Linux☆萌えるマシンにする方法

～使えるマシンのつくりかた～

さぶそにっく！

目 次

第1章　萌えるマシンってなぁに？

1.1　はじめに

いきなりですが、萌えるマシンって何なんでしょうか？　キャラ萌え、ぷに萌えなど、「萌え」にはいろいろありますが、マシンに萌えを感じるのはどんな時でしょうか？

私はマシンに萌えを感じるのは次のような時ではないかと思います。

- 「っかあ~ッ使えるぜこのマシンは」って思うとき
- 「ふっふっふ、このマシンは君のザクとは違うのだよ、ザクとは」ってほくそえんじゃうとき
- 自分の好みが細かいところまで反映されているマシンに仕上がっているとき

こんなマシンが「萌え」マシンだと思うのですがどうでしょう？

いわゆるパソコン用 OS の Windows や MacOS では長年のユーザの努力やメーカの「使いやすい」マシンを作るというコンセプトのため、上に挙げたような条件はかなり満たされている観があります。しかし Linux ではこれらの条件はまだまだ満たされているとは言えません。

この本では「萌えられるマシン」をキーワードに、使える Linux マシンの作り方の一例をお見せしたいと思います。

1.2　本書の構成

各章で扱っている話題は次のようになっています。

萌えるマシンとは使えるマシンだ！

この章ではどうすれば日常の作業に使えるマシンになるかを中心にお話します。

日本語環境や通信、画像の閲覧、音楽やムービーの再生などが中心です。

萌えるマシンとはとんがったマシンだ！

　この章ではテック系なユーザーのためのとんがった仕様のマシンの例と、それをどのように Linux で使いこなすかについてお話します。

萌えるマシンとは自分だけのマシンだ！

　この章では流行に流されない（笑）ポリシーのある環境作りについてお話します。

第2章　萌えるマシンとは使えるマシンだ！

2.1　日本語環境の構築

私たち日本の萌え系ユーザーにとっては、日本語環境が使えなければどうにもなりません。英語しか出ない環境で「はにゃ～ん☆」なんて言えないですからね。最近のディストリビューションでは、最初から日本語環境を作り込んであるので、そういうディストリビューションで Linux を始める人にとってはこのへんは考えなくてもいいことかもしれません。

私は RedHat4.2 の頃にメインマシンの環境を整備しましたが、その頃はまだ日本語ディストリビューションというのはありませんでした。まぁ強いて言えば Debian がそれに近かったかもしれません。

その頃、日本語環境を構築するには、英語環境（というか 1byte 文字コード環境）のベースディストリビューションをインストールし、その上に JE や PJE などの日本語アドオンパッケージをインストールしていました。最近の日本語 Linux ディストリビューション開発の背景には、「アドオンパッケージでは自ずと限界がある」という考えがありますが、私はそれでもどちらかというと PJE のような形態の方が好きですね。最初から日本語化されたパッケージでは日本語を強要されるような部分もありますから...

さて、具体的な日本語環境の構築の話ですが、必須なのは次のソフトウェアです。

- 日本語エディタ
- X のフォント
- かな漢字変換ソフト

とりあえずこれだけはどうしても必要です。以前、Alpha 用 Linux を整備したときに mule と wnn4.2 をインストールするまではほとんど使いものにならなかったのを覚えています。(^^)

さらに追加で

- kinput2 のような X 用インプットメソッド
- TEX
- VFlib
- 日本語 TrueType フォント
- 日本語対応 ghostscript
- X-TT（松原葵）
- Netscape 用日本語対応リソース

なんてところもおさえておきたいです。
あとは好みですが

- jman
- X-jman
- JF

あたりのマニュアル・文書も日本語のものをおいておくと楽ができます。

2.1.1 mule

まずはこれがなくては始まらない、エディタの話です。エディタの好みは「宗教」とも言われるほど、人によって信念めいたものがあり、話をするのは難しいのですが、emacs 中毒の私としては、多言語対応版 emacs である mule をおすすめします。

emacs は非常に多機能で、おそらく地球上で最も強力なエディタでしょう。最初は非人間工学的と言われるキーバインドに眩暈がしますが、使っているうちにだんだん中毒症状を呈してきます。特に Ctrl-k によるカーソル以後のカットと Ctrl-a による行の先頭への移動が麻薬のように効いてきます。（笑）

そして emacs 以外の unix 用ソフトウェアを使った時でも多くのキーバインドが有効なことに気づくと、だんだんやりたいことと emacs キーバインドが融合してきて、脊髄反射的に指が動いてしまうようになるのです。（これは多くのソフトウェアで、gnu のコマンドラインインタフェースライブラリを使っていることに由来します。）

こうなるともはや emacs 以外のエディタを使うのが苦痛になってきてしまいます。生産性を維持するためには emacs 以外を使うことができなくなり、この頃からなんでも emacs でやろうとし始めます。(^^; この結果が emacs に豊富に用意されている MUA やニュースエージェント、lisp による各種メジャーモードアドオンファイルです。

さて、では実際にどんな emacs を使えばハッピーになれるのでしょうか？　最も新しいリリースは emacs 20.5 で、これには mule 拡張が統合されています。ver.20 以降の新しい emacs では lisp 処理系なども強化されていて、新しめのアドオンはこれらでなければ満足な動作をしません。

しかし 20 以降の emacs は、それ以前の emacs と比べると重く、また多言語対応を「美しく」実装しているため、leim というアドオンを中間層に入れる必要があります。また主要なかな漢字変換システムである Wnn を正式にはサポートしておらず[1]、Wnn を使うためには非公式パッチを当てる必要があるなど、使い勝手の点であまり嬉しくありません。また日本語対応に関しては、コード変換のためのキーバインド (Ctrl-x Ctrl-k f) も変更されており、同機能を愛用している私からするとかなり「かったるい」です。

そんなわけで、私は emacs 19.34.1 ベースの mule 2.3 をいまだに愛用しています。おそらく大半の日本語対応ディストリビューションでも、同様の理由 (C-x C-k f は除く（笑）) で同バージョンを採用しているものと思われます。

自分で emacs 19.34.1 ベースの mule 2.3 をコンパイルするのはかな漢字変換システムさえコンパイルできればそう難しくありません。mule はかな漢字変換システムとの通信用に、かな漢字変換システム側のライブラリをリンクするため、mule のコンパイルに先だって、かな漢字変換システムのコンパイルを完了していなければならないのです。

Wnn 4.2 を使う場合について言うと、後述する FreeWnn になる前は Wnn のコンパイル前の設定作業が面倒たったので x86 マシンでも mule のビルドはやや大変でした。また、Alpha マシンでは Wnn 4.2 のコンパイルがかなり困難だったため、mule のコンパイルも大変でした。

現在では FreeWnn のコンパイル自体は簡単になったので、mule をコンパイルするのもそんなに苦労しないはずです。Alpha ではその後ビルドしていないので不明ですが...

2.1.2 かな漢字変換ソフトウェア

かな漢字変換は、最近では商用のものも数種類発売され、かなり選べるようになってきています。やはり変換効率では商用のものが群を抜いて優れているようです。

しかし Linux でかな漢字変換を選ぶ場合、変換効率だけが選択基準にはなりません。

- mule へ直接入力ができるか

[1]「たまごっち」はもうリリースされたのでしょうか？

- XIM プロトコルによって X クライアントへ変換結果を渡すことができるか
- 同時接続数に制限はないか

といった要素も含めて選択する必要があります。特に Linux で文章入力をやろうとする場合、mule との親和性が重要になります。また、最近の恵まれたハードウェア資源を活用して、いくつも mule を起動して文章編集をしようとすると、かな漢字変換サーバとの接続数が限定されていると問題となります。

mule 側と直にやりとりができるかな漢字変換システムは、

- canna
- Wnn4 (FreeWnn)
- SJ3
- Wnn6
- SKK

です。この中で変換効率のもっともよいのは Wnn6 なのですが、これはオムロンから発売されている商用かな漢字変換システムで、残念ながら標準ライセンスでは同時接続数がたったの 2 接続に限定されています。これでは mule 一つと kinput2 を使うだけで接続可能数を使い切ってしまい、それ以上の使い方をすることができません。特にマルチユーザで使う可能性のあるシステムでは、この接続数制限は大変厳しいものになるでしょう。

そこで変換効率はかなり悪化するものの、接続数に制限のない Wnn4 をお勧めします。Wnn4 はリリースからかなりの時間が経ったソフトウェアで、その間改変を認めない Wnn 独特のライセンスのために目立った進歩がありませんでした。

しかし、1999 年 4 月に Wnn4 をめぐる環境は大きく変化しました。オムロンが中心となって、Wnn4 の著作権を持つ各社に働きかけ、Wnn4 のコードが改変自由となり、GPL に基づく配布形態へと移行しました。この直後、これまで非公式パッチとして存在していた変換サーバのパッチが Wnn4.2 に統合され、FreeWnn として再出発することになりました。

私はこの FreeWnn の Ver1.1 を使っていますが、これまで ノートマシン等で使っているとよく変換サーバが死んでいたのが、全く死ななくなるなどの大きな信頼性の向上がありました。

また、Wnn4.2 がリリースされてから現在までのかなりの年月の間に、数回に渡って X のリリースが行なわれたため、imake による設定に関するファイルが変わってし

まい、Wnn4.2 のコンパイルはなかなか大変だったのですが、FreeWnn ではおなじみの configure スクリプトが採用され、インストールが実に簡単になりました。

今後は FreeWnn 独自の改良がなされていくことが期待されます。とはいえ、現在のところ初回のパッチ以外、FreeWnn 独自の改良というのは全くされていませんが[2]。まぁ、改良したくなれば勝手にやっていいわけで、そういうところも含めると本当に良いかな漢字変換システムなのではないでしょうか。

