MacOS X WorkShop
— apt-rpm system on MacOS X —
10.4-3
for PowerPC/Intel

KOBAYASHI Taizo

2008/07/03

CarbonEmacs パッケージを入れ、TeX のパッケージを入れ、、、
ghostscript を入れ、gnuplot を入れ、、、、
ドットファイル群を設定して、、、、
でも、LaTeX Equation Editor が動かなかったりする。。。。

web 上の掲示板でしばしば目にする光景です。

みんな殆ど同じ事をするのに、
ひとり一人が、或は一台一台大変な作業を繰り返すのは
開発者か趣味でもない限り大いなる無駄！！

だと思いませんか？

MacOS X で LaTeX や emacs 等の環境を整えられてきた
先人達の成果を集積したものと、
Vine Linux の行き届いた環境を融合させたもの、
それが MacOS X WorkShop です。

MacOS X WorkShop を利用すれば、
定評のある Vine Linux の TeX 環境が一発で構築でき、
すぐさま仕事に入れます。

ただし、このプロジェクトの成果物を利用して不具合が生じても、
プロジェクトとしてもプロジェクトに関係する如何なる人間も
一切責任を負わないものとします。
また、バグ報告やパッケージングの要望は歓迎しますが
迅速な対応は期待しないでください。
と云うよりも、要望をお持ちでしたら、

是非、要望を実現した姉妹 apt-rpm tree を作ってください！！

最後に、
我々は企業のサポート窓口ではありません！
こちらで不具合を再現出来る程度の情報がバグ報告に無い限り、
返事も対応も無いと考えてください。

Copyright ©2004-2008 KOBAYASHI Taizo
All rights reserved.

目 次

図 目 次

1 更新情報

1.1 10.4-3 での変更

- ~/.yatex.el に修正を入れました。(emacs-22.0.990 以降では yahtml の設定で用いていた magic-mode-alist の記述が引っかかる為) ~/.yatex.el を編集していない場合は $ osxws-upgrade 、或いは再ログインで修正は自動で反映されますが、カスタマイズされている場合は MacOSX_WorkShop/dot_emacs_el か/Library/Application Support/OSXWS/jp/.yatex.el を参考にして各自修正をお願い致します。
- 基本パッケージの更新と共に、TeX 環境を刷新しています。ptetex3 と奥村さんの「美文書作成入門 第 4 版」に倣って TeX のデフォルト文字コードを UTF-8 へ変更しました。
- emacs のデフォルト環境変更について
  - OSXWS のデフォルト設定や今後設定方法が変更になる可能性のあるものは site-start.d 内に纏めた
  - .emacs.my.el を廃止して .emacs_osxws.el に一本化
  - .emacs で OSXWS 以外の emacs を利用する場合の設定ファイルを分岐
  - .custums_osxws.el を導入し M-x custum での設定が書き込まれるようにした
  - fixed-width-fontset 1.1.0 の ATSUI を利用：行間が広がり default の 80x40 hiramaru-14 だと、MacBook では下が画面からはみ出てしまうので、画面サイズに合わせてポイントを自動で切り替えるようにした。
- TeX のデフォルト環境変更について
  - ptetex3 と奥村さんの「美文書作成入門　第 4 版」へ対応します。
  - tetex-3.0-10.4osx5 にて土村さんの ptetex3-20061213 版ベースに更新しました。VineLinux 4.2 と同等の環境になります。
- パッケージ追加情報
  - subversion
  - texmacro-aastex
    Astrophysical Journal 投稿原稿作成用 LaTeX class file (Thanks! ぜ-san)
  - texmacro-jps
    日本物理学会誌投稿原稿作成用 LaTeX class file
- 変更パッケージ情報
  - clamav
    UniversalBinary 化しました。
  - gtk2
    cairo と pango の ATSUI 対応が大分実用化され、sylpheed, synaptic のフォントがキレイになりました。

- **その他**~/.mew.el の print に関する記述 (Thanks! fu7mu4-san)
  内部で mac-print-buffer を呼ぶように書き換えました。~/.mew.el は osxws-upgrade では自動更新されませんので、以下の変更を手で加えてください。

```
 ***.mew.el.old 2006-09-01 13:21:04.000000000 +0900
+++ .mew.el     2006-09-01 13:21:14.000000000 +0900
@@ -167,10 +167,10 @@
 ;   (lambda() (goto-char (point-max))))

 ;; 印刷コマンド設定
*(setq mew-print-command-format "mpage -2 -P")
*(defun mew-print-region (begin end)
+;(setq mew-print-command-format "mpage -2 -P")
+(defun mew-print-region (mark point)
       (interactive "r")
*(shell-command-on-region begin end mew-print-command-format))
+     (mac-print-buffer))

 (defun mew-print-buffer ()
     (interactive)
```

1.2 10.4-2 での変更

- Version 10.4-2 for PowerPC/Intel

- パッケージの大部分を Universal Binary 化

- Intel Mac に対応
  binary package は i386, fat, ppc, noarch の組み合わせで行きます。

- アンインストールをサポート
  以下のコマンドとそれに続く確認に了承すればアンインストールできます。

```
$ sudo apt-get remove OSX-system
```

- 英語環境を睨んで
  各ユーザーの dot files を/System/Library/User Template/Japanese.lproj/ から /Library/Application Support/OSXWS/jp/ へ移動。この結果 OSXWS インストール後に新規ユーザーを作成しても OSXWS とは切り離された素のユーザー環境が作られます。その新規ユーザーが OSXWS を利用したい場合は以下のコマンドを実行して dot files を整えてください。

```
$ /usr/local/bin/osxws-upgrade
```

- **パッケージ追加情報**

- clamav, gmp
  ClamAV を packaging しました。daemon の扱いを MacOSX に準拠させ/Library/StartupItems?/clamav/ 以下に起動と停止のスクリプトをおきました。自動で clamd, freshclam が daemon として動きます。
- cmucl, Maxima, Imaxima, clisp(test tree)
  デフォルトの lisp を cmucl に変更して Maxima を復活させました。 test tree に clisp と maxim-exec-clisp を置いておきますが clisp はメンテナンス対象外です。
- fugu
  Cocoa で書かれた sftp client
- fftw3
  研究で必要になったから
- Desktop Manager
  一年以上利用しているのと source が tar ball で配布されたので packaging しました。
- ImageMagick
  やはり無いと不便であるから。
- synaptic
  これで GUI でパッケージ管理出来ます！ 関連して gtk2 も用意しました。起動 (mlterm 上) とマニュアルの表示は以下で行ってください。

  ```
  $ sudo synaptic
  $ open /usr/local/share/synaptic/html/index.html
  ```

- gcc-g95
  gcc-g77 と排他利用になりますが用意しました。

• 変更したパッケージ

- ispell から aspell
  辞書にない単語を登録している場合は、各自 ispell の時のリストを登録しなおす必要がある。`~/.ispell_english` ファイルを開いて ispell-buffer を実行する。新しい辞書は「`~/.aspell.en.pws`」。– seto
- LatexEquationEditor から LaTeXiT
  今後の発展を見込んで移行。ただし LatexEquationEditor のサポートも続けます。お好みに応じて使い分けてください。
- kterm から mlterm
  locale を ja_JP.UTF-8 へ変更するに伴い移行。
- teTeX-3 ベースに更新
  dvipdfmx と齋藤さんの OTF パッケージを自動で組み込む updmap-otf の調整に手間取ったが、漸く仕事で使える様になった。TeX 関連では、昨日瀬戸さんと議論の上、.emacs.el から yatex に関する記述を .yatex.el へ移した。
- ghostscript の version は 8.51 で組んでみることにした。ヒラギノをデフォルトにしました。
- less から lv

- 削除したパッケージ
  - vim
    vim は multi_byte でコンパイルされている。`~/.vimrc` を弄って利用可
  - bizp2
  - freetype

2　はじめに

　このプロジェクトの目的は、
TeX や emacs を用いて仕事をしている人が、
MacOS X 上にストレス無く即座に仕事に掛かれる環境を構築する事
と
大学や研究機関等の計算機管理者が、
MacOS X 上に独自の研究環境を簡単に築き管理する事
にあります。
　念頭に置いている TeX, emacs 環境は大学で支持を得ている **Vine Linux**[1] です。
この Vine Linux の中で日々の仕事に必要なパッケージを MacOS X に合わせて構築し直した物と、
MacOS X 上の便利なソフトを組み合わせたものが MacOS X WorkShop の実態です。
現在公開しているパッケージは
MaxOS X 10.4.x (Tiger) 対応版と **MaxOS X 10.3.x (Panther) 対応版**[2] です。
尚、今後 Panther 版の拡張は致しません。
パッケージに関する詳細情報は rpm2html による **RPM 解説データベース (Tiger)**[3] **RPM 解説データベース (Panther)**[4] をご利用ください。

　MacOS X WorkShop 固有の拡張を施してあるパッケージの内容に関しては
「パッケージメモ (Section10 参照) 」をご覧ください。

2.1　何故 apt-rpm か？

　パッケージの作成、管理、利用の全てで
楽ができるから です。

　例えば、TeX の環境を構築したいのであれば、
ターミナルを開いて

```
$ sudo apt-get update
$ sudo apt-get install OSX-base
$ sudo apt-get install task-tetex
$ sudo apt-get clean
```

とするだけで TeX 関連のパッケージをまとめてインストールし、
且つ、各ユーザーのドットファイル群を含む面倒な各種設定まで
自動で片付けてくれます。

[1] http://www.vinelinux.org/
[2] ../Panther/index.html
[3] ../Tiger/rpm2html/
[4] ../Panther/rpm2html/

