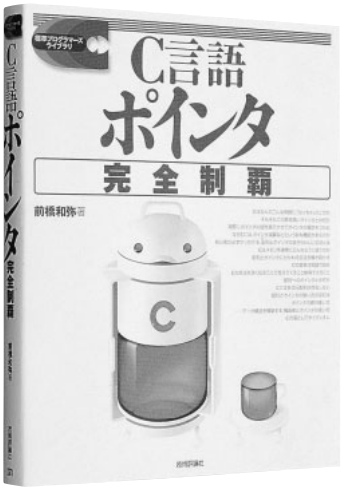
Windows版のGIMPは以前より開発されていたが、いつの間にか「バグが多い」といったネガティブなイメージが定着してしまっていた。本書の著者も述べているように、手軽に利用できるペイント系ソフトを求めていたWindowsユーザー達も、GIMPという名前を知ってはいても手が出しづらい状況にあったのだ。しかし、最近のバージョンではかなり安定して動作するようになっている。GTK+の多言語対応機能を利用してメニューも日本語化され、しだいに日本のWindowsユーザーの間でも広まってきているようである。

Windows版はLinux版とまったく同じユーザーインターフェイスを備えている。つまりWindows版の解説書である本書をLinux版の解説書として読むことができるわけだ。GIMPのテクニックが全ページカラーでわかりやすく紹介されおり、双方のユーザーにとって格好の入門書だ。CD-ROMにはWindows版のバージョン1.1.28を収録。

## C言語 ポインタ完全解説

**前橋和弥 著**
**技術評論社**
**B5変形判／324ページ　　本体価格 2280円**

どのプログラミング言語に限らず、そもそもプログラミングを書籍を読むだけで習得するのは非常に難しい。特にC言語は、ポインタの概念や取り扱いで挫折してしまう人が多いとされている。実際、本書以外にもポインタについてのみに絞って解説している書籍も多く、このことからもポインタについて理解することがC言語習得の「壁」になって存在していることが窺える。本書では、ほかの高級言語にも存在するポインタという概念が、なぜC言語ではこんなにも「ややこしい」ものになってしまったのかというあたりにツッコミを入れながら、ポインタに関する問題を整理して、もともとのポインタの定義や正しい使い方について軽快な文章で解説している。また、これまで「C言語の常識」とされていた事柄について再考し、プログラマーが陥りやすい誤解などを指摘しているので、すでにCをマスターしている方も自分のポインタに関する理解を再確認してみてはいかがだろうか。





## 今年こそ引く！　インターネット常時接続 導入ガイド

**村上俊一 著**
**翔泳社**
**B5変形判／192ページ　　本体価格 1680円**

不毛と言われ続けてきた日本のインターネットアクセス環境だが、この1年あまりの間に急速に整備されてきている。特に常時接続については、CATVにフレッツ・ISDN、ADSLときて、さらには光ファイバまで、より高速な通信環境が提供されるようになりつつある。このところ毎日のように新たなサービスのアナウンスがあり、それが新聞やビジネス誌（ときには週刊誌や女性誌）などの一般のメディアで取り上げられることも多い。

一般の認識度が高まり選択肢が増える一方でユーザー側には混乱もあるようだ。どんなサービスがあってどのサービスが良いのか、申し込みはどうするのか、すぐにでも加入すべきなのか待ったほうがいいのか、そもそも常時接続って必要なのか？　本書は、こういった疑問に答えるのはもちろん、実際に加入するまでの手順を、申し込み書の記入例、工事のようす、機器の設置など、写真を交えて紹介している。今が旬の内容なのでお早めにどうぞ。


## ただ。 無料インターネットサービス厳選ガイド

**関谷博之 著**
**ソフトバンク パブリッシング**
**A5判／272ページ** **本体価格 1600円**

「ただ」と聞くと「ただより高いものはない」という、ことわざを反射的に思い浮かべてしまう心配性な性格のせいか、無料サービスというものにはどうしても警戒感が先に立ってしまう。でも、考えてみれば（本書にも書かれているとおり）地上波の民放っていうのは無料サービスなわけで、心配性であっても、さすがにテレビまで警戒しながら見ているというわけではない。その代わり、民放を見ている間はスポンサーのCMを「見せられ」ているのだ。

テレビと同じく情報伝達手段のひとつであるインターネットにも、この「民放方式」の無料サービスが増えてきた。本書はこうしたインターネットの無料サービスを紹介している。細かいサービス内容や専用ツールの使い勝手などを検証しているので、たとえば同じ無料メールアカウントでも、どのサービスのどの点が良くて、どの点が悪いのか比較できるようになっている。「使える」無料サービスをお探しの方にお勧め。