発表当時は変換効率の高さから注目されていた Wnn4 も、現在の水準から見ると非常に変換効率の悪い部類に入ります。よしだともこさんのまとめた「よしだともこ必殺パラメータ」を使うことでかなり挙動は改善しますが、その後はもうひたすら辞書の登録数を増やしていくしかありません。私はこれまでにおよそ 850 語を登録ました。

Wnn4 の辞書には自然科学系とコンピュータ系の単語は比較的充実しているが、日常生活系の単語には意外なほど弱い、コンピュータ系でも最近の単語には弱い、アニメ・マンガ系、パソ通系の単語にも弱いという傾向があります。

私の使う単語はこれらの分野のものが多いので、このような領域をカバーしたいい辞書があれば是非使ってみたいと思います。私の辞書がもう少し育ったら公開してもいいんですけどね。

2.1.3 フォント、印刷環境などについて

これまで見てきたエディタとかな漢字変換は、日本語環境の中の入力部分です。この他に整備しなければいけないのは、

- X ウィンドウでのフォント環境
- 印刷のための GhostScript を中心としたフォント・ラスタライズ環境

です。このへんの話は非常に奥が深いので、今回はごく簡単に触れるだけにしておきます。

X-TT

FreeType という TrueType フォントのラスタライズライブラリと、X サーバを融合させて X の画面表示に TrueType を使えるように拡張したものが X-TT です。

[2] 小さなバグの修正と対応 OS の拡大、そして基本辞書を pubdic から pubdicplus へ変更した FreeWnn 1.1.1 が 2000 年 1 月にリリースされるそうです。

このソフトウェアによって X のフォント表示は格段に奇麗になりました。最近のディストリビューションではほとんど X にこのパッチを当ててビルドしているはずです。

以前の X 環境での日本語フォントでは、品質の高いベクトルフォントが使えず、多くの場合、画面表示を改善するためには固定サイズのフォントを多数インストールして alias を組むことで対応していました。

この方法だとサイズによって書体が変わったり、適切なサイズのフォントがない場合にはビットマップの拡大・縮小を行なって表示していたためにフォントフェイスがひどく汚くなったりしていました。

Netscape でのフォント表現や、大きなフォントを使用するソフトウェアを使うと X-TT の威力がわかると思います。

GhostScript および印刷環境の構築

unix 環境では古くは roff によるタイプセット、近年では TEX によるタイプセットが普通であり、これらの出力を PostScript ファイルに変換し、印刷は PostScript ファイルから行ないます。unix ユーザーの趣味がそうさせたのか、あるいは印刷環境の追求を理詰めで行なった結果がこうなったのかは定かではありませんが、どんなものでも一旦 ps ファイルにして印刷するというのは unix の文化になってしまっています。

PostScript プリンタがあれば、印刷に関しては ps ファイルをだらだらとプリンタに投げてやればいいのですが、個人用 PostScript プリンタというジャンルは様々な事情から全く流行っていません。そこで PostScript ラスタライザをコンピュータ上で実行して、その結果をプリンタに投げる方法が良く用いられています。また、PostScript プリンタがある場合でも、ディスプレイ画面向けのラスタライザは ps ファイルのプレビューになくてはならないものです。

このラスタライザが GhostScript です。普及に伴って GhostScript にもユーザによる拡張が加えられ、VFlib を使ってスケーラブルフォントを用いたラスタライズが可能になったり、ps ファイルと内部構造の似ている pdf ファイルのラスタライズが可能になったりしました。各ディストリビューションに含まれる GhostScript は多くの場合このような拡張パッチを適用した拡張版ですが、少し前のバイナリ配布された GhostScript 5.10 には困った問題がありました。

それはランドスケープモード（横長画面）でページをラスタライズしたとき、画面の有効範囲のハンドリングが正しく行なわれておらず、画面の右端が白い帯になってしまうという問題でした。これを直すパッチは存在したのですが、当然ソース配布なので、ランドスケープモードを使いたかったら GhostScript を自分でコンパイルする

必要がありました。

GhostScript のバージョン 5.03 に戻ればこのバグはないのですが、日本語 pdf ファイルを見るためには 5.10 が必要で、このあたりに悩ましい問題がありました。

結局私は 5.10 を自力でコンパイルしました。その際、良さそうなパッチは全部当ててビルドしたので、コンパイル作業は結構面倒でした。またプリンタに何を使うかで、一緒にコンパイルするプリンタドライバも変わってくるため、コンパイル前の準備がなかなか大変でした。

さて、無事に GhostScript がコンパイルできたとしましょう。しかしここでもまた問題があるのです。それは TEX などで作成した、いろいろな大きさの英字フォントの入った文書の印刷です。TEX では英字フォントは基本的に metafont によって作成されていますが、これがデフォルトではかなり足りないのです。

特にフォントの縮小拡大をした場合、対応するフォントが存在しないと他の拡大率のフォントを使って便宜的に合成されるため、横縞が入ったようなフォントイメージが印刷されてしまいます。

これを避けるためには、pk フォントを大量に作成する必要があります。ただし作成のためのパラメータはあまりにも繁雑で全部手で行なうのは非効率です。そこで Fukui Rei さんの作成した makefont パッケージを使って作成すると便利です。

makefont パッケージは pk フォント作成用のシェルスクリプトの集合体です。これを元にしてしつこくフォントを作成してインストールすれば、とりあえず縞々フォントとはさよならできます。

2.2　通信環境の構築

2.2.1　はじめに

通信環境といえばちょっと昔（'95 年くらいでしたっけ...）まではパソコン通信環境のことでした。しかし今ではパソコン通信は見る影もないほど廃れてしまいました。私もプライベートな BBS を細々と運営していたのですが、98 年の HD クラッシュ後はメンバーも「もういい」などと言って感心を示さなくなったので、運営を終了しました。(余談ですが、仲間の中には全面的に ISDN に移行し、モデムを捨てた者までいました。彼などどうやっても私の BBS には物理的にアクセスできない状況になってしまいました。(苦笑))

最近では本当に IP 接続が幅を利かせています。前述のようなプライベートサービスを提供する場合でも、通信経路としてインターネット経由にしないとユーザーがつきません。逆に言えば、クライアント側もサーバ側も、通信環境としては IP 接続だ

けを整備していればいい状況になっています。

さて、その IP 接続ですが、多くの場合はアナログ回線か ISDN 回線を通じてプロバイダにダイアルアップし、PPP を使って TCP/IP 接続をするというのが現状だと思います。手元の環境がマシン 1 台であれば単純に PPP のクライアントソフトを入れてダイアルアップするだけですが、自分の周りに LAN を構築し、すべてのマシンからインターネット接続を利用しようとすれば LAN とルーターを整備しなければなりません。

2.2.2　ダイアルアップによる PPP 接続

PPP による接続をするためには PPP クライアントをインストールしなければなりません。正確には PPP ではクライアントもホストも等価なのですが、IP アドレスの付与や着信端末の管理などの動作がホスト側には必要なので、敢えてホストとクライアントを分けて考えることにします。

PPP クライアント

クライアントソフトには有名どころとして pppd と PPxP があります。unix の伝統に則ったソフトは pppd の方なのですが、設定と操作がやや面倒です。この pppd は mgetty という端末プログラムと組み合わせることで、ppp サーバにすることもできます。私はかつてはこの pppd をクライアントとして使っていました。しかし常用しているプロバイダが設備を更新した際に繋がらなくなってしまったので PPxP の方に乗り換えました。

PPxP は真鍋敬士さんによる ppp プロトコルの実装で、pppd が ppp カーネルモジュールを使った実装だったのに対して、こちらではトンネルデバイス (userlink モジュール) を利用したユーザ空間での実装になっています。

この PPxP は設定も簡単で、接続性も良好です。設定は ppxp プログラムを起動して、その中から qdial コマンドを実行することで curses ベースのらくちんな環境ですることができます。また設定をした後では、xppxp という GUI による操作コンソールを使ってコマンドのタイプなしで ppp 接続を制御することもできます。

私はデスクトップ、ノートともこの PPxP をクライアントソフトに使っています。特にモバイル環境では、必ずしも両手でタイプできるとは限らないので、GUI による接続は便利だと思います。

この PPxP 用に ToHerat のセリオのキャラクターを使った SSS(Serio's Satellite

Service) というソフトが作られているのをご存知の方もいると思います。これは「初音のないしょ」に含まれる SD 等身のセリオのビットマップを使って、xppxp と同様のサービスを行なうものです。私が試した時は画像ファイルの読み込みがうまくいかず、動きませんでしたが、ソースは公開されているので近々[3]動くようにしてみるつもりです。こういう萌え系のツールがあるのも PPxP の嬉しいところです。

PPP サーバ

私は週日と週末で住んでいるところが違うのですが、週末の住所から週日の住所にあるコンピュータにログインしたいことが度々ありました。そのコンピュータは常時 IP 接続してはいるのですが、IP マスカレードを使った接続のため、電話回線を使ってダイレクトに接続しない限りログインできません。そこで使わなくなっていた pppd をサーバとして甦らせることにしました。

pppd をサーバとして使うためには、端末監視プログラムとして mgetty を使う必要があります。mgetty はシリアルポートを監視し、着信があった場合、適切な処理を行なうプログラムを起動します。単純な mingetty や agetty では login を起動するだけなのですが、mgetty を使うと FAX の受信や、ppp サーバの起動を行なうことができます。

mgetty のコンパイルは難しくありませんが、ppp の着信監視に使う場合、Makefile の中の 110 行目付近にあるコンパイルフラグを指定する必要があります。

```
CFLAGS=-O2 -Wall -pipe -DAUTO_PPP
```