パッケージの更新はバグが見つかる度に成されますが、その場合でも、

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get clean
```

でおしまいです。
いちいちインストールし直す必要は無いのです。

この様な楽ができるのは apt-rpm system に先人達の成果を集積しているからです。
ソース・パッケージ (hoge-ver-rel.src.rpm) には
ソースだけでなく、パッチやコンパイル、インストールの仕方まで
こと細かに書かれています。
つまり web 上に散らばった情報を集積している訳です。
OSXWS をインストールして パッケージを開発する (Section6 参照) してみれば
貴方が何時間も、時には何日も掛けて探しまわった情報と作業が
たった一つの src.rpm ファイルに凝縮されている事に気づく筈です。

どうですか？
かなり楽が出来そう
ではありませんか？

MacOS X 上での UNIX 研究環境を構築するには **Fink**[5] をはじめ、
琉球大学の **EasyPackage**[6] や、**DarwinPorts**[7] 等があり、
各々がそれぞれのパッケージングシステムを持っています。
他にもパッケージングシステムを持たない総合情報として **Mac Wiki**[8] が在り、
自分が必要とするソフトを手で一つ一つ入れる事も出来ます。

当然の事ですが、それぞれに利点と欠点があります。
Fink の利点は何と言ってもパッケージの多さでしょう。
その反面大きなプロジェクトである為に、
我々の些細な（しかし研究上無視出来ない）変更を施すのは余分な労力を要します。

EasyPackage の利点は琉球大学で既に利用されていて、
且つ、ある程度活発なコミュニティーが存在していることだと思います。
しかし rpm の様に枯れた技術とは言い難く、

[5] http://fink.sourceforge.net/
[6] http://www.ie.u-ryukyu.ac.jp/darwin2/
[7] http://darwinports.opendarwin.org/
[8] http://macwiki.sourceforge.jp/cgi-bin/wiki.cgi

Linux の使用歴が長い研究者にとっては多分に不安を覚えるのも事実です。

DarwinPorts は Apple に最も近い存在ですが、
既にかなり大きなプロジェクトになっている事と、
小林が FreeBSD を殆ど知らない事から対象外になっています。

apt-rpm を用いる副次的な利点としては、同じシステムを用いている
Vine Linux 等 Linux の成果を活かし易い 事があげられます。
以下、システムの簡単な概要を説明します。

rpm[9] は **Red Hat**[10] が Linux distribution のパッケージングシステムとして開発したものです。
rpm, rpmbuild 等のコマンドを通して、パッケージの

- 作成
- インストール
- 更新
- 除去

を行います。パッケージ間の依存関係の情報をパッケージ自身が持っているので、
必要なライブラリを抜かしてインストールする様なミスを防ぐ事が出来ます。
Vine Linux[11] 等、多くの Linux Distributer がこのパッケージングシステムを採用しており、
MacOS X WorkShop に移植する際にそれらを拝借出来ます。
また、ソフトベンダーが Linux 向けの製品を提供する場合は殆 rpm 形式が使われており、
Linux での標準的なパッケージングシステムになっています。

apt[12] は **Debian GNU/Linux**[13] が Linux distribution のパッケージ管理ユーティリティとして開発したものです。
Debian の特徴は rpm ではなく独自のパッケージシステムを利用している事と、
8000 以上の膨大なパッケージ数を抱えている事です。
パッケージングシステムは兎も角、
この様な膨大なパッケージを利用する為には何らかの強力なパッケージ管理ユーティリティが必要です。
apt はその目的を果たす為に開発されています。
apt-get, apt-cache 等のコマンドを通して、

- 現在利用可能なパッケージの情報を得る。
- 更新されたパッケージを自動でアップデートする。

[9] http://www.rpm.org/
[10] http://www.redhat.com/
[11] http://www.vinelinux.org/
[12] http://www.debian.org/doc/manuals/apt-howto/
[13] http://www.jp.debian.org/

- パッケージ間の依存関係を自動で調整して適切なインストールと削除をしてくれる。

等、一度利用したら手放せなくなる機能を提供してくれます。

apt-rpm[14] は **Conectiva Linux**[15] が Linux distribution のパッケージ管理ユーティリティとして Debian の apt を rpm に対応させたものです。

MacOS X WorkShop では Conectiva の apt-rpm を Vine Linux が日本語対応にした物を流用しています。

rpm や apt の利用方法は「MacOS X WorkShop の使い方 (Section5 参照) 」を御覧下さい。

2.2 プロジェクトのポリシー

この MacOS X WorkShop プロジェクトには、以下のポリシーがあります。

- 管理に手間を掛けない。
- パッケージ数は必要十分に留める。
- 自分たちに都合の良い設定やパッチを用いる。
- それぞれの大学や研究室での派生プロジェクトを立ち上げやすくする。

です。

管理者がたった一人でも `PowerBook` と一日の時間さえあれば、
一通り全パッケージのメンテナンスが出来るくらいの
小さなディストリビューションに出来るだけ留めます。

結局のところ如何に手間ひまを掛けずに必要十分な事を好き勝手にするかが本音です。

煩わしい計算機管理は出来るだけ楽に済まし、自分の本分にリソースを集中する環境を作るのが、
MacOS X WorkShop の目的でありポリシーでもあります。

2.3 派生プロジェクトの歓迎

プロジェクトの目的の一つとして、
立命館大学物理学教室で立ち上がったこのプロジェクトをひな形にした
姉妹 apt-rpm tree が作られる事を歓迎します。
各大学や研究室で独自の拡張や変更を施した姉妹 apt-rpm tree を是非作ってください。

[14] https://moin.conectiva.com.br/AptRpm
[15] https://moin.conectiva.com.br/

インストーラの作り方は「インストーラを作る (Section7 参照) 」を、
apt-rpm tree の作り方は「apt-rpm tree を作る (Section8 参照) 」を、
それぞれ御覧下さい。
貴方がたに必要なパッケージだけを集めた add-on tree も全く同様に作成可能です。
まずは、この MacOS X WorkShop を母体にした貴方独自の　apt-rpm add-on tree を
local disk に作る事から始められる事をお薦めします。
もしも、姉妹 apt-rpm tree を作られ { る, た } 際には是非ご一報ください。
姉妹 trees !! (Section3 参照) のページで紹介するとともに、
MacOS X WorkShop の apt-line に追加します。
そしてお互いに楽をし合いましょう。

2.4　連絡先とメンバー

MacOS X WorkShop に関する議論や連絡は **Mac Wiki**[16] を利用させて戴いています。
Mac Wiki で議論すれば、情報が蓄積されていき多くの人にとって有益です。

この web page が更新されるまでの変更は Mac Wiki でアナウンスしますので
出来るだけチェックするようにしてください。
また、バグ報告は基本的に OSXWS 標準の環境に対してのものにしてください。
勿論、.emacs.my.el 等を改変して独自の拡張を施すのは一向に構いませんが、
その結果現れた不具合の場合は必ず OSXWS に問題がある事を特定してから報告してください。
また、問題が解決した場合にも必ず報告して、言いっ放しにはしないでください。

現在のメンバーです。(順不同)

小林泰三 九州大学情報基盤研究開発センター、学術研究員

瀬戸亮平 パリ南大学 PD

坂田泰啓 ニコン株式会社 ('03 池田研マスター卒業生)

新山友暁 立命館大学物理学教室池田研究室 D2

大関 努 立命館大学物理学教室倉辻研究室 '06 卒業

2.5　ライセンス

収録しているパッケージのライセンスは、
パッケージに収録しているソフトウェアのライセンスに従います。
`$ rpm -qi hoge` でパッケージ hoge のライセンスを確認出来ます。
また `/usr/local/share/doc/hoge` 以下にもライセンスに関するファイルが在ります。

インストーラのライセンスは GPLv2 以降に従うものとします。
インストーラに同梱されている ReadMe.rtf, License.rtf を参照して下さい。

[16] http://macwiki.sourceforge.jp/cgi-bin/wiki.cgi

3　姉妹 trees !!

繰り返しになりますが MacOS X WorkShop の目的は、
TeX や emacs を用いて仕事をしている人が、
MacOS X 上にストレス無く即座に仕事に掛かれる環境を構築する事
と
大学や研究機関等の計算機管理者が、
MacOS X 上に独自の研究環境を簡単に築き管理する事
です。

一口に「楽をする」と云っても計算機環境に求められるものも好みも千差万別です。
その様な状況で管理者とユーザーの双方が楽をする為には、
それぞれの環境に合わせた apt-rpm tree を構築するのが一等です。
apt-rpm tree を零から新たに作るのは結構な作業に成りますが、
この MacOS X WorkShop をひな形にすれば、数日で実現可能です。
例えば、
デフォルトのログイン環境や .emacs.el を変えたければ、
OSX-Preferences パッケージを弄るだけで済みますし、
emacs に lisp file を加えたければ、
emacs-lisps パッケージに加えればおしまいです。

このページでは、姉妹 apt-rpm trees の紹介をします。
全て、MacOS X WorkShop の apt-line (/priveta/etc/apt/sources.list 内に記述) に加えてあります。
貴方の求めているものに最も近い tree をご利用ください。

- **HEPonX**[17]
  KEK の藤井恵介さんが高エネルギー物理学の計算機環境を MacOSX 上に実現する為に作られた apt-rpm tree です。

  藤井さんは、PPC Linux の黎明期から Mac 上の Linux 環境の整備に貢献してこられ、MacOSX 上に rpm を最初に 移植した方です。MacOSX WorkShop (OSXWS) の rpm の基本部分は藤井さんの成果を利用しています。

- **MacOS X WorkShop**[18]
  立命館大学物理学教室で立ち上げられ利用されています。

[17] http://www-jlc.kek.jp/ fujiik/macosx/10.4.X/HEPonX/
[18] http://www.bach-phys.ritsumei.ac.jp/OSXWS/

4　MacOS X WorkShop をインストールする

このセクションでは MacOS X WorkShop をインストールする手順について説明します。
尚、 MacOS X 10.3 (Panther) に関する情報はこちら[19] をご覧ください。

4.1　インストールする前に…

警告！

MacOS X WorkShop のインストール環境は、素の MacOS X に下記のパッケージをインストールした状況を想定しています。Fink との共存は可能かもしれませんがプロジェクトとしては考慮していません。また、他の方々が配布されている CarbonEmacs, teTeX 等がインストールされている場合、予期しない結果になる可能性があります。

MacOS X WorkShop は MacOS X 上で emacs などの UNIX ツールを利用する為の環境を構築するものです。
従って、以下の MacOS X のインストール条件を満たす必要があります。

- `X11`
  mlterm, gv, gnuplot, xgraph, yaplot.... 等の X11 のソフトを利用するのに必要です。
  `/usr/X11R6/bin/Xquartz` がインストールされていれば大丈夫です。

- `Xcode`
  ここ[20] から最新版を入手してください。
  Apple が提供している開発環境です。インストールする時に `X11` 開発環境を必ず選択してください。

4.2　ターミナルの設定

次に apt-rpm を操作するターミナルの設定をします。

1. 先ず「アプリケーション」→「ユーティリティ」の中にある『ターミナル』をダブルクリックして起動します。

2. メニューバーの「ターミナル」から「ウインドウ設定...」を選択し『ターミナルインスペクタ』フローティングパネルを表示させます。

3. 「シェル」と表示されている選択項目をプレスして『エミュレーション』を選択します。

4. 「非 ASCII 文字をエスケープする」のチェックを外します。

[19] ../Panther/index.html
[20] http://developer.apple.com/tools/xcode/

5. 選択項目をプレスして『ディスプレイ』を選択します。

6. 「ワイドグリフは２桁とカウントする」にチェックを付けます。

7. 好みに応じて「アンチエイリアス処理を行う」にチェックを付けます。

8. 好みに応じて「フォント設定...」でフォントのサイズを変更します。

4.3 Install

MacOS X WorkShop を始めるには
MacOS X WorkShop start kit MacOSX-WS-10.4.3.dmg[21]
をダウンロードしてインストールします。
(ソース一式は **MacOSX-WS-10.4.3.tar.bz2**[22] として置いておきます。)

注意！

このインストーラには必要最低限のバイナリしか含まれていません。
必ず「MacOS X WorkShop の使い方 (Section5 参照) 」を参照してインストールを完結してから
必要なパッケージをインストールして下さい。

尚、このインストーラは内部で以下の処理を行います。

1. `apt-rpm` のインストール
   apt-rpm を利用する為の核となるものです。
   apt, popt, rpm, beecrypt, neon, expat package の中から必要な物を抜き出したものです。
2. `rpm data base` の構築

```
$ sudo rpm --initdb
```

   を実行します。
3. `OSX-system, OSX-X11` パッケージのインストール
   MacOS X に存在するリソースを rpm に知らせるパッケージをインストールします。
   `/private/etc/{csh.login-osxws,profile-osxws,zprofile}` が加えられ、`/usr/local/bin` 等

[21] MacOSX-WS-10.4.3.dmg
[22] MacOSX-WS-10.4.3.tar.bz2

にパスを通します。

尚、オリジナルファイルは.rpmorig のサフィックスを付けて保存されます。

インストールされる設定ファイルは **OSX-system**[23] にてご確認ください。

4. ユーザー用初期設定ファイル (dot files) のインストールと配布

OSX-Preferences package をインストールします。
また、各ユーザーに以下の設定ファイルを配布します。
既に存在する時はファイル名の末尾を .rpmold に変えて保存した上で配布されます。

- `.Xclients`
  X11 を起動する時に実行されるスクリプトです。
  mlterm や kinput2, login window をここで起動しています。
- `.Xresources`
  主に xterm の設定が記述されています。
- `.bashrc, .bash_profile`
  bash の設定ファイルです。
  OSX-Preferences package の更新に対応する為に個人用の記述は .bashmyrc の中に記述して下さい。
- `.cshrc, .tcshrc`
  csh, tcsh の設定ファイルです。
  .tcshrc は .cshrc へのシンボリックリンクです。
  OSX-Preferences package の更新に対応する為に個人用の記述は .cshmyrc の中に記述して下さい。
- `.zshenv, .zshrc`
  zsh の設定ファイルです。
  OSX-Preferences package の更新に対応する為に個人用の記述は .zshmyrc の中に記述して下さい。
- `.emacs.d`
  emacs の設定ファイルです。
  OSX-Preferences package の更新に対応する為に個人用の記述は .custom_osxws.el の中に記述して下さい。
- `.inputrc`
  ターミナル上で日本語をシームレスに扱う為の設定が書かれています。
- `.mew.el`
  emacs 上で mew を利用するにはこのファイルを各自編集します。
- `.yatex.el`
  yatex の設定ファイルです。
- `.vimrc`
  vi の設定ファイルです。

[23] OSX-system/

- `.rpmmacros`
  rpm package を構築する時は、
  予め各自このファイルを編集しておく必要があります。
- `.signature`
  メールの署名ファイルです。
- `.xinitrc`
  .Xclients を有効にします。

- `rpm`
  rpm package を構築する時の作業ディレクトリです。

インストールされる設定ファイルは **OSX-Preferences-10.4.tar.bz2**[24] をダウンロードしてご確認ください。

また、OSXWS デフォルトの設定ファイル群は
`/Library/Application\ Support/OSXWS/jp/`
以下に有りますので、
local file の編集に失敗した時など必要な時にコピーしてお使いください。

> **注意！**
> 各ドットファイルはピリオドから始まるため、Finder から直接見る事は出来ません。
> 展開後に terminal 上で cat コマンド等を利用して確認して下さい。

4.4 Remote Install

MacOS X 標準の installer コマンドを用いて
リモートでインストールする場合には以下の手順を踏んでください。

> **注意！**
> w コマンドなどを用いてユーザーが作業していない事を確認してから行ってください。

1. **イメージをマウント**
   インストーラが入ったディスクイメージ[25] をマウントします。

```
$ hdid MacOSX-WS-10.4.3.dmg
```

2. **インストール**
   installer コマンドを用いてインストールします。

```
$ sudo /usr/sbin/installer -pkg /Volumes/MacOSX-WS-10.4.3/MacOSX\ WorkShop\ start\ kit.pkg -tar
$ hdiutil eject /Volumes/MacOSX-WS-10.4.3
```

[24] OSX-Preferences-10.4.tar.bz2
[25] MacOSX-WS-10.4.3.dmg

3. **再起動**
   再起動します。

```
$ sudo reboot
```

4.5 Panther 版からの Upgrade

警告！
プロジェクトとしては新規インストールを推奨します。

MacOS X WorkShop をインストールした Panther を、
上書きで Tiger に Upgrade する場合は以下の手順を踏んでください。

1. **X11**
   Tiger への Update ではデフォルトでは X11 が選択されません。
   「カスタムインストール」にして明示的に指定してください。

2. **Xcode**
   ここ[26] から最新版を入手してください。
   Apple が提供している開発環境です。インストールする時に **X11 開発環境を必ず選択**してください。

3. **source.list の編集**
   /private/etc/apt/sources.list を Tiger tree へ向けます。

```
##
## MacOS X WorkShop Tiger
##
# (master)
rpm      http://www.bach-phys.ritsumei.ac.jp/OSXWS Tiger/ppc main
rpm-src  http://www.bach-phys.ritsumei.ac.jp/OSXWS Tiger/ppc main
```

4. **apt の更新**
   先ず apt と rpm を Tiger 版に更新します。
   この時に削除されるパッケージが出る場合がありますが、
   その場合は**「全パッケージの更新」**の後で入れ直してください。

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get clean
```