## fmlバイブル

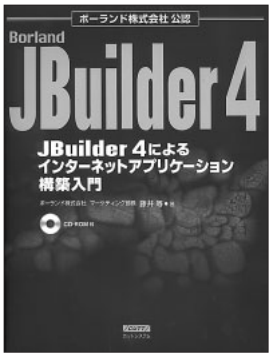

深町賢一 著

オライリー・ジャパン／オーム社

**B5変形判／524ページ**

本体価格 4200円

## Jbuilder 4による インターネットアプリケーション構築入門




藤井 等 著

カットシステム

**B5変形判／356ページ／CD-ROM1枚付き**

本体価格 3800円

## Miracle Linuxネットワーク構築ガイド


原 昭浩 著
日本オラクル／ミラクル・リナックス監修

リックテレコム

**B5変形判／296ページ**

本体価格 2800円

## Rubyを256倍使うための本 極道編


助田雅紀 著

アスキー

**B6判／240ページ**

本体価格 1200円

# 読者の声

俺にもいわせろ！

初めまして。今月、読者ページを担当させていただく清水です（いつもの担当者は逃げました）。Linux、いやコンピュータ全般について、あまりの無知さに編集部員を驚かせる日々を送っています。未熟者もいいところ、ナマモノの私なのです。ゼロから勉強していきますので、読者の皆様のなかにも、超初心者の方がいらっしゃいましたら、一緒にLinuxを始めてみませんか？ ベテランの皆様、ご指導、ご鞭撻のほどよろしくお願いいたします。

ところで、就職活動まっただ中の私が、こんなところ（いや、失礼しました）にいてもよいのでしょうか？　実社会でやっていけるのでしょうか？　私のLinuxマシンの先行きとともに、今一番の不安材料です。

### 負けるな、ガベコレ!!

**Linuxガベコレが最高！　これがわからん編集者は脳みそ8086かZ80程度。**

**P.S.　しのはら様。カーネルはサンダースだけでなくカーネルシンボリも説明してほしかったです。**

**（E-Riderさん）**

**アンケートの「おもしろかった記事」投票、ガベコレに1票入れました。しのはらさん、27位でもめげずに頑張ってください。**

**（大塚順治さん）**

**Linuxガベコレの人気が低迷しているとのことですが、残念でなりません。私はこの連載が読みたいがために、他誌から購読誌を変えたほどです。もし、この連載が中止になるくらいなら、再び購読誌を他誌に変えるかもしれません。ぜひ連載を継続して夢の巻頭カラーを目指してください。**

**（竹内幸一さん）**

前々号のランキングでは不人気の結果に終わったガベコレに、励まし（？）のメッセージをたくさんいただき、ありがとうございました。おかげさまで、前号ではかつてないほどの投票数をいただき、今月はカラーページ（一部モノクロ）でお届けします。

皆様のメッセージを励みに、「より一層アクの強さに磨きをかけていく（しのはら氏談）」とのことです。これからもよろしくお願いします。

### あなたの町のインフラ事情

**最近、CATVの常時接続環境を手に入れた。ほとんどWindowsで使用しているが、つい先日、Linuxでも対応させた。最近の雑誌では、フレッツ・ISDNのことで盛り上がっているが、CATVとなると、日本全国にたくさんのCATV局があるせいか、特集された雑誌を見たことがない。私の場合は、CATV研究所のWebページ(http://www.catv.co.jp/)で情報を集めたわけだが、それでも設定に1週間かかった。**

**今度、主なCATV局別の設定方法を特集していただけるとありがたい。常時接続はフレッツ・ISDNだけではないですよ。**

**（セリケンさん）**

**当地もようやくフレッツ・ISDNのスケジュールに入ってきたようで、うれしく思います。子どもたちも自分のアカウントを持つようになり、ルータを導入し、LANを引く計画です。**

**今年の大雪でFMアンテナが倒れ、工事が必要になりましたので、ついでに屋内配線もしてもらうことにしました。雑誌記事も、技術進歩の最先端を追いかけるだけでなく、後からついていく地方在住のユーザーもフォローしてください。**

**（[直角] さん）**

担当の実家では、市内（そもそも市ではないが……）にアクセスポイントがないため、ISDNで市外のアクセスポイントに繋いでいるとか。インターネットが普及することで、情報伝達の地域間格差が縮まるはずなのに、その前提となるインフラ整備の点で、大きな地域間格差が生じてしまうとは、なんとも皮肉なことですね。