この-DAUTO_PPP というフラグがないと、望んだような機能を果たしません。

mgetty をインストールしたら、次は設定を行ないます。mgetty をインストールすると/etc/mgetty+sendfax というディレクトリができているはずです。この中の login.config に次のエントリを書きます。

```
/AutoPPP/ -      a_ppp   /usr/sbin/pppd auth -chap +pap login debug
```

これは ppp ログインを検知すると pppd を起動し、認証を PAP で行なうことを指定するものです。

次に pppd の設定を行ないます。pppd の設定ファイルは/etc/ppp の下にあります。認証は PAP で行なうと指定しましたので、この中の pap-secrets に ppp 接続のアカウン

[3]コミケの修羅場が終ったら…

トとパスワードを記述します。例としてユーザ名が eimi でパスワードが chansama[4]の時のファイルの中身を示します。

```
# PAP authentication file: /etc/ppp/pap-secrets
# This file should have a permission of 600.
# Username      Server  Password        IP addresses
"eimi"           *      "chansama"      *
```

ここでハマりやすいのが Server と IP address のフィールドで、ここを迂闊に書くと、指定したマシン / アドレス以外から接続できなくなってしまいます。個人で細々とやる分には、上記のようにアスタリスクで any に指定した方がいいでしょう。

また、ppp サーバの動いているマシンをルータにして、LAN やその先のインターネットへでていく場合、options.s を次のように書き、options という名前でシンボリックリンクを張っておきます。

```
-detach
modem
crtscts
lock
192.168.0.7:192.168.0.8
proxyarp
```

2.2.3　それでもパソ通環境が...

前述したように現在ではほとんどインターネット接続が通信と同義語になっているので、昔ながらのパソコン通信用のターミナルソフトはほぼ不要になっています。

しかしモデムの設定をあれこれ変えて実験したり、シリアルポートに直に何か文字列をを送りたい時など、やはりターミナルソフトがあると便利です。

私はこのような用途に xc を使っています。これは Larry Gensch らによる通信ソフトで、もともとは 1byte コード専用でしたが、日本語対応パッチが存在し、これを当ててコンパイルすることで、漢字の表示と送信が可能になっています。

最もよく使うのは、モデムの設定値を見たり変更したりする用途なのですが (^^;、それには

[4]頭悪そうなアカウントだ...（笑）

```
xc -l /dev/modem
```

のようにしてデバイスを指定して xc を実行します。xc の中のプロンプトで t と入力するとターミナルモードになるので、ここで

```
AT&V
```

などのコマンドを実行すると次のようにモデムの設定値が読み出されます。

```
ACTIVE PROFILE:            Japan
B0 B2 E1 M1 N1 T Q0 V1 W0 X3 &C1 &D2 &G0 &K3 &P1 &Q5 &S0
%C3 %U0 \J0 \K5 \N3 -C0
S00:000 S03:013 S04:010 S05:008 S06:004 S07:050 S08:002 S10:020 S11:095
S30:000 S36:007 S37:000 S48:007 S89:060 S95:000

TELEPHONE NUMBERS:
&Z0=
&Z1=

OK
```

このように設定値を確認しながらセッティングを煮詰めていくと便利です。

実際にやってみると「どうやってターミナルモードを終るんだー！」と迷うと思いますので、老婆心ながらチップスを。ctrl-a と入力すると画面表示こそ変わりませんが、実はターミナルモードから抜けています。そこでおもむろに q と入力すると xc が終了します。

本来の用途であるパソ通接続にも使ってみたい気持ちはあるのですが、いまだにこれで BBS に接続したことはありません。(苦笑)

2.3 CG 閲覧環境の整備

やはり「萌え」るほど使える環境と言ったら各種フォーマットの CG が見られなければいけませんね。(^^;

xv

unix で一番ポピュラーな画像閲覧ソフトと言えばやはり xv ということになるでしょうか。このソフトは最後のリビジョンアップが 1994 年 12 月と大変古いのですが、今なお他の後発ソフトと十分対抗して一線級のソフトウェアの地位を保っています。

このソフトは画像加工ソフトとしてはあまり強くありませんが、表示ソフトとしては大変優れています。まず第一に対応フォーマットの種類が大変多く、jpeg, gif, tiff, postscript, pgm, pbm, ppm, xbm, xpm, bmp 等のロードとセーブが可能なため、日常の使用から研究用のべたファイルの確認までかなりのことがこれ 1 つでできます。また、ユーザーによって png フォーマットと日本独自の mag, pic, pi, pic2 などのファイルフォーマットに対応するパッチが作られているため、今後の脱 gif 時代にも過去の萌え CG 閲覧にも対応できます。

2 つめの長所はユーザインタフェースです。このソフトはユーザインタフェースが非常に優れていて、直観的な GUI による操作が実現されています。GUI 自体のデザインも非常に洗練されていて、青を基調とした立体感のあるデザインは GNOME や KDE のウィジェットのデザインと比べても美しいものです。

またソースコードも完全に公開されているので、カスタマイズしたり、実装を参考にすることもできます。私はスライドショーでのショートカットをスペース&バックスペースから拡張して、b でも戻れるようにしました。less などで慣れたショートカットなので、それまでも無意識に押していたのですが、ちゃんと期待通りに動くようになってハッピーです。(^^)

また、各フォーマットの実装も奇麗にモジュール化しているので、画像ファイルフォーマットの取り扱いについて実装から理解するのにも使えるでしょう。私は png フォーマットの圧縮方法を zlib の LZ77 から bzlib のブロックソート法に変更して圧縮率を調べて遊んだりしました。

このソフト、強いて問題点をあげるとすれば、2 つほどあげられます。まず第一にリリース時期が古いため、バリバリ使えるようにするためには結構パッチを当てなければならないということ。私が当てているパッチはプログレッシブ jpeg のための libjpeg 6a を使うパッチ、日本製フォーマットを開くためのパッチ、png フォーマットを扱うためのパッチです。しかし日本製フォーマットのパッチと png のパッチはパッチ対象箇所がバッティングしており、どちらか一方を当てるともう一方はほとんどリジェクトされ、壮大な.rej ファイルができてしまいます。(^^;　私はこれを見ながら日本拡張パッチの方を手で当てました。

もう一つの問題点は、これがシェアウェアということでしょう。現実にはレジストしなくても使えてしまうのですが、いささか厄介ではあります。ライセンス料はちょっ

と高めで 25 ドルです。10 ドル程度なら寄付のつもりでさっさと払ってしまうのですがちょっと高いですね。しかもカードでは払えないという不便さです。

この点を快く思わない人達も当然います。その中には「フリーではない」ということが直接の動機となってソフトウェア開発までしている人もいます。そんな人の作ったソフトが次に紹介する ImageMagick です。

ImageMagick

完全フリーの画像閲覧・変換ツール群です。ImageMagick というアプリケーションがあるわけではなく、display, convert, animate, combine などという小さめのアプリケーションからなるセットの名前をそう呼んでいます。私の記憶では、作者は「xv がフリーではないからこれを作った」とどこかで言っていたはずです。

こちらのプログラムは複数の実行ファイルから構成されているため、xv に比べると見通しが悪く、使い勝手も一見悪いように思えます。実際は慣れるとこちらも案外悪くありません。

例えば画像表示用の display というプログラムでは、xv にできない大きな画像の原寸表示ができますが、大きな画像を開くと自動的にナビゲーション用の小ウィンドウが表示され、中のフレームをマウスで動かすと原寸の画像がリアルタイムでスクロールするなど、よく考えられた作りになっています。

また、xv では gif アニメーションの表示が全くダメで、最初のフレームしか見ることができないのですが、この display を使うと全フレームを見ることができます。

画像変換用のプログラム convert を使えばバッチ処理でディレクトリ中のファイルのフォーマットを変換することもでき、大変便利です。例えばディレクトリの中の png フォーマットのファイル全てを jpeg に変更したい場合、

```
#! /bin/bash

while [ "$1" ]
do
    jpgname=`echo $1 | sed -e s/.png//`.jpg
    convert -quality 80 $1 JPEG:$jpgname
    shift
done
```

のようなスクリプトを書いて png2jpg という名前でセーブし、

```
$ png2jpg *.png
```

と実行すれば後はぼーっとしているだけでコンバートできます。

日頃の CG 閲覧にはとっつきやすい xv を使うとしても、この ImageMagick を入れておくのは悪くない考えです。

2.4　ムービー再生環境の整備

linux 環境でのムービー再生には xanim と mtv をおすすめします。mpeg1 ムービーの再生には mtv を使い、その他のフォーマットには xanim を使うようにします。

mtv (MpegTV)

mtv (http://www.mpegtv.com) はシェアウェアなのですが、再生クオリティはソフトウェア再生としては非常によく、音声もきちんと再生されます。登録料はたったの 10 ドルなのでぜひ登録してみて下さい。ニューバージョンのリリースも頻繁に行なわれています。

ちなみに mpegtv.com の ftp サイトには posix thread library のモディファイバージョンがあったりしてなかなか楽しいです。このライブラリは概ね linux thread 0.71 なのですが、画面描画は同時には 1 スレッドからしか許可しないという改造がなされています。したがって thread ready でない X ライブラリを使用しても競合に関する問題がおきません。まぁ性能向上もあまり見込めないのですが...