5. **全パッケージの更新**
   最後に全パッケージを Tiger 版に更新します。

[26] http://developer.apple.com/tools/xcode/

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get clean
```

4.6 Uninstall

警告！

MacOS X WorkShop のみをインストールしている状況を想定しています。
`/usr/local,/private/var/local` 以下にファイルを置いている場合は該当するファイルをバックアップしておいてください。

アンインストールは簡単です。
以下のコマンドを実行し指示に従えば、
システムと各ユーザーの環境を素の状態に戻すことができます。

```
$ sudo apt-get remove OSX-system
```

5 MacOS X WorkShop の使い方

MacOS X start kit のインストールが無事済んだならば、
後は自分が利用するパッケージを apt で入れるだけでおしまいです。
お使いの計算機が firewall の内側である場合に限り、
.bashmyrc を編集して環境変数 http_proxy や ftp_proxy を適宜設定
しておく必要が在ります。
該当箇所がコメントアウトされていますので、
お使いの環境に合わせて記述してください。

start kit をインストールした後に先ずすべき事は
OSX-base パッケージを apt でインストールする事です。

MacOS X WorkShop で必須の根幹パッケージをインストールしてくれます。
計算機がインターネットに接続されている事を確認してターミナルから

```
$ sudo apt-get update
$ sudo apt-get install OSX-base
$ sudo apt-get clean
```

を実行してください。
これが済んだら基本的に後は自由に必要なパッケージをインストールして戴いてかまいません。
CarbonEmacs の環境を手っ取り早く構築したい人は、
計算機がインターネットに接続されている事を確認してターミナルから

```
$ sudo apt-get update
$ sudo apt-get install task-emacs
$ sudo apt-get clean
```

を実行してください。
これだけで Carbon Emacs を利用する為の基本的な環境が構築されます。

注意！

emacs は /Applications/Emacs.app です。
「牛のアイコン」をダブルクリックして起動して下さい。
勿論 terminal や mlterm 等から $ emacs hoge.txt としても起動出来る様に alias を設定してあります。
デフォルトでは emacs で TeX のファイルを作成すると文字コードは日本語 EUC になります。

TeX の環境を構築したい人は

```
$ sudo apt-get update
$ sudo apt-get install task-tetex
$ sudo apt-get clean
```

を実行してください。
これだけで齋藤さんの OTF パッケージでヒラギノを利用できる TeX 環境が構築されます。

> **注意！**
> TeX を利用する環境として emacs + YaTeX を想定しています。
> pTeX は S-JIS で make されています。
> terminal や mlterm 上で emacs で作成したファイルをコンパイルする時には
> platex-euc コマンドを使用して下さい。

apt と rpm を用いて細かな操作をする必要がある人は以下の記述が役に立つかもしれません。
勿論 man コマンドを活用してくださいね。

5.1 apt の「いろは」

Tiger 版には apt の GUI frontend である **Synaptic**[27] を用意しました。
X11 の mlterm 上で

```
$ sudo apt-get update
$ sudo apt-get install synaptic
$ sudo apt-get clean
$ sudo synaptic
```

で利用できます。
利用法は**マニュアル**[28] を参照してください。
以下の説ではターミナルでの apt の利用方法を簡単に紹介します。

5.1.1 毎回最初に必ずすべき事

apt を利用するには何は兎もあれ**データベースの更新**をする必要があります。
これをしないと現在の apt line[29] の状態を反映出来ず、
存在しないパッケージをインストールしようとしたりしてまともに働いてくれません。

ですから apt を弄る時は、必ず最初に

[27] http://www.nongnu.org/synaptic/
[28] file:///usr/local/share/synaptic/html/index.html
[29] apt が利用するパッケージやその情報が置いてある場所の事

```
$ sudo apt-get update
```

する様に癖をつけてください。

5.1.2 パッケージの探し方

例えば、現在利用できる emacs に関係するパッケージを知りたいとしましょう。
その様な時は apt-cache search を利用します。
具体的には

```
$ apt-cache search emacs
Mule-UCS - MULE-UCS is a coding system and character code translator system.
apel - Emacs 用の 基礎的な関数を提供するライブラリ
emacs -GNU Emacs エディタ
emacs-lisps - Carbon Emacs 用の便利な Lisp ライブラリ集
emacsen-common - Common facilities for all emacsen.
flim - Emacsen 用の message に関する表現形式や符号化のためのライブラリです。
mew - Emacs でメールを読むためのインターフェース
mew-common - Emacs/XEmacs 用 Mew 両方で利用するファイル/プログラム
readline - A library for editing typed command lines.
semi - Emacsen 用の MIME の機能を提供するライブラリ
task-emacs - emacs バーチャルパッケージ
yatex - 野鳥 (YaTeX) - Yet Another TeX mode for Emacs
```

の様にすれば、emacs に関連したパッケージの一覧が得られます。

5.1.3 パッケージのインストール

apt-cache を用いてインストールしたいパッケージが見つかったら、
apt-get install を利用してインストールします。

例えば、a2ps をインストールする場合には

```
$ sudo apt-get install a2ps
パッケージリストを読みこんでいます... 完了
依存情報ツリーを作成しています... 完了
以下の追加パッケージがインストールされます:
  psutils
以下のパッケージが新たにインストールされます:
  a2ps psutils
アップグレード: 0 個, 新規インストール: 2 個, 削除: 0 個, 保留: 0 個
```

```
1637kB のアーカイブを取得する必要があります。
展開後に 4714kB のディスク容量が追加消費されます。
続行しますか? [Y/n]
取得:1 http://www.bach-phys.ritsumei.ac.jp Panther/ppc/main psutils 1.17-10.4osx0.1 [80.5kB]
取得:2 http://www.bach-phys.ritsumei.ac.jp Panther/ppc/main a2ps 4.13b-10.4osx0.2 [1556kB]
1658kB を 6s 秒で取得しました (256kB/s)
変更を適用しています...
準備中...                ############################################ [100%]
   1:psutils                ############################################ [ 50%]
   2:a2ps                   ############################################ [100%]
完了
```

の様になります。

パッケージ間の依存関係が解決されて、psutils が同時にインストールされているのが判ります。

5.1.4 パッケージの削除

いらなくなったパッケージを削除したい時にはどうすれば良いでしょう？
その様な時は apt-get remove を利用します。
例えば、psutils を削除したい場合

```
$ sudo apt-get remove psutils
パッケージリストを読みこんでいます... 完了
依存情報ツリーを作成しています... 完了
以下のパッケージが削除されます:
  a2ps psutils
アップグレード: 0 個, 新規インストール: 0 個, 削除: 2 個, 保留: 0 個
0B のアーカイブを取得する必要があります。
展開後に 4783kB が解放されます。
続行しますか? [Y/n]
変更を適用しています...
準備中...                ############################################ [100%]
完了
```

の様になります。

ここで psutils に依存している a2ps が存在する場合、a2ps も削除するかどうか確認してきます。
ですから、あるパッケージを抜いてしまったが為に動かなくなるパッケージは、バグでない限りありません。

5.1.5 パッケージの更新

計算機のソフトにバグはつきものですし、機能が追加されてどんどん更新されていくものです。
MacOS X WorkShop でも、当然バグつぶしに因るパッケージのアップデートはしていきますし、
開発元が新版をリリースすれば出来る範囲で追随します。
即ち、パッケージはどんどん新しくなっていきます。
その様な新しいパッケージに自動で更新する方法があります。

一つ目は **依存関係を解決する時に、パッケージの削除が伴わないものだけ**を更新する方法で、
`apt-get upgrade` を利用します。

二つ目は **パッケージの削除を伴っても依存関係を解決して最新の状態にする**方法で、
`apt-get dist-upgrade` を利用します。

```
$ sudo apt-get dist-upgrade
```

小林は開発に携わっている事もあり、常に apt-get dist-upgrade しています。

5.1.6 後片付け

apt-get で取得したパッケージは
`/private/var/local/cache/apt/archives/` 以下に置かれます。
これは、`apt-get clean` を実行しない限り、残り続けます。
必ず最後に実行しておきましょう。

```
$ sudo apt-get clean
```

5.2 rpm の「いろは」

パッケージをインストールしたり更新したりするのは apt に任せれば良いのですが、
パッケージそのものを相手にする場合は rpm コマンドを直接操作する他ありません。
ここでは普段小林がよく使うコマンドについて簡単に解説します。

尚、パッケージの作成方法に付いては「パッケージの開発 (Section6) 」をご覧ください。

5.2.1 パッケージの情報あれこれ

今インストールされているパッケージの情報を知りたいとします。
先ず、今インストールされている全てのパッケージを知るには `rpm -qa` を用います。
実際には sort にパイプして

```
$ rpm -qa | sort | less
```

としたり、

```
$ rpm -qa | grep devel | sort
```

として目的のパッケージを探します。

こうして調べたいパッケージを見つけたならば、
何時誰が作ったパッケージで何時インストールされたのか、
等の情報を得ることができます。
それには rpm -qi を用います。
例えば a2ps の情報であれば

```
$ rpm -qi a2ps
Name        : a2ps                         Relocations: (not relocateable)
Version     : 4.13b                             Vendor: (none)
Release     : 10.3tk1                       Build Date: 金  8/20 11:53:53 2004
Install Date: 日  8/22 16:21:11 2004      Build Host: ww5pt176.local
Group       : Applications/Publishing       Source RPM: a2ps-4.13b-10.3tk1.src.rpm
Size        : 4511931                          License: GPL
Signature   : (none)
Packager    : KOBAYASHI R. Taizo        <xxxxxxx@mac.com>
URL         : http://www.inf.enst.fr/~demaille/a2ps/
Summary     : テキストなどの Postscript へのフィルタ
Description :
a2ps は優れた印刷能力をもった，テキストを PostScript へ変換するフィルタ
です。
これは、プログラム言語や文字コード (ISO Latins，Cyrillic，EUC-JP 等)、
用紙、(インタフェースに対して)NLS などを広範囲にサポートしています。
いくつかのファイルを別のアプリケーションでフィルタリングさせる機能も持っ
ており、DVI や PostScript 等を全く同じインタフェースで区別することなく印
刷することができます。
```

として入手出来ます。

では、a2ps で一体どのようなファイルが何処にインストールされているのかを知るにはどうすれば良いのでしょうか。
それには rpm -ql を用います。

```
$ rpm -ql a2ps
```

```
/private/etc/a2ps-site.cfg
/private/etc/a2ps.cfg
/usr/local/bin/a2ps
/usr/local/bin/card
/usr/local/bin/composeglyphs
...................
/usr/local/share/ogonkify/ptmri-o.ps
/usr/share/info/a2ps.info.gz
/usr/share/info/ogonkify.info.gz
/usr/share/info/regex.info.gz
```

最後に、このパッケージの履歴をみてみましょう。
それには以下の様にします。

```
$ rpm -q --changelog a2ps |less
```

5.2.2 パッケージのインストールと更新

ダウンロードしてきたパッケージをインストールする方法や、
自分で構築したパッケージをインストールする方法を述べます。
例えば、パッケージ hoge-1.23-10.4osx1.ppc.rpm をインストールするには

```
$ sudo rpm -ivh hoge-1.23-10.4osx1.ppc.rpm
```

或は

```
$ sudo rpm -Uvh hoge-1.23-10.4osx1.ppc.rpm
```

とします。
-i オプションはインストールを、
-U オプションは更新を意味しますが、
殆どの場合 -Uvh で済んでしまいます。
他に、`--force` や `--nodeps` 等のオプションがありますが、
パッケージの作成をしない限り、まず使う状況は無い筈です。

5.2.3 パッケージの削除

大抵は apt-get remove で事足りるのですが、
開発作業中にどうしても依存関係を破壊しても一時的に削除しなければならない場合は
rpm コマンドに頼る他ありません。

その様な時は

```
$ sudo rpm -e --nodeps hoge
```

とします。

6　パッケージを開発する

ここでは MacOS X WorkShop のパッケージを開発する方法を述べます。
コマンドは rpm ではなく rpmbuild を使います。

MacOS X WorkShop に固有の事項について説明しますので、
一般的な rpm パッケージの作成方法は、
Vine Linux の **Making RPM**[30] や、
Momonga Linux の **Specfile-Guidance**[31] を参考にしてください。

亦、パッケージに固有の項目に関してはパッケージメモ (Section10 参照) を参照して下さい.
尚、

```
$ rpm -i hoge-1.0-10.4osx1.src.rpm
```

とすると、
spec file は `~/rpm/SPECS` に、
source files は `~/rpm/SORCES` に、
それぞれ入ります。

apt tree に在る rpm source package を利用するのであれば

```
$ cd ~/rpm/SRPMS
$ apt-get source hoge
```

とすると、
hoge の source package が `~/rpm/SRPMS` にダウンロードされた後に
spec, source files を所定の位置に展開してくれます。

6.1　設定ファイルの編集

rpm のパッケージを作る前に、パッケージャの情報を `~/.rpmmacros` に記述しておきます。
vi 等のエディタで `~/.rpmmacros` の packager の項目に、
アルファベットで自分の名前とメールアドレスを以下の様に整えます。

```
%_topdir ${HOME}/rpm
%packager KOBAYASHI Taizo    <xxxxxxx@xxxx.xxx>

%_tmppath %{_topdir}/temp
```

[30] http://www.vinelinux.org/MakingRPM/index.html
[31] http://www.momonga-linux.org/docs/Specfile-Guidance/ja/index.html