ちなみに担当は、首都圏に住みながらも、キャッシュフローに余裕がないため、なかなか常時接続に踏み出せないのでした。

### 未知と冷や汗の共存？

**パソコン歴は、1年未満ですが、Linuxを知りたくてインストールに挑戦しています。今のところ、3回挑んで、すべて撃沈しています。知識がある人にはちょっとしたことでも、すぐにひっかかってしまうのです。サルでもわかるインストールの解説はないものでしょうか。**

**（水戸部 長雄さん）**

㊗慣れている人にとっては、何でもないエラーでも、未知の世界でのできごとでは、すべてが冷や汗の連続ですよね。起こりうるすべてのエラーに対処できる解説記事を作ることは困難ですが、少しでもわかりやすい記事をお届けできるよう、いっそう努力してまいります。

### はじめの一歩

**LinuxがUNIX互換のOSだということは、前より知っていたので、難しいと思い手を出さずにいました。私の住んでいる地方の書店でもLinuxの雑誌が数冊並ぶようになり、その中で貴誌の表紙が果物で、なんとなくLinuxも簡単に扱えるのではないか？　という内容だったので購入してみました。これがそもそも間違いの元で、貴誌が出るたびに違うディストリビューションをインストールしています。しかし、インストールするのが精一杯で、周辺機器はおろか、CD-ROMに収録されたソフトさえ使えないでいます。いつかはLinuxを使えるようになりたい思い、悪戦苦闘している今日この頃です（表紙が果物でなければこんなに苦労しないですんでいたかも？）**

**でも、なぜかLinuxをやめられないでいます。どうしてでしょうか？**

**（gia_moさん）**

**Linuxを使ってみようと思ってから、6カ月くらいたってしまいました。「専門書などを見ればなんとかなるだろう」と、簡単な気持ちで、いろいろな本を買いチャレンジしていますが、なかなかうまくいきません。すでに何度か挫折もし、月日のみがたってしまいましたが、「ここまできたら必ず成功するまでやってやる」と、このごろ思っています。記事の内容もまだわからないことが多いですが、勉強して早くLinuxを使いこなせるようになりたいです。**

**（Masayukiさん）**

㊗そうなんです、何を隠そう私も表紙のあの果物に引っかかってしまった一人です（笑）。気軽に、とりあえず興味のあることから始めていくのがよいのだ！　と信じて、挑戦する毎日です。

本だけではイメージがつかみにくく、理解しづらいこともしばしばです。周囲に詳しい人がいれば一番よいのかもしれませんが、そうもいかない皆様のお役に立てる存在になれるよう、精進してまいります。ご意見、ご質問などがありましたら、Webのアンケートページへお願いいたします。

### お忙しい中、ご購読ありがとうございます。

**いつも読ませていただいております（遠方？　のため約半月遅れです）。細かいファインチューニングの仕方なども載せていただけたら幸いです。**

**（駐在員@ブラジルさん）**

**だんだん読む機会が減りつつあります。何だか上手く時間を作ることができない生活が続いているのです。仕事の合間に読むことしかできず、効率がよくないです。**

**（別府 航さん）**

㊗時間の使い方って難しいですね。予定が詰まっているときには、時間が足りないって思っているのに、予定が空いてもなかなか上手く使えない。試験期間中に限って、雑然とした部屋がみょ〜に気になる心境といったところでしょうか？

### カーネルインストール

**カーネルを2.4.0にしようとしたけど、modules_installで引っかかってしまいました。何をやっていいかわからず。**

**初心者にはやはりちょっときつい。またこのつまずきがよいものですね。**

**（ウルトラ　アントさん）**

**3月号を読んで、カーネル2.4のインストールを少し考えたけど、今使っている状況がそれなりに安定しているので、無理をせず、ディストリビューションがカーネル2.4を使用するまで待つ予定だけど……もっと積極的になったほうがよいのかな？**

**（松村 岳さん）**

㊗make modulesまではうまくいっているでしょうか。ここまでで問題がなければ、make modules_installで失敗することはまれですので、もう一度チャレンジしてみてください。＞ウルトラ　アントさん

業務に使用しているマシンや、サーバなどでLinuxをお使いの場合は、安定稼働している環境を無理に変える必要はないと思いますよ。＞松村さん　多少のトラブルが起きても大丈夫という環境でしたら、現在のカーネルでも起動できるようにして、カーネル2.4を試してみてはいかがでしょう。