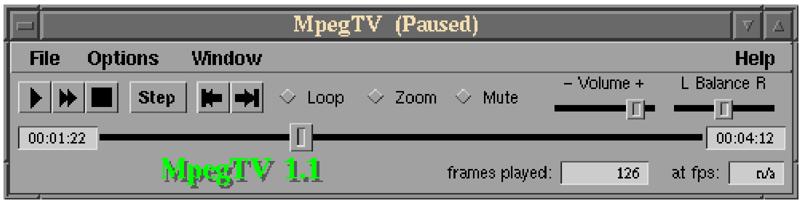

図 2.1: mtv(MpegTV) のコントロールウィンドウ

xanim

xanim (http://xanim.va.pubnix.com/home.html) はフリーウェアで、主に QuickTime ムービーの再生に使います。ムービーの圧縮の世界はなかなか生臭く、具体的な仕様が公開されていなかったりするので、シネパック、CYUV などのコーデックはソースコードが添付されていません。これらは dll として予めコンパイルされたバイナリが xanim の作者によって配布されています。これをインストールして使用します。

最近の QuickTime ムービーはほとんどがソレンセンコーデックなのですが、残念ながらこれは Apple Computer にしかライセンスされておらず、xanim 用のバイナリは配布されていません。したがって最新の QuickTime ムービーの再生に関しては問題があります。

余談ですが、ビデオコーデックは「死ねパック」とか「ソ連船」などと妙な語呂合わせができるものが多く、Free soft ユーザの苦々しい気持ちを表現するのに適していて笑えます。いつか xanim 上でソレンセンコーデックを再生してブラックではなく笑いたいものです。

図 2.2: xanim

2.5　mp3 環境の整備

'99 年現在のパーソナルコンピューティング環境を考えると、mp3 なしのコンピューティングは考えられません。私の感じている mp3 の利点をあげると、

- たくさんの曲を一気に聴ける（3 時間以上連続で再生とか。）
- メディア交換の手間がいらない

- 再生する曲順や曲の組合せをいろいろ用意しておいて、その日の気分で再生することができる
- ノートで再生すると CD などに比べて大いにバッテリーの節約になる
- どうせノートを持ち歩くのだったら、他のオーディオ機器を持ち歩かなくてもよいのはうれしい

などがあります。ノート環境の話は後ほどの章で述べますが、私はいつも移動の時には Thinkpad600E とヘッドホンを持ち歩き、電車の中で作業をしたり日記を書きつつ mp3 を聴いています。作業中ずっと音楽を聴きっぱなしというのは CD ではとてもできない芸当です。

mp3 を合法的に楽しむためには当然エンコーダが必要になります。このセクションでは mp3 環境についてお話します。

2.5.1　デコーダー

mpg123

Linux のみならず unix 環境で最も広く利用されているデコーダが、Michael Hipp による mpg123 (http://www-ti.informatik.uni-tuebingen.de/~hippm/mpg123.html) です。これはマシンへの負荷が軽く、再生もかなり優れた音質で行なうことのできるソフトです。この再生コアを利用して GUI をかぶせたデコーダーも数多くリリースされています。

mpg123 はコマンドラインさえあれば利用可能なので、曲の中でサーチなどを使わなければ起動も非常に軽く快適です。後ほど述べる FD clone を使ってローンチする場合のプレイヤーとしても最適です。

tk3play

tk3play (http://www.msc.cornell.edu/~bef2/) は mpg123 のコアを使用して、wishx, tclx による皮をかぶせた GUI 再生ソフトウェアです。プレイリストをセーブできるため、テーマ別に聴きたい時などはこれです。電車の中で聴く時も途中で操作がいらないのでこちらが便利です。

tk3play のプレイリストはテキストファイルなので、エディタで細工をすることも簡単です。

また、曲のスキップやダイレクト選曲、早送り・巻戻しなどもでき、イイ感じです。

この tk3play の特長は、tcl で書かれたスクリプトなのでカスタマイズがしやすいことと動く環境をあまり選ばないという点です。高機能な GUI 再生ソフトは特定のライブラリに強く依存していたり、thread aware な X ライブラリがないとまともに動かなかったりするのですが、この tk3play はその辺りの難しいことがないのが気持ちいいです。

ただ若干の問題もあって、エンジンとなっている mpg123 のバージョンが古く、windows 環境でエンコードされた joint stereo フォーマットなどでは、再生音質が悪い場合があります。このあたりは最新版とマージをするなどの対策をした方がよいでしょう。

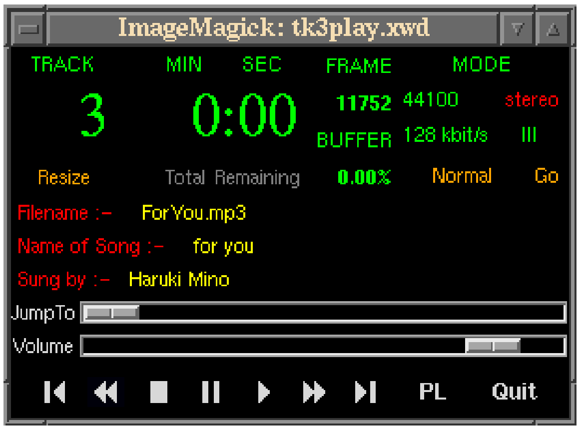

図 2.3: tk3play

2.5.2 エンコーダ

さて、楽しいエンコーダの話になります。エンコーダは速度と音質という多くの場合背反する要求を満たさなければならず、いじり回すのはかなり奥が深いお楽しみになります。

iso 標準エンコーダ

'97 年に mp3 を使い始めた頃は iso のサンプルエンコーダをダウンロードしてエンコードを行なっていました。

今考えるとこのエンコーダは非常に遅く、音質も高域が上がる癖があって、性能の悪いエンコーダでした。入力も AIFF というマックの世界でポピュラーな形式を使っており、リッパーの出力がほとんど wav な AT の世界では変換をしなければならず、使い勝手が良くありませんでした。

自分でデフォルトオプションを変更したりちょこっと高速化をしましたが、あまり目覚しい成果は出ませんでした。そうこうしているうちに次の BladeEnc を知ったので、このエンコーダは使わなくなってしまいました。

それでも後で紹介する lame や午後のコ～ダはこのソースを元に派生していったエンコーダで、存在意義や歴史的な価値は非常にあります。(笑)でも、もう使う必要はないでしょう。

BladeEnc

BladeEnc (http://hem.bredband.net/tord) はスウェーデン[5]製のエンコーダです。これは iso に比べるとかなり高速になっていて、またヘッドホンで聞いてみるとかなり iso エンコーダよりも音質が向上していました。このエンコーダはかなり長く使っていました。

1999 年の 1 月頃ソースが公開され、しばらくいじくり回して楽しませていただきました。最近の開発状況がどうなっているのかはわからないのですが、もしかするとかなり改良されているかもしれません。

lame

lame (http://www.sulaco.org/mp3/) は iso のサンプルエンコーダへのパッチとして始まりました。lame ではエンコード速度の大幅な改善と音質の改善がなされました。一度使ってみたところ、BladeEnc には戻れなくなってしまいました。

このエンコーダは最初からソースが公開されていたので、デフォルトオプションを変えたりしてかなり使い込みました。現在でも音質・速度について精力的に開発が続けられているようです。特に可変ビットレート圧縮ではフリーソフトとしては先端を行っているはずです。

[5]なんでもスウェーデンではソフトウェアやアルゴリズムでは特許がとれない制度であるそうな。素晴らしいです。

午後のこ～だ

午後のこ～だ (http://www.kurims.kyoto-u.ac.jp/~shigeo/) は京都大学の光成 滋生さんらによるエンコーダです。このエンコーダは FFT などでアセンブラコードを用いて CPU の特殊機能を引き出しているのが特徴で、MMX, SSE, 3D NOW などの拡張機能のある CPU を使うと信じられないような高速でエンコードしてくれます。

ドキュメントには音質は若干低いような記述がありますが、ヘッドホンで聴いても lame に遜色ない音質でした。

このエンコーダを使うと、128bit/sec stereo, psycological model ありの状態で私の PentiumII 300MHz は元データの演奏時間の 1/3 程度の時間でエンコードを終了します。ver 2.2.2 からはマルチスレッドを使い、SMP マシンでは n 台の CPU を使って、同じ CPU1 台の場合に比べて $1/n$ に近い時間でエンコードできるようになりました。ちなみに私のデュアル penII300MHz マシンでは演奏時間の 1/6 程度の 時間でエンコードすることができました。

性能を総合的に考えると、現時点ではこの午後のこ～だが一番よいエンコーダでしょう。

私の保有するマシンではテストできないのですが、PentiumIII の SSE 命令を使うと異常なまでに高速化するそうです。(^^)Pentium III を SMP マシンで使ったらすごい性能が出そうです。

2.5.3 リッパー

CD からデジタルデータを吸い出し、wav ファイルを作ってくれるのがリッパーです。数種類があるようですが、私の使ったことのあるリッパーは cdda2wav と cdparanoia の 2 種類です。

cdda2wav

cdda2wav (ftp://ftp.gwdg.de/pub/linux/misc/cdda2wav/) は linux におけるリッパーの原点です。SCSI マシンでは General SCSI が使えるようにカーネルをコンパイルする必要があったり、そこそこ設定には苦労しました。

cdda2wav では、ジッターなどのノイズが生じるか否かはほとんどドライブ性能任せです。CPU やバスの負荷が高い状態で使うと取りこぼしが生じることがあるので、リッピング中はあまり他のタスクを実行しない方がいいでしょう。ソースの中にリアルタイムスケジューリングクラスに設定する部分やプライオリティーを上げるコード

があるのですが、実際にはほとんど効果はありませんでした。

複数トラックを一気に取り込むと、トラック境界で別れたファイル群が生成されます。CD ドラマなどでこれをやられるとなかなか悲しいことになります。(苦笑)