```
%_signature gpg
%_gpg_name XXXXXXXX
```

これで貴方が作るパッケージには貴方の名前とメールアドレスが刻まれます。
gnupg の public key をお持ちの方はご連絡いただければ OSX-keyring に登録いたします。

6.2 spec file のタグ

ここでは MacOS X WorkShop 固有のタグに関する方針を述べます。

Version, Release rpm のパッケージは
(Name)-(Version)-(Release)-(architecture).rpm
の形をしています。
(Name), (Version) はパッケージングするソフトに依存するので一意に決定されますが、
(Release) の付け方はディストリビューション毎に取り決めがあるのが普通です。
MacOS X WorkShop では

Panther `10.3tk(release number)`

Tiger `10.4osx(release number)`

と付ける事にします。

(architecture) は特に指定しなければ ppc になります。
スクリプトやドキュメントだけのパッケージでは `BuildArch: noarch` を指定すると noarch になります。

defattr %files セクションに、そのパッケージに含まれるファイルを書き込みますが、
それらのファイルのオーナーとグループを指定してやる必要があります。
それが %defattr タグです。

MacOS X WorkShop では、
`%defattr(-, root, wheel)`
を標準とします。

6.3 rpm macro

ここでは MacOS X WorkShop 固有のマクロについて述べます。

デフォルトのマクロは `/usr/local/lib/rpm/macros` に記述されているので、
パッケージを作成する前に必ず一度は目を通しておいてください。

先ず、マクロの内容が MacOS X WorkShop 固有のものを列挙します。

_prefix `/usr/local`
基本的に全てのバイナリーやライブラリ、ドキュメント等は`/usr/local` 以下にインストールします。

_var /private/var/local

_sysconfdir /private/etc

__libtoolize /usr/bin/glibtoolize, /usr/bin/glibtool を使います。
/usr/bin/libtool はライブラリを作る ar, ranlib の代替になる存在の様で、
gnu の libtool とは役割が異なります。

次に、MacOS X WorkShop のみに存在するマクロを列挙します。

```
#-----------------------------------------------------------------------
#
# MacOS X macro
%_cpmac         /Developer/Tools/CpMac
%_mvmac         /Developer/Tools/MvMac
%_fontsdirmac   /Library/Fonts
%_docdirmac     /Users/Shared/Docs/%{name}-%{version}
%_appdirmac     /Applications
%_stuffit       open -a StuffIt\\ Expander.app
%_Xcode         open -a /Developer/Applications/Xcode.app
```

リソースフォークを持つファイルを扱う時にこれらのマクロを利用します。

6.4 Universal Binary

2006 年の 1 月以降 OSXWS はデフォルトで Universal Binary のパッケージを作成しています。
ここでは Universal Binary rpm package の作成方法を簡単に示します。
主な内容は **MacWiki**[32] の UniversalBinary の項目に記してありますので、
合わせてご覧ください。

はじめに

原則として PPC, Intel 双方のバイナリを独立してビルドし
最後に lipo で結合する方法を採ります。

spec file の基本形

冒頭

%define uname_release `(uname -r) 2>/dev/null` を置き、
configure への host 指定に利用します。
BuildArch: fat を指定します。

%prep section

以下の様にして PPC, Intel それぞれに tree を分けます。

[32] http://macwiki.sourceforge.jp/cgi-bin/wiki.cgi

```
%setup -q -c %{name}-%{version}
pushd %{name}-%{version}
%patch1 -p1 -b .fat
popd

mv %{name}-%{version} PPC
cp -rp PPC INTEL
```

%build section

以下の様にして PPC, Intel それぞれ独立にビルドします。
ただし、以下の例は Universal Binary のライブラリが標準で
インストールされている Intel 版 MacOS X 上での記述であり、
PPC 上では `-sysroot /Developer/SDKs/MacOSX10.4u.sdk`
の記述を CFLAGS に追加する必要があります。

```
pushd PPC
CFLAGS="-O2 -arch ppc" \
CXXFLAGS="$CFLAGS" \
%configure --host=powerpc-apple-darwin%{uname_release} \
--disable-dependency-tracking
perl -pi -e 's@-dynamiclib@-dynamiclib -arch ppc@g' libtool
make
popd
pushd INTEL
CFLAGS="-O2 -march=i686 -mtune=pentium-m" \
CXXFLAGS="$CFLAGS" \
%configure --host=i686-apple-darwin%{uname_release} \
--disable-dependency-tracking
perl -pi -e 's@-dynamiclib@-dynamiclib -arch i386@g' libtool
make
popd
```

%install section

以下の様にして Universal Binary を作成します。

```
pushd PPC
mkdir -p ${PWD}-root%{_bindir}
make install DESTDIR=${PWD}-root
popd
pushd INTEL
mkdir -p ${PWD}-root%{_bindir}
```

```
make install DESTDIR=${PWD}-root
cp -fRP COPYRIGHT README VERSION TODO html ..
popd

## Make Universal Binaries
filelist=$(find ./PPC-root -type f | xargs file | sed -e 's,^\./PPC-root/,,g' | \
        grep -E \(Mach-O\)\|\(ar\ archive\) |sed -e 's,:.*,,g' -e '/\for\ architecture/d')

for i in $filelist
do
        /usr/bin/lipo -create PPC-root/$i INTEL-root/$i -output `basename $i`
        cp -f `basename $i` PPC-root/$i
done

# install
mkdir -p %{buildroot}
tar cf - -C PPC-root . | tar xpf - -C %{buildroot}
```

6.5 その他

ライブラリに関する問題

libintl, libiconv 等が読み込まれなかったり、
/usr/include/{glob,regex}.h 等が gnu のものとコンフリクトしたりする事があります。
大抵、ld が multiple definitions of symbol.... や
undefined symbols.... 等とメッセージを出し、
該当する関数名とライブラリ名を表示してくれるので、
出来るだけ MacOS X 側が用意しているライブラリやヘッダファイルを利用する様
にします。

libtool, autotools に関する問題

MacOS X 10.4.4 + Xcode 2.2 では、

libtool	1.5 (glibtool, glibtoolize)
aclocal, automake	1.6.3
autoheader, autoconf	2.59

が用意されています。
MacOS X WorkShop では、automake1.{479}, autoconf-2.13 を用意しています。
これらは必要に応じて、aclocal-1.9 -I /usr/share/aclocal 等として利用します。

libtool を利用する時は、
configure の前で glibtoolize --copy --force とし、

configure の後で `cp -f /usr/bin/glibtool libtool`
とするとうまく行く事があります。

7　インストーラを作る

MacOS X WorkShop 10.4 のインストーラを Tiger 上の Xcode-2.2 で小林が開発した際の備忘録です。
屹度間違いがあるとおもいます。
総合的且つ正確な情報は Apple の **Tools: Software Distribution**[33] をご覧下さい。

インストーラのソース一式は「インストール (Section4.3 参照) 」のページからダウンロード出来ます。
姉妹 tree の作成に役立ててください。
尚、インストーラを作るには
`/Developer/Applications/Utilities/PackageMaker.app`
を使います。

7.1　段取り

一般的にインストーラを作成するのに必要な段取りは以下になります。

1. 作業場所を作る。
2. インストールするファイル類を用意する。
3. インストールする手順を所定の各ファイルに記述する。
4. 「PackageMaker」でインストーラを作成する。
5. 「ディスクユーティリティ」でディスクイメージを作成する。

これらの作業は一寸面倒です。
覚え書き程度に書いていきます。

7.2　作業場所を作る

インストーラを作る作業場には
「インストールするファイル類」と「手順を記したファイル類」を置く場所が必要
です。

MacOS X WorkShop では、ディレクトリ「OSX-WS」を作業場のルート・ディレクトリとし、
「インストールするファイル類」は「OSX-WS/ROOT」に、
「手順を記したファイル類」は「OSX-WS/Resources」に
置いています。

以下これらのディレクトリ構造を基にして記述していきます。
適宜読み替えて下さい。

[33] http://developer.apple.com/documentation/DeveloperTools/Conceptual/SoftwareDistribution/index.html

7.3　インストールするファイル類を用意する

MacOS X WorkShop のインストーラがすべき事は、
最低限の `apt-rpm` 環境をつくる事 です。

インストーラのソース一式[34] の中の make-tree.sh が、必要なファイルを所定の位置にコピーするスクリプトです。`RPMDIR=/Users/tkoba/rpm/RPMS/noarch` を適宜書き換えた上で、「OSX-WS/ROOT」の中で実行して下さい。
このスクリプトの中でしている事は

1. **ディレクトリの作成**
2. **`apt, rpm` バイナリー類のコピー**
3. **基本パッケージ `OSX-{Preferences,X11,system}*` のコピー**

です。
ただし、基本パッケージのコピーの後、
同一パッケージの複数のバージョンが入っていない事を確認してください。
これで、「OWX-WS/ROOT」以下にインストールされるファイル類が準備されます。

> **注意！**
> ファイルのコピー先に**直接 `/etc` を指定しない**で下さい。
> /etc は /private/etc へのシンボリックリンクになっています。
> /etc を指定すると PackageMaker の設定に依っては/etc のシンボリックリンクを消去して
> 新たに /etc ディレクトリを作りファイルをインストールしてしまいます。
> 即ち、**最悪の場合 `MacOS X` が起動しなくなります**。

7.4　インストールする手順を所定の各ファイルに記述する

ソフトをインストールするには、インストールしようとしている環境が適切であるか確認する必要がありますし、
ファイルを所望の位置に置いた後に、某かのお決まりの設定をする必要がある事もままあります。
ここではその様な手順を実現する方法を述べます。
PackageMaker Help に記述がありますが、インストーラが行う手順は以下になります。

1. `InstallationCheck`
2. `VolumeCheck`
3. `preflight`

[34] MacOSX-WS-10.4.2.tar.bz2

4. preinstall or preupgrade
5. (INSTALLER EXTRACTS AND INSTALLS THE PACKAGE'S CONTENTES.)
6. postinstall or postupgrade
7. postflight

MacOS X WorkShop ではこの中の、
InstallationCheck, postinstall, postupgrade を利用
しています。
以下順次説明していきます。

InstallationCheck 「OSX-WS/Resources/InstallationCheck」がスクリプトの実態で、
「OSX-WS/Resources/Japanese.lproj/InstallationCheck.strings」がメッセージ（日本語環境 UTF-8）
の実態です。

ここでは、インストールしようとしている環境が適切であるかを確認します。

している事は、

- MacOS X のバージョンは適切か？
- X11 がインストールされているか？
- Xcode がインストールされているか？

のチェックと、状況に応じたメッセージの表示です。

詳細は **Apple のドキュメント**[35] をご覧下さい。

postinstall, postupgrade 「OSX-WS/Resources/postinstall」がスクリプトの実態で、
「OSX-WS/Resources/postupgrade」は、現在 postinstall へのシンボリックリンクです。

ここでしている事は、

- rpm database の初期化
- 基本パッケージ OSX-{Preferences,X11,system}* のインストール
- ドットファイルを各ユーザーへ配布

です。

最後に、「OSX-WS/{Welcome,ReadMe,License}.rtf」を作っておきます。

[35] http://developer.apple.com/documentation/DeveloperTools/Conceptual/SoftwareDistribution/index.html

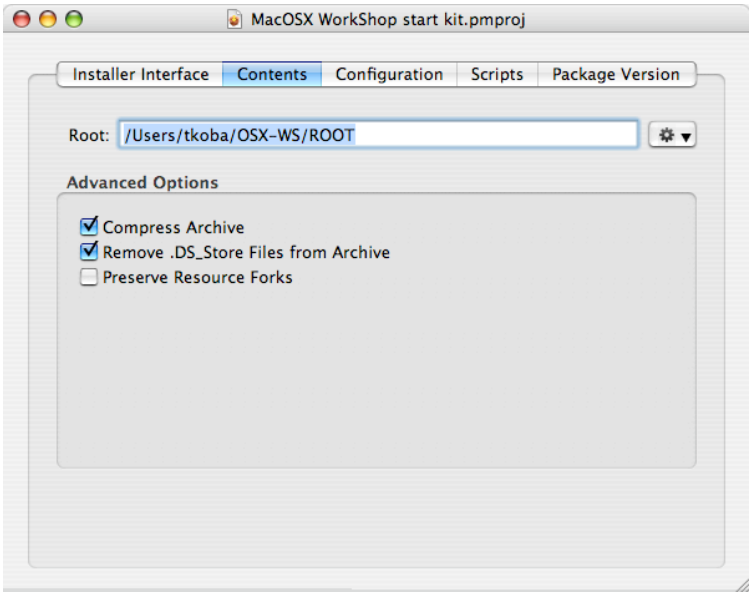

図 1: 「PackageMaker-Contents」

7.5 インストーラを作成する

ファイル類の準備ができたら、PackageMaker を使ってインストーラを作ります。

● PackageMaker を起動すると、「Installer Interface」タグが選択されたウィンドが現れます。
「Title」「Description」に必要事項を記述します。

● 「Contents」タグではインストールするファイルを置いてあるディレクトリ「OSX-WS/ROOT」を指定し、
全体を圧縮する為に「Compress Archive」にチェックを付けます。

● 「Configuration」タグでは以下の設定をしています。

- `Default Lacation`
  インストール先を指定します。
  ここではルートディレクトリ「/」を指定します。
- `Authorization`
  インストール時に必要な権限の指定
  root の権限を与えます。
- `Post-Install Action`
  インストール後の動作を指定します。
  login のし直しで大丈夫な筈ですが、大事を取ってここではリスタートするように指定しています。
- `Flags:  Root Volume Only`

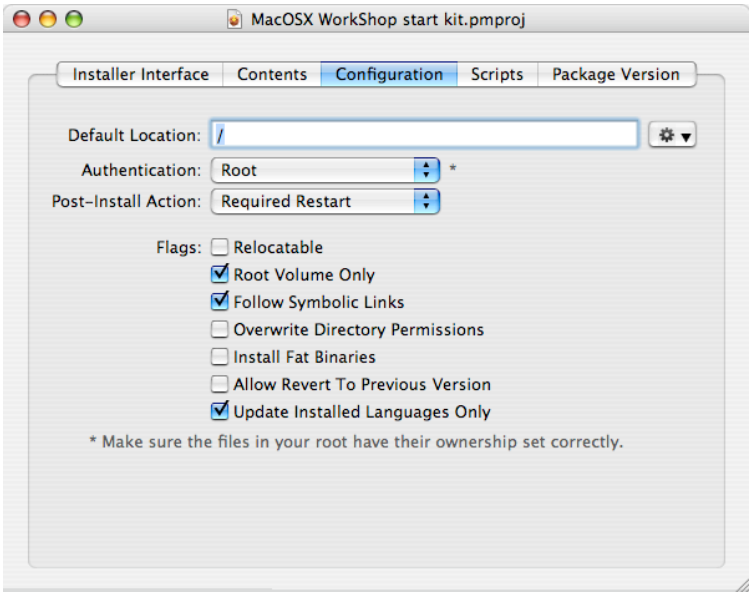

図 2: 「PackageMaker-Configuration」

インストール先のボリュームを起動ボリュームに限るか否かの設定です。
当然チェックを付けておきます。

- `Flags:  Follow Symbolic Links`
  インストール先がシンボリックリンクであった場合に、そのシンボリックリンクを認めるか否かの設定です。
  チェックを付けないとそのシンボリックリンクを消してしまうので、チェックを付けておきます。
  ただし、これはあくまで保険として付けておくもので、インストール・ファイルを置いておく「OSX-WS/ROOT」以下の段階で
  /etc 等シンボリックリンクに向けたディレクトリを作らない事が大事です。

その他のフラグは必要に応じてチェックして下さい。

● 「Scripts」タグではスクリプト・ファイルを置いてあるディレクトリ「OSX-WS/Resources」を指定します。

● 「Package Version」タグでは、その名の通り version を記述します。

● 全部設定し終わったら保存した後に、「Project」→「Build...」でインストーラを作成します。

7.6 ディスクイメージを作成する

これまでの作業でパッケージのインストーラ MacOSX WorkShop start kit.pkg が出来ました。

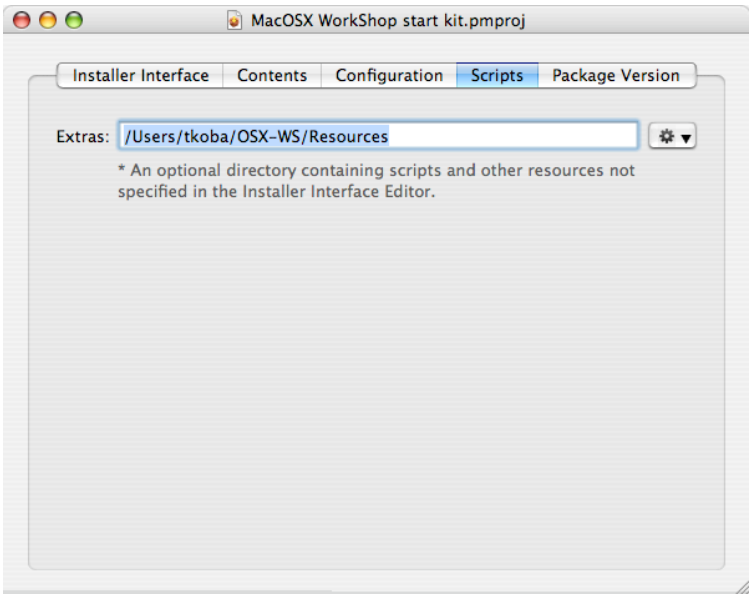

図 3: 「PackageMaker-Scripts」

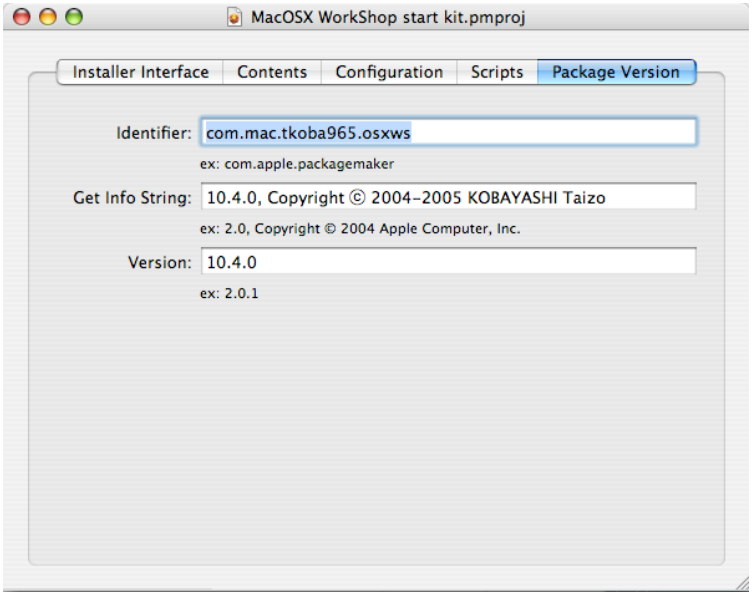

図 4: 「PackageMaker-Version」

後はこれをディスクイメージの中に置いておしまいです。
以下の作業をします。

イメージの作成 「ディスクユーティリティ」を立ち上げて「新規イメージ」ボタンを押し、
ディスクの容量を必要なだけ設定して（私は 7MB にしました）空のイメージを作ります。
空のイメージをマウントして、
そこに ReadMe.rtf と作成したインストーラを入れてマウント解除します。

イメージの圧縮 「ディスクユーティリティ」のメニューから「イメージ」→「変換...」を選択し、
上で作成したイメージを選択します。
「イメージフォーマット」に「圧縮」を選んで保存します。

尚、これらの処理をスクリプトにしてあります。
インストーラのソース **MacOSX-WS-10.4.3.tar.bz2**[36] の中にある make-pkg.sh をご覧ください。

[36] MacOSX-WS-10.4.3.tar.bz2

8　apt-rpm tree を作る

MacOS X WorkShop tree を用意した際の備忘録です。
屹度間違いがあるとおもいます。
間違いを発見されましたらご連絡ください。
また、貴方独自の add-on tree を用意される場合もほぼ同じ手順で可能です。

8.1　段取り

一般に apt-rpm tree を作成するのに必要な段取りは以下になります。

1. tree を置く場所を用意する。
2. パッケージを置く。
3. データベースを作る。
4. apt-line を記述する。

これらの作業は web や anonymous ftp に物を置いた経験があれば簡単です。
覚え書き程度に書いていきます。

8.2　tree を置く場所を用意する

tree は「web」と「ftp」と「local disk」に置けます。

ftp も web も local disk も単純にディレクトリを作るだけなので、
ここでは web に置く方法を述べます。
以下の条件を満たしていれば大丈夫です。

- 数百 MB のファイルを置く容量を確保出来るか？

- サーバの性能は十分か？
  G4 1GHz 程度の性能があれば、実質問題はないでしょう。

これ以降では tree のルートディレクトリを「OSXWS」とします。

8.3　パッケージを置く

MacOS X WorkShop は現時点では Panther 版と Tiger 版があり、
それぞれを別途置かなければ成りません。

現在はそれぞれ「OSXWS/{Tiger,Panther}」の中に置いています。

- ソース・パッケージ
  ソース・パッケージ（*.src.rpm）は「OSXWS/{Tiger,Panther}/SRPMS.main」に置きます。
  このサフィックス「main」はパッケージをカテゴリー分けする際に意味をなします。
  即ち、core, devel, plus 等に分類したい場合はそれぞれ「SRPMS.core」などとすれば良い訳です。
  MacOS X WorkShop では小さな tree ですから main 一つで十分だと判断し、
  カテゴリー分けはしていません。
- バイナリー・パッケージ
  バイナリー・パッケージ（*.{ppc,noarch}.rpm）は「OSXWS/{Tiger,Panther}/ppc/RPMS.main」
  に置きます。

8.4 データベースを作る

tree の実態が置かれたら、データベースを作成します。
MacOS X WorkShop の場合は

```
[Tiger]  genbasedir --progress --bz2only [OSXWS]/Tiger/ppc main
[Panter] genbasedir --progress --bz2only [OSXWS]/Panther/ppc main
```

としています。
当然 [OSXWS] は適切なディレクトリに置き換えてください。
この操作はパッケージを更新する度に必要になりますから、
シェルスクリプトにでもしておくと良いでしょう。

8.5 apt-line を記述する

最後に apt-line を記述する為に apt パッケージを変更します。

```
$ cd ~/rpm/SRPMS
$ apt-get source apt
```

して、apt のソースパッケージを展開します。
次に、~/rpm/SOPURCES/sources.list-0.5.15-osxws に貴方が作った apt-line を記述します。
既にある記述を真似れば難しくないと思います。
先に /private/etc/apt/sources.list を編集して動作を確認しておき、
正しく動作するものを ~/rpm/SOPURCES/sources.list-0.5.15-osxws にコピーするのが良いでしょう。
最後にスペックファイルを編集（リリース番号の更新と変更履歴を記述）したら、

```
$ cd ~/rpm/SPECS
$ rpmbuild -ba apt-0.5-osx.spec
```

で所望の新しい apt パッケージが ~/rpm/RPMS/ppc 以下に作られます。
この新しい apt パッケージも忘れずに貴方の tree に加えてください。

9 Panther 版からの変更点

Tiger 版は Panther 版から以下の変更がなされています。
具体的な仕様の変更は以下の通りです。

- G5 専用の ppc64 パッケージは作らない。(或は作る価値のある物だけに限定)
- ターミナルの設定を「ディスプレイ→ Unicode (UTF-8)」に、「エミュレーション→非 ASCII 文字をエスケープする、のチェックを外す」に変更。
- パッケージの revision を 10.3tk1 から 10.4osx1 へ変更する。

9.1 パッケージについて

- 追加するパッケージ

  - `ImageMagick`
    やはり無いと不便であるから。
  - `synaptic`
    これで GUI でパッケージ管理出来ます！
    関連して gtk2 も用意しました。
    起動 (mlterm 上) とマニュアルの表示は以下で行ってください。

    ```
    $ sudo synaptic
    $ open /usr/local/share/synaptic/html/index.html
    ```

  - `gcc-g95`
    gcc-g77 と排他利用になりますが用意しました。

- 変更するパッケージ

  - `kterm → mlterm`
    locale を ja_JP.UTF-8 へ変更するに伴い移行。
  - `less → lv`
    less は lv への symlink として提供。
  - `teTeX-3 ベースに更新`
    土村さんの ptetex3 ベース
    .emacs.el から yatex に関する記述を .yatex.el へ移した。

  - `ghostscript`
    version は 8.51
    デフォルトフォントをヒラギノに変更

- 削除するパッケージ

- `bzip2`
  システム付属
- `freetype`
  システム付属
- `vim`
  システム付属の vim が multi_byte でコンパイルされているから。

10　パッケージメモ

ここでは主に各パッケージに施した変更や拡張について記します。

全てのパッケージについて記述している訳ではありません。

パッケージの詳細は rpm2html による **RPM 解説データベース (Tiger)**[37] **RPM 解説データベース (Panther)**[38] をご利用ください。

10.1　Emacs 関連

MacOS X WorkShop では、今のところ CarbonEmacs のみを提供しています。
CarbonEmacs に関する情報は **MacEmacs**[39] や、銭谷さんの **Carbon Emacs パッケージ**[40] をご覧ください。
収録しているパッケージ内容は銭谷さんのものを基本としています。
ただし、CarbonEmacs 本体と、Emacs Lisp は別のパッケージにして、
独立して更新やメンテナンスが出来る様に配慮してありまし、
elisp の置き場所もパッケージ内部ではなく `/usr/local/share/emacs/` 以下に変更しています。
自分独自の elisp （但しバイトコンパイルしていないもの）を置く場合には、
`/usr/local/share/emacs/site-lisp/` 以下に置いて下さい。
また、Vine Linux から alternatives を移植してありますから、
XEmacs など他の Emacsen を共存して入れる事も可能（な筈）です。

関連するパッケージは以下になります。

apel Emacs 用の 基礎的な関数を提供するライブラリ

emacs GNU Emacs エディタ (Carbon 版)

emacs-lisps Carbon Emacs 用の便利な Lisp ライブラリ集
　　銭谷さんのパッケージに入っている Lisp を纏めたものです。

emacsen-common Common facilities for all emacsen.

flim Emacsen 用の message に関する表現形式や符号化のためのライブラリです。

[37] ../Tiger/rpm2html/
[38] ../Panther/rpm2html/
[39] http://macemacsjp.sourceforge.jp/
[40] http://home.att.ne.jp/alpha/z123/emacs-mac-j.html

mew Emacs でメールを読むためのインターフェース

semi Emacsen 用の MIME の機能を提供するライブラリ

aspell emacs 上で利用できるスペルチェッカー
「Tools」メニューからスペルチェックを選べば利用出来ます。
コンソールからの利用も可能です。

task-emacs emacs バーチャルパッケージ
このパッケージを apt でインストールすると、次のパッケージが自動でインストールされます。
alternatives emacs emacsen-common emacs-lisps apel flim semi Mule-UCS

yatex 野鳥 (YaTeX) - Yet Another TeX mode for Emacs

10.2 TeX 関連

パッケージの内容は、土村さんと小林が主にメンテナンスしている Vine Linux の teTeX package 群と基本的に同一です。

teTeX 本体 S-JIS で作成しています。
EUC で利用する為に `/usr/local/bin/{platex-euc,ptex-euc}` を同梱しています。
VFlib を必要としない様にしてあります。

previewer 内山孝憲さんが作成されている Cocoa の dvi previewer です。
MacOS X WorkShop では xdvi ではなく、この Mxdvi をデフォルトにしています。
apt-get でインストールすると、必要なフォントのパッケージ Mxdvi-fonts も同時にインストールされます。

TeXmacros tetex-macros tetex で用いる追加マクロパッケージ集です。
次のマクロを収録しています. jsclasses, prosper, epsbox.sty, eclepsf.sty

texmacro-otf 齋藤修三郎さんによる「OpenType Font 用 macro と VF」です。
齋藤氏が配布されているマクロや VF 以外に次の補助ツール類を同梱しています。
`updmap-otf`
dvipdfmx, udvips 等で埋め込むフォントを設定する為のツールです。
`sudo updmap-otf auto` とすると
OTF-Hiragino パッケージがインストールされていればヒラギノを埋め込む様に設定し、
無ければ noFont の設定をします。
`sudo apt-get install task-tetex` としていれば、デフォルトでヒラギノを埋め込みます。
他にも、モリサワ基本 7 書体パッケージ (OTF-Morisawa-basic7) がインストールされていれば、
`sudo updmap-otf morisawa` とすると利用可能になります。
利用方法は `updmap-otf` で表示されます。

font 関連 jvf

makejvf

OTF-Hiragino MacOS X 付属のヒラギノフォントを利用する為の設定パッケージです。

OTF-Morisawa-basic7 購入して MacOS X にインストールされたモリサワ基本７書体 OpenType Fonts を利用する為の設定パッケージです。

OTF-Morisawa-RmSgSmg 購入して MacOS X にインストールされた以下のモリサワ OpenType Fonts
A-OTF-RyuminPro-{Regular,Heavy}.otf,
A-OTF-ShinGoPro-{Regular,Heavy}.otf,
A-OTF-ShinMGoPro-{Regular,Bold}.otf
を利用する為の設定パッケージです。

urw-fonts free で品位の高い 35 の標準 PostSecript Fonts です。

ttfonts-ja free の日本語 TrueType Font である「さざなみフォント」[41] をインストールします。

Mxdvi-fonts Mac 様に内山さんが作られた TEXFonts で、次のフォントがインストールされます。
ComputerModern, amsps-oztex, pxfonts, txfonts

その他 LaTeXiT LaTeXiT[42]
Apple の Keynote 等で数式を扱う際に利用出来ます。

task-tetex TEX 関連パッケージを簡単にインストールするための仮想パッケージです。

dvipdfmx dvi file を PDF file に変換するツールです。
齋藤さんの OTF パッケージに対応しています。

latex2html TEXfile を html file に変換するツールです。
このドキュメントも latex2html を用いて書かれています。

yatex

10.3 X11 関連

X server には Xquartz を、Window Manager には quartz-wm を利用します。
.Xclients や .Xresources では設定できない項目があり、