次に述べる cdparanoia はこの cdda2wav を元にして作られたそうです。

cdparanoia

cdparanoia (http://www.xiph.org/paranoia/) は非常に強力で、取り込んだストリームデータに問題があれば、その部分の修復をしてくれます。このあたりが「パラノイア」と自称する所以でしょう。

私の使い方では、エラーの修復はほとんどしませんが、充実したコマンドラインオプションによって、トラック境界の取り扱いや取り込み時間の取り扱いなどを制御することができるので、大変重宝しています。

cdda2wav で問題になるような複数トラックの 1 ファイルへの一括リッピングもコマンドラインから一発で指定できます。(もちろん cdda2wav のように 1 トラックずつファイルにすることもできます。)

2.6 MIDI ファイルの再生

私は昔 Macintosh を使っていましたが、SC-88 などの音源を持っていなかったため、MIDI ファイルの忠実な再生は大変困難でした。当時は QuickTime 2.0 などのソフトウェアシンセサイザーで GM 音源のエミュレートを行なって MIDI ファイルを再生していました。

Linux を使い始めてからは SoundBlaster AWE64Gold を使って、ウェーブテーブルシンセサイザーで MIDI ファイルを再生するようになり、格段に音質と忠実性が向上しました。しかしちょうどこの頃を境に、CPU の処理能力が劇的に向上し、ソフトウェアシンセサイザーの方が高いクオリティを示すようになってきました。

ウェーブテーブルシンセサイザーは音一つ一つのクオリティはやはり今でも優れていると思うのですが、カード毎に違うハードウェア構成に合わせたチューニングを煮詰めていかない限り、専用音源のエミュレートは困難です。一方、ソフトウェアによる波形合成は、カードの能力としては wav データの再生さえできればＯＫで、チューニングは専ら wav にする前の段階で行なわれます。このため、カードに依存しないチューニングや煮詰めという作業を行なうことができ、開発リソースが集中できる利点があります。

TiMidity++

今日では、出雲 正尚さんらによる TiMidity++ [6](http://www.goice.co.jp/member/mo/timidity/) というソフトウェアシンセサイザーがかなり高い完成度に達しており、ハードウェアによるウェーブテーブルシンセサイザーのクオリティを凌駕しています。このソフトを使うと SC-88 や SC-88Pro、あるいは XG 音源といった広範囲の音源の専用データを多くの場合違和感なく再生できます。

TiMidity++では、音データとして、Gravis Ultra Sound 用の "patch" ファイルを使います。これはネットワーク上にいろいろなサイズ・クオリティのものが存在しており、好みとニーズに合わせて選択できます。私のおすすめは 10M バイト程度のパッチファイルです。

たぶん TiMidity のインストールで一番やっかいなのがこの patch ファイルのインストールでしょう。基本的にはファイルを/usr/lib/timidity/inst とか/usr/local/lib/timidity/inst に置いて、音色のマップ等について、その上のディレクトリの設定ファイルを編集していけばいいのですが、この作業は凝りだすとキリがないので、仕事や論文などの〆切前にやるのは大変危険といえます。(^^) だいたい忙しい時に限って無性に音楽が聴きたくなったりするのが人の性なのですが、自重しましょう。(自戒)

TiMidity++では wrd ファイルの表示もできます。しかし、この機能はマシンによっては同期がとれなかったりして、まだまだという感じです。肝心の音楽の方はほぼ完成の域に達していると思うので、今後の開発に期待します。(自分でもいじってみたいけど...)

2.7 ファイルの取り扱い

ファイルを効率的に取り扱うためには、もちろんシェルを使いスクリプトを書き、unix のパワーをフルに発揮させるのが正しい生き方です。(^^) しかし人生、そんなにパワーを必要としない局面も結構あります。そんな時はファイルメンテナンスソフトを使って楽をするのもいいと思います。

FD clone

白井 隆さんによるソフトウェア FD clone はとても使いやすいファイルメンテナンスソフトです。(ftp://ftp.ics.es.osaka-u.ac.jp/pub/FDclone) DOS を使っていた人には FD というソフトに馴染みがあるかと思います。この FD clone は、DOS の FD を

[6]原型である TiMidity は Tuukka Toivonen 氏らの作品です。

unix 上で再現することをコンセプトに作成されました。ま、そういうわけで馴染みのある人には釈迦に説法だと思いますが、一応便利な使い方についてふれておきます。

個人用のマシンでは、サイトワイドのデフォルト設定ファイルをなんとかしておいた方がハッピーになれます。fd はまず/etc/fdrc を参照してから各ユーザのホームディレクトリの.fdrc を参照します。もしこの中に設定があって、自分で定義した.fdrc の中に同じ項目の設定がない場合、文字列のちょっとしたマッチングの差異で自分の予想しなかったふるまいが発生します。大抵このような場合は/etc/fdrc があること自体を忘れているので、大変混乱します。(経験談)

また、FD では拡張子にマッチさせてアプリケーションを起動できますが、この機能を目一杯活用する場合、デフォルトの MAXLAUNCHTABLE の数 32 は全くもって足りません。ソースの fd.h の中のエントリを探して 64 なり 128 なり、大きな数に変更しましょう。

以下は私の.fdrc にある launch table です。

```
launch ".jpg"   "xv %C &"
launch ".JPG"   "xv %C &"
launch ".jpeg"  "xv %C &"
launch ".JPEG"  "xv %C &"
launch ".gif"   "xv %C &"
launch ".GIF"   "xv %C &"
launch ".bmp"   "xv %C &"
launch ".BMP"   "xv %C &"
launch ".png"   "xv %C &"
launch ".PNG"   "xv %C &"
launch ".pi"    "xv %C &"
launch ".PI"    "xv %C &"
launch ".mag"   "xv %C &"
launch ".MAG"   "xv %C &"
launch ".pic"   "xv %C &"
launch ".PIC"   "xv %C &"
launch ".pcx"   "xv %C &"
launch ".PCX"   "xv %C &"

launch  ".mid.gz"       "timidity -A 100 -B 500 -in %C"
launch ".ps"    "gv %C &"
launch ".eps"    "gv %C &"
launch ".dvi"   "xdvi %C &"
launch ".mid"   "timidity -A 100 -B 500 -in %C"
launch ".MID"   "timidity -A 100 -B 500 -in %C"
launch ".rcp"   "timidity -A 100 -B 500 -in %C"
launch ".mpg"   "mtv %C &"
launch ".MPG"   "mtv %C &"
launch ".mov"   "xanim %C &"
```

```
launch ".MOV"   "xanim %C &"
launch ".mp3"   "mpg123 -b 1024 %C"
launch ".MP3"   "mpg123 -b 1024 %C"
launch ".html"  "lynx %C"
launch ".htm"   "lynx %C"
launch ".wav"   "wplay %C"
launch ".WAV"   "wplay %C"
launch ".AVI"   "xanim %C &"
launch ".avi"   "xanim %C &"

launch ".pdf"   "gv %C &"
launch ".PDF"   "gv %C &"

launch ".pak"   "lfview -a %C"
launch ".euc.gz" "less %C"
```

第3章　萌えるマシンとはとんがったマシンだ！

他のふつーの人のマシンとは一味違うマシンを使ったら、毎日がハッピーだとは思いませんか？　ましてやそれがかなりじゃじゃ馬で、使いこなしに高いスキルと業が必要だとしたら、そして使いこなしていることで周囲の人からそんK されるとしたら、チャレンジしてみたくはなりませんか？

この章ではそんなあなたにぴったりのモンスターマシン[1]をいくつか紹介してみます。はっきり言って私はこういう妖しいマシンを見ると血がたぎるのが抑えられません。(小学生の頃読んだ松本零士先生のマンガ[2]に松本先生の似たようなメッセージが書いてあったな。(笑))

3.1　SMP マシン

3.1.1　SMPって、何？

Symmetric Multi Processing, 日本語でいえば “対称型マルチプロセッシング” — なんか日本語で言ったのは最初だけのような気がしますが — これを縮めて SMP といいます。よくマザーボードで CPU が 2 つ取り付けられる” デュアル” [3]タイプというのがありますが、あれでできるマルチプロセッシング処理が SMP 用マザーボードです。

この手のマシンは本来はサーバ用で、「負荷が異様に高い場合でも負けずに処理をがんばる」というコンセプトで設計されています。CPU を 2 個、あるいは 4 個載せたところで普通のタスクは少しも速くなりません。コンスタントに恩恵を受けることができるのはウェブサーバとかファイルサーバに使った時で、同時に多数の要求があってもそれほど速度低下することなく処理が続けられるというメリットがあります。

[1]大げさかなぁ...
[2]潜水艦スーパー 99 だったかな。
[3]三月さん萌え。

（まぁCPU 以前にディスクなどで処理がバッティングしないというのが必要ですが。大雑把に言うとメモリを大量に積んでキャッシュするようにすればこのボトルネックを解消できます。）

しかし、やはり我々萌え系ユーザーとしては、「CPU が 2 個！[4]」という妖しい響きにはもうそれだけでたまらない魅力があります。漫然と使うと 2 個の CPU は速度低下を抑える程度にしか働きません。2 個の CPU を高速化に使うためにはマルチスレッドプログラミングを行なって、タスクの中に並列性を持ち込まなければなりません。そしてマルチスレッドやマルチプロセッサにすると、アクセスの競合が発生するため、資源のロックとアンロックの問題が浮上してきます。