```
$ defaults write com.apple.x11 xxx yyy zzz
```

その場合はする必要があります。
詳細は

```
$ man Xquartz
$ man quartz-wm
```

で調べてください。

ImageMagick 画像ファイルの表示/処理を行う X のアプリケーション

[41] http://sourceforge.jp/projects/efont/files/
[42] http://ktd.club.fr/programmation/latexit_en.php

Plotmtv X11 上で動作するグラフ作成ツール

Xaw3d A version of the MIT Athena widget set for X.

ghostscript A PostScript(TM) interpreter and renderer.
デフォルトでヒラギノを使用します。

gnuplot A program for plotting mathematical expressions and data.

gv A X front-end for the Ghostscript PostScript(TM) interpreter.

kinput2 kinput2 - kanji input server for X11
ことえりパッチ適用。

mlterm 他言語の表示が可能な X 上のターミナルソフト
「Ctlr + 右クリック」でコントローラが現れます。

openMotif The Open Motif runtime components.

ttfonts-ja Free Japanese TrueType fonts

内容はさざなみフォント[43] です。

urw-fonts Free versions of the 35 standard PostScript fonts.

xgraph11 xgraph - 2D data plotting program (+ hack 9 + color PS + and so on.)

yaplot yaplot - an easy 3D modeller and animator

10.4 開発関連

rpm The RPM package management system.

詳細は spec file を参照してください。

apt RPM を扱える Debian のパッケージツール apt(Advanced Packaging Tool)

static build して strip してあります。

gcc-g77 g77 A GNU Fortran compiler.

HPC MacOS X[44] のものです。

[43] http://sourceforge.jp/projects/efont/files/
[44] http://hpc.sourceforge.net/

gcc-g95 g95 A GNU Fortran compiler.

HPC MacOS X[45] のものです。

gcc-g77 とは排他利用になります。

10.5 System 関連

OSX-system MacOS X の標準ライブラリやツールを rpm system に教える為のパッケージです。

`/private/etc/{profile-osxws,csh.login-osxws,zprofile}` が加えられ `/usr/local/bin, /usr/X11R6/bin` 等にパスを通します。

MacOS X WorkShop インストーラによりインストールされます。

OSX-X11 X11 のライブラリやツールを rpm system に教える為のパッケージです。

MacOS X WorkShop インストーラによりインストールされます。

OSX-Preferences OSX-Preferences パッケージは MacOS X WorkShop の基本システムの一部で、デフォルトのユーザ設定ファイル (.Xresources, .bash_logout, .bash_profile, .bashrc) 等を収録しています。

各ユーザーに配布された設定ファイルを更新する為に、
osxws-upgrade スクリプトを収録しています。
`OSX-Preferences package` の更新に対応する為に個人用の記述は
`.bashmyrc, .cshmyrc, .zshmyrc, .emacs.my.el` を作成し、
その中に記述して下さい。

`/System/Library/User\ Template/Japanese.lproj/` 内にインストールされますから、
新規ユーザー作成時に自動で設定ファイル類がコピーされます。

MacOS X WorkShop インストーラによりインストールされます。

OSX-base rpm system を利用する為に最低限必要なパッケージをインストールする為の仮想パッケージです。

MacOS X WorkShop インストーラを実行した直後に `sudo apt-get install OSX-base` しておく必要があります。

[45] http://hpc.sourceforge.net/

11　スクリーンショット

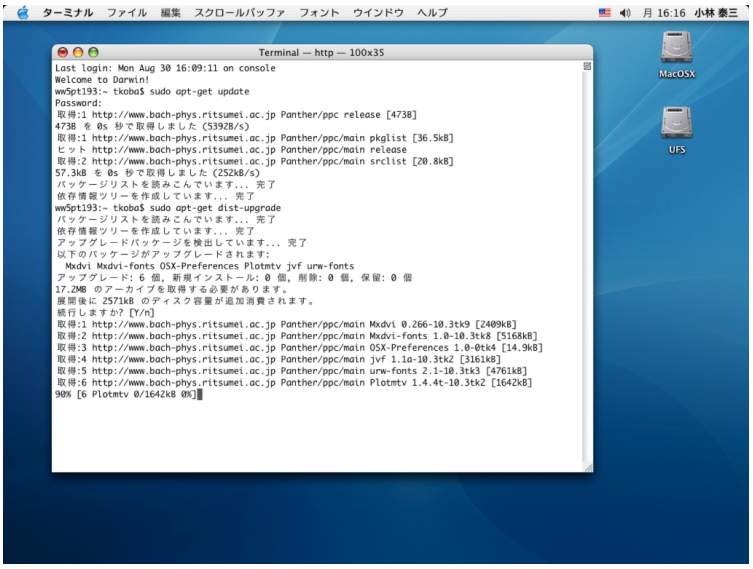

図 5: apt-get でアップデートパッケージをダウンロード中。

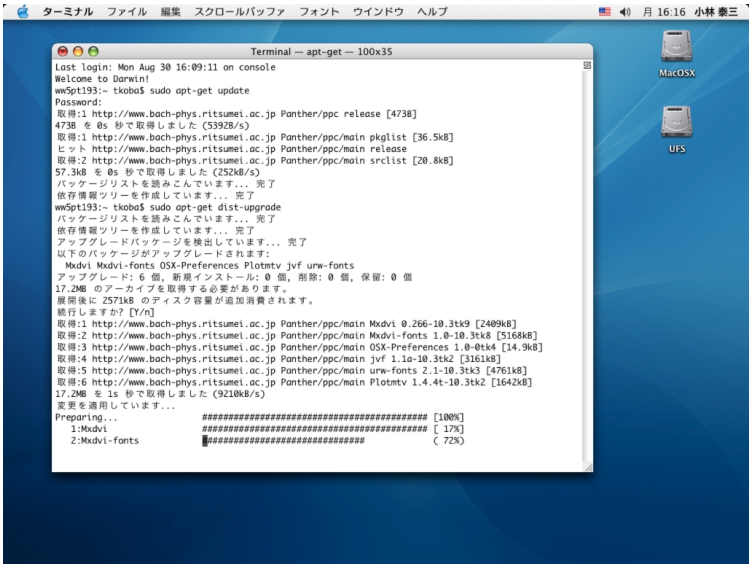

図 6: apt-get でパッケージを更新中。

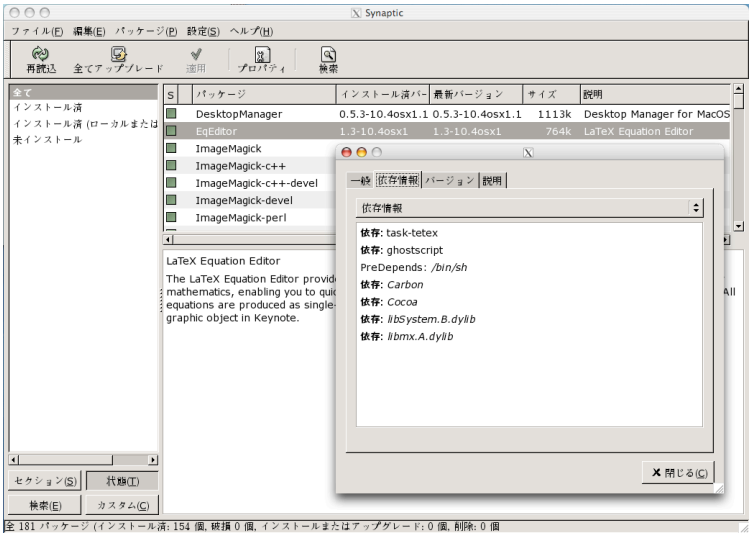

図 7: X11 上の apt-rpm frontend である synaptic

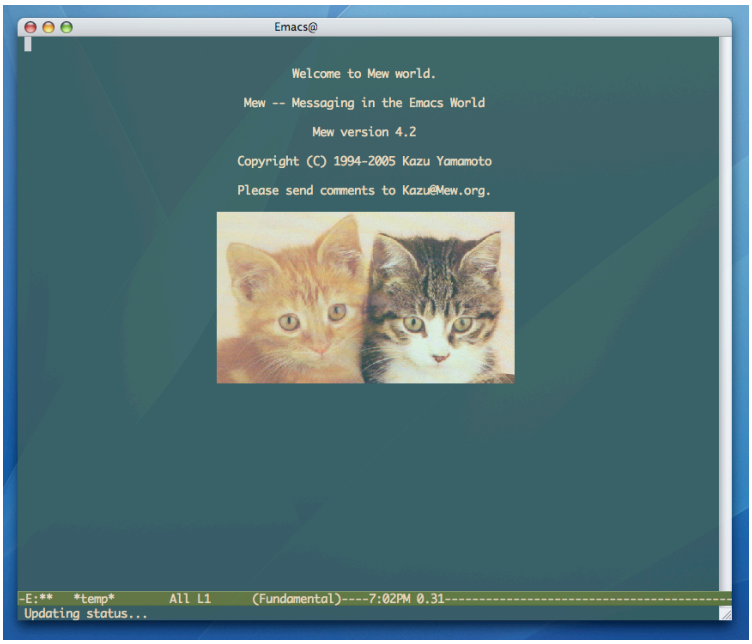

図 8: CarbonEmacs で mew を立ち上げているところ。

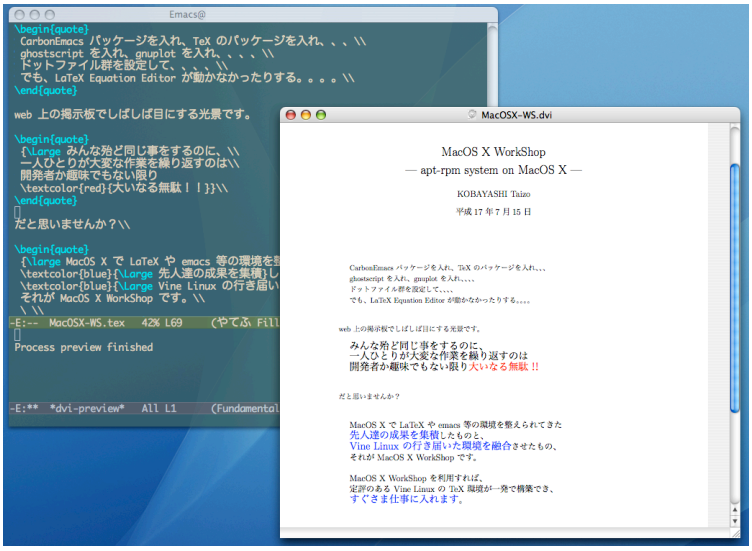

図 9: CarbonEmacs で yatex を用いて LaTeX の文章を書き Mxdvi でプレビューしているところ。Mxdvi からそのまま印刷可能です。

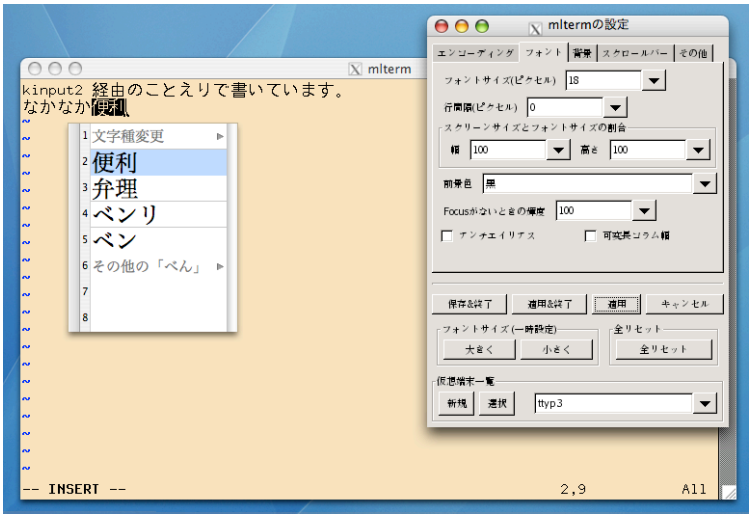

図 10: kterm 上の vim で kinput2 を通して日本語の文章を書いているところ。ことえりパッチを適用してあります。

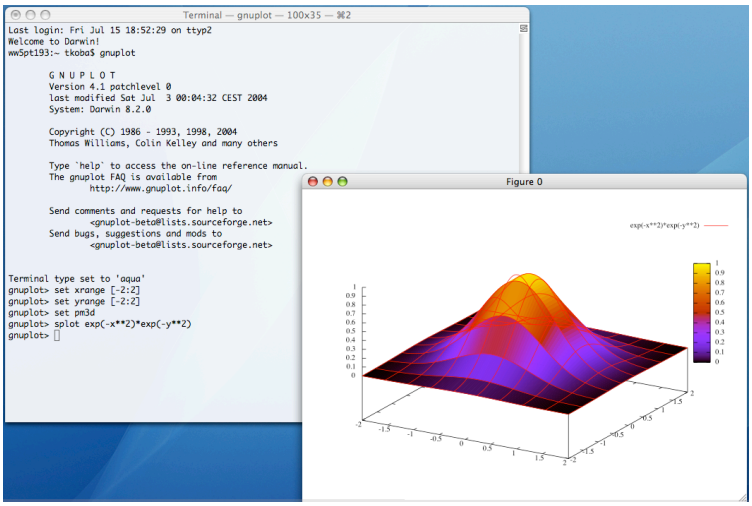

図 11: お馴染み gnuplot です。aquaterm, PDFlib-Lite も同時にインストールされます。

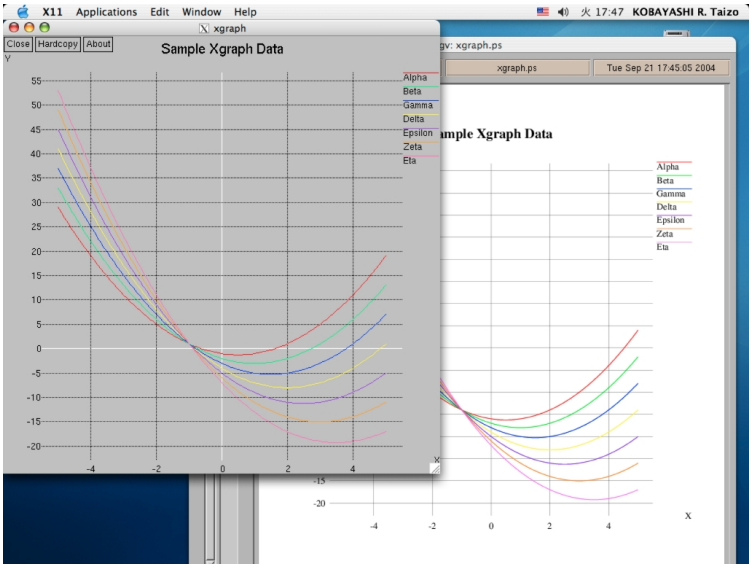

図 12: Vine Linux と同じパッチを適用してあります。Color PS を生成できます。PS ファイルをダブルクリックすれば大抵の場合 PDF に変換でき、プレビューで確認と印刷ができます。

12　既知の問題点と注意点

2006 年 2 月 1 日現在、MacOS X WorkShop には以下の様な問題があることが分かっています.

etc. その他気付いていない問題点があるかもしれません.

13 過去の議論

ここは **Mac Wiki**[46] にて行われた過去の議論の書庫です。

13.1 dot.emacs

- Emacs 初期設定の改正
  - CarbonEmacsPackage など他の Emacs と初期設定を使い分ける目的で、最近全ユーザ共通の領域で一部の初期設定を行うようになりましたが、この方法を止めて、初期設定ファイルは全てホームディレクトリ以下に置き、Emacs ごとに条件分岐で読み込むようにする事を考えています。以下に具体的な案を示します。これを機に.emacs の全体的な見直しを考えていますので、ご意見頂ければと思います。– seto
  - CarbonEmacsPackage との共存、というより、システム標準の emacs との共存を良く考えた方が良いです。何らかの理由で OXSWS の CarbonEmacs が壊れてしまった場合、最後に頼りにするのはシステム標準の Emacs です。システム標準 Emacs を使えなくしてまで独自の設定を放り込むとしたら、逆に大きな責任を背負うことになります。– ぜ
    - 確かに、OSX のアップデートで CarbonEmacs が使えなくなる事は過去に何度もありましたし、そういう時に/usr/bin/emacs を使う可能性はありますね。今回の変更では、そういった Emacs を起動しても初期設定ファイルを読みに行かないのでエラーは出ませんが、同時に文字コードなどの基本設定も読み込みません。それはいいですよね？（実際にエラーがでる項目は utf-8m と日本語メニューバーとタイムスタンプだけ見たいですが）–seto
      - タイムスタンプは (require 'time-stamp) で解決します。また、utf-8m は /usr/bin/emacs では不要です。Terminal.app がそのへんの変換処理をうまくやってくれますので。–ぜ
      - 何時も有り難う御座います。２月の半ばには本業も一段落する見込みですので、その頃に纏めの作業に入ろうと思います。–tkoba
- OSXWS-10.4-3 の emacs 設定ファイル
  - グローバル共通
    OSXWS の Emacs は起動時に/private/etc/emacs-[EMACSVER]/site-start.d/を読みに行きます。 ここでは、upstream で今後も改変が予想される設定と、OSXWS のデフォルト設定を行います。これは、distribution 全体としての整合性や体裁（つまり、「街」としての機能を保つこと）を目的とします。ユーザーが自分の /.*.el を書き損じても、それら全てを無効にすればOSXWS としての CarbonEmacs の機能が一通り利用できる事を目指します。