このような悪条件を越えてマルチプロセッシングの世界に踏み込もうとするのは、「萌えるから」に他なりません。（笑）

3.1.2　何に向いているの？

では、個人ユーザで SMP マシンを使うとどんなメリットがあるのでしょうか？一つは、何かの思いタスクを走らせていながらでもある程度の反応速度が得られるということがあります。シングルプロセッサのマシンではコンパイル中の対話操作は若干遅くなりますが、デュアル、クアッド等のマシンではほとんど遅くなりません。特に数値計算などをやりながらの対話操作が遅くならないのはうれしいところです。

もっと実用性の高い効果としては、コンパイル速度の向上があります。make -j3 などとタイプして同時に複数のジョブを起動すると、ソースの構造にもよりますが、プロセッサをフルに使って並行コンパイルをしてくれます。

また mp3 のエンコードでも SMP マシンは役に立ちます。普通のエンコーダを使っても、スクリプトを書いて逐次投入されていくようにすれば、同時に 2 曲のエンコードができます。2 つの CPU をフルに活用してエンコードできるので、複数の曲をエンコードする場合は総エンコード時間を約半分にすることができます。

最近リリースされた午後のこ～だ 2.2.2 以降を使うと、CPU 数分の実行スレッドを生成して一曲のエンコードを並列処理するため、一曲のエンコード時間を半分・4 分の 1 などに短縮することができます。

rc5des[5]などの暗号解読クライアントもマルチスレッドをサポートしています。暗号ブロック間では処理に相互依存性がないため独立に計算を行なうことができ、プロセッサの台数に応じて処理能力が線形に増加します。私自身、2 つ目の PentiumII を買う予定を前倒しした強い動機となったのは rc5 のスコアだったりします。（笑）

---

[4]エンゼルハートが 2 個のカリンカはリニアな台数効果が出てますね。(^^)

[5]今は dnetc というクライアントになっており、現在 CSC 暗号を解読中です。

3.1.3 SMP カーネル

2.0 系の頃の SMP カーネル作成は configure だけではできませんでした。Makefile の一部を修正してコンパイルする必要がありました。2.0 系では、カーネルコードをジャイアントロックしていて、カーネルコードは 1 つのタスクからしか実行できなかったため、システムコールを同時発行することができず、パフォーマンスが上がりませんでした。

2.1 系でより細かい資源ロックとより高性能な SMP 実装が開発され、2.0 での問題点は解消されました。現在一般向けにリリースされている 2.2 系は 2.1 系で開発された SMP 実装を内包しています。こちらでは SMP の設定は簡単で、カーネルのコンパイル時に SMP サポートをオンにするだけです。

Linux の SMP では、プロセスを特定のプロセッサに固定する afinity の機能は提供されていません。しかしこれは 98 年末頃 SMP の ML で話題になり、簡便ではありますが、類似の機能が実装されました。この方法では、プロセスの動作するプロセッサを決定する際に、もしプロセッサの移動を伴う場合は若干のペナルティを加味します。これによりプロセスがそれまで動いていたプロセッサに再投入される確率が上がります。

プロセッサの移動がない方がパフォーマンスが上がるというのは、主にキャッシュの再充填を回避できるという事実によります。

3.2 SCSI マシン

最近では IDE のハードディスクがあまりにも安くなりすぎたので、SCSI マシンを組む気にはなりにくくなっています。しかし 5 台以上のディスクを付けたいとか、何人もでログインして使いたいとか、あるいは SCSI じゃなきゃイヤだとかの理由で SCSI オンリーのマシンを組む人もいると思います。

ブート

このようなマシンではブートにちょっと工夫が必要です。ブートセクタの読み込みとカーネルのスタートアップは SCSI カードの SCSI BIOS 側からできますが、カーネルが SCSI カードをイニシャライズする時、「初期化コード(ドライバ)はこれから初期化するデバイスの先にある」という鶏・卵問題になってしまいます。

これを解決するため、initrd という RAM ディスクを使います。initrd は libc, ld, insmod, そして SCSI ドライバモジュールを詰め込んだファイルシステムイメージに

なっていて、SCSI カードの初期化に先立ってメモリ内に読み込まれます。カーネルによる SCSI カードの初期化は、この仮想ドライブのファイルを使って行なわれます。Linux カーネルによる SCSI デバイスの初期化が終ると、RAM ディスクは解放され、以後はディスクにあるファイルを使って起動処理を続けます。

Linux ディストリビューションでは普通、数多くの SCSI カードに対応するため、SCSI カードのドライバはモジュールとしてコンパイルされています。この場合、上記のように必ず initrd を使う必要があります。

もし自分でカーネルをコンパイルするなら（するよね？(^^)）、SCSI カードのドライバをカーネルの中に組み込んでカーネルを作れば initrd なしでも起動できます。その場合、必ずカーネルコンパイルオプションで「initrd を使用しない」という設定をして下さい。また/etc/lilo.conf の方も新しいエントリには initrd=ほにゃららという記述を書かないようにして下さい。

SCSI ディスクと IDE ディスクの併用

私のメインマシンも数カ月前までは SCSI オンリーマシンでした。しかし 4G の UW SCSI だけでは、あまりにも手狭だったため、データ領域として IDE のハードディスクを増設しました。

このような混在環境では一つ問題があります。それは BIOS で IDE の方が先に検出されてしまい、ディスク番号 0x80 を割り当てられてしまうため、ブートドライブとして IDE の方を見に行ってしまうのです。

これを避けるためには lilo.conf に次のように記述します。

```
disk=/dev/sda
        bios=0x80
disk=/dev/hda
        bios=0x81
```

sda が ID の一番小さい SCSI ディスク、hda がプライマリマスタの IDE ディスクです。こうすれば、SCSI ドライブからシステムを起動することができます。

3.3　ノートマシン

3.3.1　どうしてノートマシンがとんがってるの？

このセクションを読まれた方は「なんでノートマシンがとんがっているの？」と疑問をもたれたことと思います。確かにノートに Linux を入れるのはさほど特殊なことではありません。

しかしノートマシンの場合、ハードウェア構成が機種毎にかなり異なっており、設定の追い込みが難しく、環境構築をする人によってかなり結果が異なります。

このような背景から、ノートマシンを限界まで活用するのは「とんがった行為」だと考え、この章にこのセクションを置きました。

3.3.2　Thinkpad 600E を味わい尽くす！

私はこれまで Thinkpad600 と 600E を使ってきました。[6]これらの 2 機種についてはかなり細いところまで設定して使い切っていると思います。

そこで 600, 600E について、サウンド、BIOS(APM)、PC カードの 3 点についてお話しようと思います。600 系固有の話が多いですが、他機種をお使いの方にもなんらかの参考になるのではないかと思います。

3.3.3　サウンド

Thinkpad 600 ではサウンドチップに Crystal Semiconductor の CS4237B を、600E では CS4239 を使用しています。これらはコアに共通の CS4232 コアを使った兄弟チップで、基本的には CS4232 互換です。

このチップを動作させるためのドライバには、カーネル附属の OSS Free の CS4232 ドライバと、ALSA(Advanced Linux Sound Architecture, http://alsa.jcu.cz/) の CS4236 ドライバがあります。

カーネルのインストールと同時にインストールできる OSS Free の方が手軽ですが、ミキサーのサポートやチップの性能の引き出し方では ALSA ドライバの方に軍配が上がります。

[6]600E はモデル末期の激安品を買いました。

OSS Free を使う

まず先にカーネル附属の OSS Free について説明します。

2.2 系のカーネルでのインストール自体は簡単で、カーネルの設定時に CS4232 および OPL3 のサポートをオンにして CS4232 ドライバをモジュールとしてコンパイルし、make modules_install するだけです。

モジュールとしてコンパイルしなければならない理由はこうです。Thinkpad 600 ではマシンがサスペンドした時にサウンドチップの電源が切られます。電源復帰後にサウンドチップを使用するためにはもう一度サウンドチップの初期化をしなければならないのですが、カーネルにビルトインで作ってしまうと初期化を行なわないため、チップの挙動がおかしくなってしまいます。

モジュールにしてある場合はこれを再びロードすれば再初期化が行なわれ、チップが正しく利用できるようになります。

モジュールとしてインストールした場合、問題は/etc/conf.modules の書き方になります。Linux ではカーネルのモジュールサポートによって必要な時に必要なモジュールがロードされるようになっていますが、その柔軟性が仇となり、conf.modules の書き方がかなりトリッキーになっています。

OSS Free のモジュールを使うための conf.modules の記述は次のようになります。

```
post-install sound /sbin/modprobe "-k" "cs4232"
options cs4232 io=0x530 irq=5 dma=1 dma2=0 #mpu_io=0x330
options opl3 io=0x330     # FM synthesizer
```

カーネルでは、サウンドが必要になると soundcore, sound-low, sound の各モジュールをロードします。この時 sound モジュールがロードされた後に cs4232 がロードされるようにするのが 1 行目の設定です。

2 行目は CS4232 の使用する io アドレス、irq, dma チャネルのパラメータです。

3 行目は FM シンセサイザーの io アドレスを指定するものなのですが、FM シンセは使ったことがないのでちゃんと動作しているかは謎です。(^^;

以上の設定で PCM に関してはちゃんと音が鳴るようになると思います。しかし OSS Free のドライバはミキサーの動作が怪しい、サラウンド機能のサポートがないなどの問題点があり、使っていてもあまり面白くありません。

ALSA を使う

もう一つのオープンソースサウンドドライバである ALSA は、OSS の古い構造に因われず、高性能で負荷の小さい実装を目指しています。

このALSAの特長は、新しいだけにOSSよりも細かくサウンドチップを操作できる点です。CS4232系のサウンドチップはCS4232を始めとして、4235, 4237B, 4238, 4239までサポートしています。また、CS4232系の特徴である3Dサラウンドもサポートしていて、この効果も楽しめます。もっとも、4237BのSRSサラウンドは不自然であまり嬉しくありませんが...。