```
 $ cat /private/etc/emacs-[EMACSVER]/site-start.d/99osxws.el

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  osxws.el for MacOS X [[WorkShop]]
;;            KOBAYASHI Taizo <xxxxxxx@xxxxxx.com>
;; Time-stamp: <07/03/07 15:17:49 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

[46] http://macwiki.sourceforge.jp/cgi-bin/wiki.cgi

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 0 fundamental configurations
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; setting the MacOS X [[WorkShop]] flag
(defconst osxws-emacs-flag t
  "This is Emacs of MacOS X WorkShop.")

;; japanese settings for Carbon Emacs Package
(if
    (equal
     (substring
      (concat
       (shell-command-to-string "defaults read -g AppleLocale") "__") 0 2) "ja")
    (save-excursion
     (require 'utf-8m)
     (set-clipboard-coding-system 'utf-8)
     (set-terminal-coding-system 'utf-8) ; 'euc-jp-unix)
     (set-file-name-coding-system 'utf-8m)
     (set-language-environment 'Japanese)
     (set-default-coding-systems 'euc-jp-unix)
     (set-keyboard-coding-system (if window-system 'sjis-mac 'utf-8))
     (load "menu-tree")))

;;; 前回の編集箇所を記録
(require 'saveplace)
(setq-default save-place t)
(setq save-place-file "~/Library/Application Support/OSXWS/emacs-places.txt")

;; 編集ファイル foo のバックアップファイル foo~ をコピーで作る
(setq backup-by-copying t)

;;; emacsclient サーバを起動
(server-start)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 1 appearance setting
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; tool-bar を消す
(tool-bar-mode nil)
```

```
;;; 括弧の対応表示
(require 'paren)
(show-paren-mode)

;;; mark 領域に色付け
(setq transient-mark-mode t)

;;; フォントのスケールをしない
(setq scalable-fonts-allowed nil)

;;; 行番号を表示する
(line-number-mode t)

;;; ステータスラインに時間を表示する
(if (equal (substring (concat
       (shell-command-to-string "defaults read -g AppleLocale") "__") 0 2) "ja")
      (progn
       (setq dayname-j-alist
            '(("Sun" . "日") ("Mon" . "月") ("Tue" . "火") ("Wed" . "水")
             ("Thu" . "木") ("Fri" . "金") ("Sat" . "土")))
       (setq display-time-string-forms
             '((format "%s 年%s 月%s 日 (%s) %s:%s %s"
                       year month day
                       (cdr (assoc dayname dayname-j-alist))
                       24-hours minutes
                       load)))))
(display-time)

;;; 最終更新日の自動挿入
;;;   ファイルの先頭から 8 行以内に Time-stamp: <> または
;;;   Time-stamp: " " と書いてあれば、セーブ時に自動的に日付が挿入されます
(require 'time-stamp)
(if (not (memq 'time-stamp write-file-functions))
    (setq write-file-functions
   (cons 'time-stamp write-file-functions)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 2 keyboard/keybindig
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; コマンド、コントロール、オプションの各キーを
;;; alt, control, hyper, meta, super
;;; の各キーとして利用
```

```
(setq mac-command-modifier 'alt)
(setq mac-control-modifier 'control)
(setq mac-option-modifier 'meta)

(cua-selection-mode t)

;;; [Home] Key と [End] Key を従来の動作に戻す
(define-key global-map [home] 'beginning-of-buffer)
(define-key global-map [end] 'end-of-buffer)

;;; mac-key-mode
;; (OSX アプリケーションのショートカット)
(when (eq window-system 'mac)
  (require 'mac-key-mode)
  (mac-key-mode 1))

;; C. fix: Unicode => Japanese mapping
;; Thanks to saiki-san (see [macemacsjp-users 870])
;; register circle around digits to cjk table (by Ando-san)
(defadvice utf-translate-cjk-load-tables
  (after my-ad-circled-digit activate)
  (dotimes (i 20)
    (let ((unicode (+ #x2460 i))
   (char (+ 54433 i)))
      (if (utf-translate-cjk-substitutable-p unicode)
   (puthash unicode char ucs-unicode-to-mule-cjk))
      (puthash char unicode ucs-mule-cjk-to-unicode))))
;; prevent to use half-width marks (by Nanba-san)
(utf-translate-cjk-set-unicode-range
 '((#x2e80 . #xd7a3)
   (#xff00 . #xffef)
   (#xa7 . #xa7)                        ;
   (#xb0 . #xb1)                        ;
   (#xb4 . #xb4)                        ;
   (#xb6 . #xb6)                        ;
   (#xd7 . #xd7)                        ;
   (#xf7 . #xf7)                        ;
   (#x370 . #x3ff)                      ; ギリシャ
   (#x400 . #x4ff)                      ; キリル
   (#x2000 . #x206f)                    ; 一般句読点
   (#x2103 . #x2103)                    ; ℃
   (#x212b . #x212b)                    ; Å
   (#x2190 . #x21ff)                    ; 矢印
```

```
   (#x2200 . #x22ff)                    ; 数学記号
   (#x2300 . #x23ff)                    ; 技術記号
   (#x2460 . #x2473)                    ; 円囲み数字
   (#x2500 . #x257f)                    ; 罫線
   (#x25a0 . #x25ff)                    ; 幾何学模様
   (#x2600 . #x26ff)                    ; その他の記号
   ))
;; C. end

;; D. fix yen key problem on JIS keyboard
;; Ando-san's code (see [Macemacsjp-users 1126])
(define-key global-map [2213] nil)
(define-key global-map [67111077] nil)
(define-key function-key-map [2213] [?\\])
(define-key function-key-map [67111077] [?\C-\\])

(define-key global-map [3420] nil)
(define-key global-map [67112284] nil)
(define-key function-key-map [3420] [?\\])
(define-key function-key-map [67112284] [?\C-\\])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 3 shell-command の設定
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; password の入力を隠す
(add-hook 'comint-output-filter-functions
 'comint-watch-for-password-prompt)

;;; escape sequence
(autoload 'ansi-color-for-comint-mode-on "ansi-color"
 "Set `ansi-color-for-comint-mode' to t." t)
(add-hook 'shell-mode-hook 'ansi-color-for-comint-mode-on)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 4 橋本さんの inline-patch 設定
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(when (eq window-system 'mac)
  (add-hook 'minibuffer-setup-hook 'mac-change-language-to-us)
  (mac-add-ignore-shortcut '(control))
  (mac-set-input-method-parameter 'japanese 'cursor-color "red"))
```

```
;; Command-Space で起動します。
(global-set-key [(alt \ )] 'toggle-input-method)
;; Shift-Space で起動します。
(global-set-key [?\S-\ ] 'toggle-input-method)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 5 [[CarbonEmacs]] ウィンドウモードの設定
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(when (eq window-system 'mac)