このALSAですが、小刻みなバージョンアップを繰り返していて、原稿執筆時点では0.4.1eが最新版となっています。しかし0.3.0-pre4からはミキサー関係のアーキテクチャが変更となったようで、Thinkpadでいまひとつ納得のいく動作をさせることができません。ここでは0.3.0-pre3について説明します。

コンパイルとインストール ALSAのインストールは、カーネルをインストールした後に行ないます。configureでカーネルがらみの情報を収集しますが、ここでサクッと一発で決めるためです。古いカーネルの動作中にコンパイルする方法もあるかも知れませんが、面倒なのでブート後にビルドすることをおすすめします。

お決まりのインストール方法でドライバのインストールは完了します。しかしそれだけではまだ使えません。alsa-libとalsa-utilsのコンパイルが必要です。alsa-libはalsaインターフェースを使うための関数群を提供するライブラリです。alsa-utilsの方はプレイヤーやレコーダーも入っていますが、とりわけ重要なのがミキサー関係のプログラムです。alsaはデフォルトの音量がゼロなので、ドライバのロード時にこれらのミキサーツールを自動で起動して適切な音量にセットする必要があります。

ドライバのインストールに続いて、これらの周辺ソフトウェアのインストールも行ないます。特にここでalsaモジュールを自動的にロードするため、alsa-libの中のalsa-lib-0.3.0pre3a/src/pcm.cにパッチを当てます。

Linuxは/dev以下のデバイスにアクセスがあった時にモジュールのロードを行なっています。もし起動したサウンドアプリケーションがalsaネイティブのインターフェースを使っていた場合、アクセスは/dev以下のデバイスへ行かず、/proc以下のalsaデバイスへ行きます。この場合、linuxカーネルはモジュールをロードしてくれないため、alsaネイティブのアプリケーションからはデバイスのオートロードが利用できないことになります。

これを避けるため、alsa-lib内のalsaネイティブルーチンに細工をします。pcm.cの50行目、snd_pcm_open()という関数の中に

```
{
int fake_fd;
fake_fd=open("/dev/dsp3",O_RDONLY);
```

```
close(fake_fd);
}
```

を追加して下さい。この関数は alsa ネイティブのデバイスオープン関数なのですが、入口で OSS 互換のデバイスオープンを行ない、即閉じることで Linux カーネルを騙し、alsa サウンドモジュールをロードさせます。

　このパッチを当てれば、最初から/proc 以下にアクセスに行く alsa ネイティブアプリケーションを使ってもモジュールのロードが問題なくできます。

設定　この alsa もモジュールの設定は/etc/conf.modules で行ないます。そしてこの部分は OSS Free に増して柔軟に — ということは複雑に — なっています。

　私の conf.modules を以下に示します。

```
alias snd-card-0 snd-card-cs4236
options snd-card-cs4236 snd_port=0x530 snd_cport=0x538 snd_irq=5 \
 snd_dma1=1 snd_dma2=0 snd_fm_port=0x388 snd_dma1_size=64 snd_dma2_size=64

alias char-major-14 snd-card-cs4236
options snd snd_major=14 snd_cards_limit=1
alias snd-minor-oss-0 snd-card-cs4236
alias snd-minor-oss-1 snd-opl3
alias snd-minor-oss-2 snd-midi
alias snd-minor-oss-3 snd-pcm1-oss
alias snd-minor-oss-4 snd-pcm1-oss
alias snd-minor-oss-5 snd-pcm1-oss
alias snd-minor-oss-12 snd-pcm1

post-install snd-card-cs4236 /usr/bin/amixer -p /etc/amixerrc -r
```

　2 行目は紙面の都合で折り返しています。

　1 行目は alsa の複数枚のサウンドカードを扱えるサウンドアーキテクチャのための設定です。1 枚目のカードとして CS4236 を定義しています。

　2 行目が CS4236 の初期化オプションです。使用する io ポート、irq、DMA などの設定をします。これは Thinkpad のステップアップマニュアルの値を参考に決めました。

第 2 段落はデバイスのメジャー番号、マイナー番号と alsa のモジュールを関連付ける記述です。これはまぁ見たままでしょう。(笑)

最後の段落の post-install の部分、これがミキサー設定を自動的に行なうための記述です。意味は snd-card-cs4236 というモジュールが読み込まれたら amixer を実行せよという意味です。amixerrc という設定ファイルを用意しておき、これを参照してミキサーを設定するように指示しています。

このファイルを作成するには次のようにします。

1. alsamixer を起動し、各デバイスの音量を好みに設定する
2. amixer -w を実行する。現在のミキサーの設定値がホームディレクトリに. amixerrc という名称でダンプされる
3. .amixerrc を/etc に amixerrc という名前でコピーする

これで alsa のサウンドモジュールがロードされると自動的にミキサーの音量が設定されます。

新しい alsa では、ミキサー周りの実装が変わっているのですが、周辺のプログラムがまだ整備されておらず、このような使い方ができませんでした。

蘊蓄　Thinkpad 600 と 600E を使ってみてわかったのですが、600 ではサウンドデバイスのイニシャライズ時に必ず「プチッ」という大きな音が入ります。これが 600E では入らないのです。密かに改良されているようです。

また、600E には PCI バス接続で CS4610 というサウンドアクセラレータが搭載されています。これは主にドルビーサラウンドなどの多チャンネルオーディオを実現するために搭載されているようです。DVD モデルなどで必要なのでしょう。DVD ドライブのない 600E でこのデバイスをどう使えばいいのかちょっとまだわからないのですが、デバイスのデータシートは公開されているので、ドライバを作成することは可能だと思います。

3.3.4 BIOS の設定

Thinkpad 600 では、リブート時に F1 キーを押し続けることで EasySetup という BIOS 設定プログラムが立ち上がります。EasySetup で設定できるのは起動時のドライブ検索順や時計などのみです。互換機のような細かい設定を行なうには DOS コマンドである PS2.EXE を使います。

この PS2.EXE を使うとモデムやサウンド、各種ポートの使用・不使用および使

用時のリソース割り当てなどを変更できます。ただしこれに先立って EasySetup で QuickBoot をオフにして下さい。これがオンに設定されていると、Thinkpad は各種リソースを BIOS によって初期化せず、plug and play OS に渡してしまいます。Linux では plug and play 機能は使えませんので、BIOS で各デバイスの設定を行なう必要があります。

さて、毎回 BIOS の設定を変えるためだけに Windows をブートするのは面倒です。幸い linux 用のプログラムに tpctl (http://jhunix.hcf.jhu.edu/~thood/tpctlhome.htm) という PS2 の代わりになるようなソフトがあり、これを使えばかなりの部分を Linux の中から設定できるようになります。

tpctl の主要部分はカーネルモジュールとして実装されています。これはハードウェアに直に触る必要があるためです。余談になりますが、私も IBM のサイトで Thinkpad 600 のテクニカルリファレンスマニュアルを見たとき、このようなカーネル設定用のツールを作りたくなりました。その時どのようにしてハードウェアリソースに触るかを考えましたが、結局カーネルのモジュールとして実装するのが一番良さそうだと考えました。tpctl はまさにそのアプローチを取っています。

3.3.5　PC カード

Thinkpad600 シリーズにはマザーボードに Ethernet をもつモデルはなく、また内蔵モデムも Linux からは使えないため、通信・ネットワーク関係を使うためには PC カードの使用が必須となります。この使いこなしもかなり挑戦しがいのあるテーマです。

pc カードを使うためには David Hinds らによる pcmcia-cs パッケージ (http://pcmcia.sourceforge.org) が必要です。以前は実質的に PCMCIA 規格のカードしか使いものになりませんでしたが、最近では CardBus 規格のカードも実用的に使えるようになってきています。

最近では 3.0.x 系のリリースは終了し、3.1.x 系のリリースが行なわれています。CardBus カードを使う場合、こちらの 3.1.x 系のリリースを使う方が良いと思います。

最近のリリースでは概ねマニュアル通りのインストールで動作すると思いますが、機種毎に綿密に設定しなければいけない部分が/etc/pcmcia/config.opts 内にあります。それは PC カードコントローラのメモリウィンドウと IRQ に関する設定で、これは機種および使用しているコントローラによって微妙に異なるので要注意です。

例えばメモリウィンドウと io アドレスに関する設定では、600 の場合

```
#for Thinkpad600
```

```
include memory 0xc0000-0xfffff, memory 0xa0000000-0xa0ffffff
exclude port 0x230-0x233
```

ですが、姉妹機である 600E では

```
#for Thinkpad600E
include memory 0xc0000-0xfffff, memory 0x60000000-0x60ffffff
exclude port 0x230-0x233, port 0x2f8-0x2ff
```

となります。特に上位メモリウィンドウの位置が全然違うあたりが要注意です。

また、IRQ に関しては、PC カード関係のデバイスに使わせたくない IRQ を列挙することになります。私はかなりくどく除外をかけていて、

```
# First built-in serial port
exclude irq 4
# Second built-in serial port
exclude irq 3
# First built-in parallel port
exclude irq 7
# CS423x sound chip
exclude irq 5
# unknown
exclude irq 6
# rtc
exclude irq 8
```

のように記述しています。こうするとほぼ IRQ=9 に固定となります。

PC カードインプレッション

以下では私の使ったカードについて感想を述べたいと思います。

Correga PCC-T PCMCIA 規格の 10Base-T カードです。このカードは値段も安く性能も期待通りでいいカードだと思います。使用チップはよくわかりませんが、NE2000 互換のチップのようでした。

ただしこのカードを使う場合、若干configファイルに修正が必要です。このカードのタプルはConfigファイルの中にあるAllied Telesyn AT-2800 10/100 Fast Ethernetとコンフリクトしてしまいます。これを修正するには/etc/pcmcia/configのAllied Telesynの部分をコメントアウトします。

Linuxだけを使っている場合はカードは1枚で十分ですが、Windowsはタコなので、家と会社等でアドレスを変える場合、いちいちリブートする必要があります。これを回避するためにタプルの違うカードを用意し、使用場所に置きっぱなしにして差し替えて使うというテクニックがあります。アドレスはそれぞれのカードに付くため、PCカードを差し替えれば複数のアドレスをブートせずに使うことができるのです。しかしこれまたWindowsでは、ブートせずに複数枚のCardBusカードを差し替えて使うことができません。そこで低速環境用に10BaseのPCMCIAカードを用意しておくと便利です。高速ネットワーク環境のあるところではCardBusカードを使い、低速な環境しかない場所ではPCMCIAカードを挿して凌ぐという方法が良いと思います。

Planex FNW3600　このカードはpcmcia規格の100Base-TXカードです。このカードはLinuxでの動作が保証されています。確かに設定や動作には問題がないのですが、性能には問題があります。このようなカードは100Base環境に10BaseのNICが混在することでネットワーク全体が10baseにフォールバックするのを避けるために売られているカードです。つまり実質10Baseなのに、電気的に100Baseのフリをしているだけです。

100Mbit/secのFast Ethernetを使うにはISAバスやそれをベースとするPCMCIAバスでは帯域が足りないのですが、バス帯域をフルに使い切っているわけではなく、実質的に転送レートは10Mbit/secのカードそのものでした。このカードはおすすめしません。

Planex FNW3601　同じPlanex社から発売されている100Base-TXカードですが、こちらはCardBus規格に準拠して製造されています。こちらのカードはDEC21143チップを使っていて、通信速度もちゃんと100Baseの標準的な速度が出ます。こちらのカードはメーカーによって動作保証されていませんが、Planex社のウェブにも情報のある通り、Linksys EtherFast 10/100として認識され、全く修正なしで使用可能です。このカードはおすすめです。

気になる人は/etc/pcmcia/configのLinksys EtherFast 10/100のエントリをPlanex ENW-3601と書き換えておきましょう。カードのCIS IDが同じなので、書き換えない限り表示が直る見込みはありません。(^^)

Xircom RBEM56G-100BTX Xircom 社から発売されている CardBus 規格の Fast Ether & 56K モデムのコンボカードです。

このカードの最大のウリは Ethernet の RJ-45 と電話用の RJ-11 が直に挿せるということでしょう。このため中間ケーブルを忘れてカードが使えないということもないですし、なにより耐久性に問題のある薄型コネクタを壊すということがありません。膝の上やベッドの中などで Ethernet を使うような人[7]にはとっても重宝です。

このカードは結構クセが強く、設定は難しいのですが、結論から言えば使えます。

このカードは Ethernet のチップに自社性の X3201-3 というチップを使っていますが、このチップは DEC の tulip ドライバに比較的近く、pcmcia-cs では tulip_cb ドライバでサポートされています。

このカードをとりあえず使うためには pcmcia-cs 3.1.1 を使うと良いでしょう。この原稿を書いている時点では 3.1.5 のプリリリースまで出ていますが、3.1.2 以降では Ethernet が動きません。ループバック 127.0.0.1 以外のどのアドレスに ping してもパケットが送信されません。

これは 3.1.2 で Doug Ledford によって加えられた変更により、バッファの送信アドレスに 16 バイトオフセットが加算された「改良」に起因しています。

3.1.1 のコードを使っても 100Base での通信には問題があります。レジスタの値が適切に設定されないため、送信性能に対して受信性能が十数倍低速です。筆者はこの部分に修正を行ない、受信性能に関しても十分な性能を引き出せるようにしました。このコードは http://www.cc.rim.or.jp/~yaz/lbook/からダウンロードすることができます。

このカード、モデムの AT コマンドが Windows 用ドライバをインストールしてヘルプを見ない限りわかりませんでした。

モデムの AT コマンド、リザルトコードはヘルプ以外に全く資料がないので、windows を起動した時にヘルプをすべてコピーし、テキストファイルに落しておきましょう。

ちょっと面白い話として、windows ドライバと同時にインストールされるソフトを使うと、モデムの使用国を変えることができます。ハードウェアは全世界共通で、ソフトウェアによって国コードを設定し、各国の仕様・規制に適合させる仕組みとなっているわけです。更に同じことが AT + GCI? (?は国を表す 16 進数) を実行することによって実現できます。

ご存知のように日本ではリダイアル規制がありますが、国コードを US/Canada に変更すればこの規制を回避することができるかもしれません。(違法 (笑))

---

[7]私のことです。(^^)

第4章　萌えるマシンとは自分だけのマシンだ！

結局、自分のマシンに萌えられるかというのは、自分でイメージした「使いやすさ」というものがどの程度実現できているかだと思います。そのためには自分ならではのポリシーを掲げて、その実現に邁進していくことが肝心でしょう。

「ポリシー＝わがまま」という素晴らしい定義を教えてくれた先生がいました。あなたのわがままも、あなたの「したいこと」に人々の共感が得られて、わがままと思われていたことが実は目標の実現のための最良の方法であることがわかればいつかはスタンダードになるかもしれません。(^^)

この章では、私の「わがまま」をいくつか述べたいと思います。

日本語が使えるのと日本語化とは違う

ここ 1、2 年の流れとして、日本語 Linux と言って日本語化 Linux がリリースされています。確かにビギナーにとってはあらゆる表記やメッセージが日本語化されていた方がわかりやすいでしょう。

しかしある程度慣れてくれば特に毎回日本語で表示されてもありがた味は少ないと思います。むしろ他の OS からログインしたとき、漢字が使えなかったり、漢字コードが違っていたときに何が何だかわからなくなってしまうという欠点の方が目につくのではないでしょうか。

また、表示された内容を処理し、パイプやリダイレクトで入力として使う場合も下手に漢字コードが入らない方が楽に処理できるといったこともあります。

私は日本語が使えることと日本語化にはかなり大きな差があると考えています。私の趣味からすると、日本語の使える OS は大好きですが、日本語化された OS は遠慮したいです。

私の考える好ましい日本語対応は、次のようなものです。

- 基本的なコマンドの出力メッセージはデフォルトではすべて英語。特にパイプで結合して使う可能性の高いコマンドでは英語。

- 表示コマンド、エディタは漢字の文字コードを自動判別して表示・処理を行なう。
- 日本語の locale サポートをする場合はきめ細かく環境変数で制御できるようにする。
- man、ドキュメント類は基本的には日本語でよい。ただし対照のために英語版もインストールする。

ソフトウェアのデフォルトに逆らわないシステムを目指す

最近の linux ディストリビューションではプリコンパイルされた大量のソフトウェア群がパッケージとして添付されています。もちろんこれらの効用というのはあるのですが、「萌え」を目指すユーザー[1]としては、新しいバージョンがリリースされれば即ダウンロードしてインストールを検討するはずです。このようなユーザーにとってはパッケージは様々な問題を抱えています。

パッケージではディストリビューションで決めた設定ファイルやインストール位置を墨守します。ここへ tar ボールからコンパイルしてインストールすれば、それまでとファイルのインストール位置が変わってしまうわけですから、ファイルが上書きされず、システムの中に 2 つのソフトウェアが存在することになってしまいます。ソフトウェアの configure 時にインストール位置を指定してもいいのですが、それはいまひとつ面倒ですし、そのソフトウェアの「標準」というか「スジ」というものが覚えられません。誰かに「squid の設定ファイルってどこにあるんだっけ？」と聞かれても「/usr/local/squid の下じゃ～ん？ふつー」という答えはいつまでたってもできません。（笑）

やはりひとつここはソフトウェア作者のテイストを尊重してデフォルトインストール位置でインストールをすることにしてみませんか？　確かにソフトウェア全体の統一的な管理はできませんが、誰かのマシンをメンテナンスする時や、新しいソフトウェアをインストールする時の負担は軽くなるはずです。

[1]毎日 FreshMeat をチェックするのを日課としているようなあなたのことです！

あとがき

以上で「さぶそにっく！」の第 1 回目の本、「Linux ☆萌えるマシンにする方法 ~ 使えるマシンのつくりかた ~ 」はおしまいです。

今回、自分のサークルでのコミケ初参加ということで、強い意気込みで製作を行なってきたのですが、思っていたことの半分もできなかった、というのが実感です。「萌え」という言葉をキーワードに、自分のパートナーになれるような頼もしくも可愛いマシンを作ろうというのがこの本のコンセプトだったのですが、皆さんの持たれる「萌え」のイメージに応えられるだけの内容にはなっていないと思います。

今後はもっとぷりちーなてざわりを大事にして本作りをしていきたいと思いますので、ご支援をよろしくお願いします。よろしければ今回の本の感想やご意見などをメールにてお送り下さい。アドレスは yaz@cc.rim.or.jp です。

最後まで読んで下さった方、本当にどうもありがとうございます。

なお、この本の pdf 版および本文中で触れている自作パッチ等については、準備が出来次第 http://www.cc.rim.or.jp/~yaz/lbook/ にて配布いたしますので、ぜひご利用下さい。

奥付

初版発行　1999 年 12 月 24 日
発行サークル　さぶそにっく！
発行者　矢澤 慶樹
連絡先　〒 285-0023 千葉県佐倉市新町 63