  ;; fixed-width-fontset (carbon-font)
  (require 'carbon-font)
  ;; if display-height is less than 900, set font size 12pt.
  (if (< (display-pixel-height) 900)
    (setq fontsize 12)
    (setq fontsize 14))
  (fixed-width-set-fontset "hiramaru" fontsize)

  ;; color-theme
  ;; M-x color-theme-?によってカラーテーマを変更できます。
  (require 'color-theme)
  (color-theme-dark-blue2)

  ;; 透過ウィンドウ Transparency3
  (add-to-list
   'default-frame-alist
   '(alpha . (90 40))) ;; (alpha . (<active frame> <non active frame>))
  (setq frame-alpha-lower-limit 20)

  ;; アンチエイリアス
  (setq mac-allow-anti-aliasing t)

  ;;----------------------------------------------------------
  ;; smart-dnd (ドラッグ＆ドロップ)
  (require 'smart-dnd)

  ;; yahtml-mode:
  (add-hook
   'yahtml-mode-hook
   '(lambda ()
      (smart-dnd-setup
       '(
```

```
  ("\\.gif\\'" . "<img src=\"%R\">\n")
  ("\\.jpg\\'" . "<img src=\"%R\">\n")
  ("\\.png\\'" . "<img src=\"%R\">\n")
  ("\\.css\\'" . "<link rel=\"stylesheet\" type=\"text/css\" href=\"%R\">\n" )
  ("\\.js\\'"  . "<script type=\"text/javascript\" src=\"%R\"></script>\n" )
  (".*" . "<a href=\"%R\">%f</a>\n")))))

  ;; yatex-mode:
  (add-hook
   'yatex-mode-hook
   '(lambda ()
      (smart-dnd-setup
       '(
  ("\\.tex\\'" . "\\input{%r}\n")
  ("\\.cls\\'" . "\\documentclass{%f}\n")
  ("\\.sty\\'" . "\\usepackage{%f}\n")
  ("\\.eps\\'" . "\\includegraphics[clip]{%r}\n")
  ("\\.ps\\'"  . "\\includegraphics[clip]{%r}\n")
  ("\\.pdf\\'" . "\\includegraphics[clip]{%r}\n")
  ("\\.jpg\\'" . "\\includegraphics[clip]{%r}\n")
  ("\\.png\\'" . "\\includegraphics[clip]{%r}\n")
         ("\\.bst\\'" . "\\bibliographystyle{%n}\n")
  ("\\.bib\\'" . "\\bibliography{%n}\n")))))

  ;; C/C++ mode:
  (add-hook
   'c-mode-common-hook
   '(lambda () (smart-dnd-setup '(("\\.h\\'" . "#include <%f>")))))
  ;;-------------------------------------------------------------
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 6 その他の設定
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; Yatex
(if (file-exists-p "~/.yatex.el")
      (load-file "~/.yatex.el"))

;; Mew 5.2  -  Messaging in the Emacs World
(autoload 'mew "mew" nil t)
(autoload 'mew-send "mew" nil t)
```

```
(if (file-exists-p "~/.mew.el")
      (load-file "~/.mew.el"))

;;; バッファの最後で newline で新規行を追加するのを禁止する
(setq next-line-add-newlines nil)

;;; 画面最下行で [↓] を押したときのスクロール数
(setq scroll-step 1)



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Local Variables:
;; mode: emacs-lisp
;; buffer-file-coding-system: junet-unix
;; End:
```

– ローカル共通

.emacs では条件分岐のみを行います。他の Emacs に別の初期設定を行いたい場合もここから読み込むように設定します。

```
 $ cat ~/.emacs

 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; 初期設定ファイルの読み込み
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(cond ((boundp 'osxws-emacs-flag)
       ; Mac OS X WorkShop 用初期化ファイル
       (load-file "~/.emacs_osxws.el")
       (load-file "~/.custom_osxws.el"))
      (t
       ; その他の Emacs 用初期化ファイル
       (if (file-exists-p "~/.emacs.el")
    (load-file "~/.emacs.el"))
       (if (file-exists-p "~/.custom.el")
    (load-file "~/.custom.el")))
)
```

– OSXWS の Emacs 専用 custom 用ファイル

```
$ cat ~/.custom_osxws.el

 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  .custom for MacOS X [[WorkShop]]
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(setq custom-file "~/.custom_osxws.el")
```

– OSXWS の Emacs 専用 .emacs

ユーザーは設定をここで行います。ユーザーが弄りそうな設定をサンプルの形で提供します。

```
$ cat ~/.emacs_osxws.el

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; ~/.emacs_osxws.el
;;  .emacs for MacOS X [[WorkShop]]
;; Time-stamp: <07/03/07 16:41:26 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; 初期設定ファイルの指定
(setq user-init-file "~/.emacs_osxws.el")

;;; C-h と Backspace の入れ替え
;;; 前一文字削除が Ctrl + h に割当てられます
;(keyboard-translate ?\C-h ?\C-?)
;(global-set-key "\C-h" nil)

;;; コマンド、コントロール、オプションの各キーを
;;; alt, control, hyper, meta, super
;;; の各キーとして利用
;(setq mac-command-modifier 'alt)
;(setq mac-control-modifier 'control)
;(setq mac-option-modifier 'meta)

;;; gz ファイルも編集できるように
;(auto-compression-mode t)

;;; 一行が 80 字以上になった時には自動改行する
;(setq fill-column 80)
;(setq text-mode-hook 'turn-on-auto-fill)
;(setq default-major-mode 'text-mode)

;;; 起動時に message を表示しない
;(setq inhibit-startup-message t)

;;; visible-bell
;(setq visible-bell t)

;;; 行番号を表示する
```

```
;(line-number-mode t)

;;; mule/emacs -nw で起動した時にメニューバーを消す
;(if window-system (menu-bar-mode 1) (menu-bar-mode -1))

;;; バッファの最後で newline で新規行を追加するのを禁止する
;(setq next-line-add-newlines nil)

;;; 画面最下行で [↓] を押したときのスクロール数
;(setq scroll-step 1)

;;; mark 領域に色付け
;(setq transient-mark-mode t)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; [[CarbonEmacs]] ウィンドウモードの設定
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;(when (eq window-system 'mac)

  ;; fixed-width-fontset (carbon-font)
  ;(require 'carbon-font)
  ;(fixed-width-set-fontset "hiramaru" 14)

  ;; color-theme
  ;; M-x color-theme-?によってカラーテーマを変更できます。
  ;(require 'color-theme)
  ;(color-theme-dark-blue2)
  ;(color-theme-ld-dark)
  ;(color-theme-gnome2)
  ;(color-theme-wheat)

  ;; 透過ウィンドウ Transparency3
  ;(add-to-list
  ; 'default-frame-alist
  ; '(alpha . (90 40))) ;; (alpha . (<active frame> <non active frame>))
  ;(setq frame-alpha-lower-limit 20)

  ;; アンチエイリアス
  ;(setq mac-allow-anti-aliasing t)
;)
;; End:
```

- 変更点

– 議論する項目

* (color-theme-ld-dark)

  - デフォルトのカラーテーマですが、黒背景に白文字はコントラストが強すぎて目に負担が大きいと思って、青中心の配色のものを選びました。これも個人の好みが出るところですので、何も設定せずに.emacs.my.el に書いてもらうというのでも良いかもしれません。–seto
  - 色はいちばん個人の好みが出るところなので、CarbonEmacsPackage では絶対に手をつけないことにしています。OSXWS が汎用ディストリビューションを目指すのでしたら、色には手を付けずに、ユーザーのカスタマイズに任せるべきだと思います。– ぜ
  - Distribution としての OSXWS のポリシーを反映するものにすべきでしょう。単体の CarbonEmacsPackage では銭谷さんの仰言る通りカスタマイズすべきではないと思います。それはモデルハウス的なものであるからです。一方 Distribution の一部として提供される場合には「街」の景観に溶け込むようにデフォルトの設定を行うべきです。Vine Linux も emacs には相当のカスタマイズを施していますが、その事によりトラブルが発生した事はありません。– tkoba

* ;;; モード行にカレントディレクトリを表示

```
(add-to-list 'global-mode-string '("" default-directory "-"))
```

  - OSX の場合、時計がメニューバーにあるので、モード行は別の情報にしてもよいかと思い、カレントディレクトリを表示さあせることを考えていましたが、メニューバーに日付はでないということで、検討しています。–seto

* ;;; フォントのスケールをしない

```
(setq scalable-fonts-allowed nil)
```

  - これはどういう効果があるのでしたっけ？
  - 効果が認められないのでトラブルを避ける為に外しました。
  - この設定がないと mew のヘッダーの文字が潰れて見難くなるので復活させました。

– 追加した項目

* ;;; mac-key-mode
  ;; (OSX アプリケーションのショートカット)

```
(when (eq window-system 'mac)
 (require 'mac-key-mode)
 (mac-key-mode 1))
```

  - 以前から、option が Meta キーに割り当てられていましたが、そのように設定した状態では、Command キーがほとんど使われていない状況になっているので、それを有効利用するために追加しました。 mac-key-mode を使っていても、Emacs の標準的な操作にかわりがないので問題ないかと思います。–seto

* ;;; ベル音を鳴らさない

```
(setq visible-bell t)
```

  - 候補として追加しましたが、かならず必要な設定ではないかも知れません。–seto

* ;; yatex-mode:

```
(add-hook
  'yatex-mode-hook
  '(lambda ()
     (smart-dnd-setup
      '(
("\\.tex\\'" . "\\input{%r}\n")
("\\.cls\\'" . "\\documentclass{%f}\n")
("\\.sty\\'" . "\\usepackage{%f}\n")
("\\.eps\\'" . "\\includegraphics[clip]{%r}\n")
("\\.ps\\'"  . "\\includegraphics[clip]{%r}\n")
("\\.pdf\\'" . "\\includegraphics[clip]{%r}\n")
("\\.jpg\\'" . "\\includegraphics[clip]{%r}\n")
("\\.png\\'" . "\\includegraphics[clip]{%r}\n")
("\\.bst\\'"  .  "\\bibliographystyle{%n}\n")
("\\.bib\\'"  .  "\\bibliography{%n}\n")))))
```

  - YaTeX での smart-dnd に画像は clip オプションもあった方が無難ということで加えました。また BibTeX の利用を考え .bst、.bib ファイルの挿入も加えました。

– 取り除いた項目

* ;; gz ファイルも編集できるように

```
(auto-compression-mode t)
```

  - デフォルトで t なので不要です。– ぜ

* ;; このファイルに間違いがあった場合に全てを無効にします

```
(put 'eval-expression 'disabled nil)
```

  - この行は「.emacs に間違いがあった場合に全てを無効」する設定なのですか？ とてもそのようには見えませんが... – ぜ
  - ありがとうございます。気がつきませんでした。また、これがあってもなくても、eval-expression コマンドの振る舞いは変わらないようですから取り除きました。 –seto

* ;;; マクロサーチパスの追加

```
;;;  /lib/emacs 以下にユーザ用の *.el, *.elc を置くことができます
(setq load-path (append '("~/lib/emacs") load-path))
```

  - 管理者権限のないユーザの利用を前提に考えているので、ユーザが lisp を追加するにはホーム以下のどこかに置かなければなりません。しかし、その設定を最初からしておく必要はないかもしれません。–seto
  - 中級者は自分の .emacs.el (.emacs.my.el) に同じ設定を書くでしょうから、不要だと思います。– ぜ

* ;;; active でない window の空 cursor を出さない

```
(setq cursor-in-non-selected-windows nil)
```

  - ウィンドウの active、non active は、透明度を変更するため、カーソルに関してはデフォルトの設定のままでよいと判断しました。–seto

* ;;; C-h と Backspace の入れ替え

```
;;; 前一文字削除が Ctrl + h に割当てられます
```

`(keyboard-translate ?\C-h ?\C-?) (global-set-key "\C-h" nil)` ;;; Command-? を help にする

`;(global-set-key [(alt \?)] 'help)`

- もともとのキーバインドである help は非常に重要な機能なので、最初から変更されているとストレスに感じるユーザが少なからずいると想像していました。Cocoa の C-h は Backspace ですから悩ましいですけど。–seto

– コメント

* 2008-01-23 (Wed) ぜ：ディスプレイサイズのところは (display-pixel-height) を使えば簡単になると思います。
  - ををっ！ 有り難うございます。これですっきりします！！ -tkoba
* 2007-03-08 (Thu) 11:00:23 tkoba：皆様、様々なご教示を戴き有り難う御座います。現時点でのテスト環境に更新致しました。引き続きご意見を宜しくお願い致します。
* 2007-03-08 (Thu) 08:40:16 ぜ：あと、(setq scalable-fonts-allowed nil) が carbon-font と共存できるかどうかが心配です。
  - 何時も有り難う御座います。効果が不明でトラブルの元になりかねないので外しました。–tkoba
* 2007-03-08 (Thu) 08:38:26 ぜ：それから、 /Library/Application Support/OSXWS/ は、OSXWS のインストーラーが作成するのでしょうか？
  - 現状では OSX-Preferences に含まれている dot file 管理ツールの osxws-upgrade 内で行っています。-tkoba
* 2007-03-08 (Thu) 08:36:46 ぜ：「アプアランス」ではなく「アピアランス」だと思います。
* 2006-03-23 (Thu) 10:57:02 seto：expand-file-name で” ”などを展開してから使っていますが、それがなくても問題なく動作するようです。現在の Emacs では必要ないと考えてよいでしょうか？
  - load-file を使えば、expand-file-name は不要です。load-file 関数の定義は次のようになっていますから... - ぜ

```
(defun load-file (file)
"Load the Lisp file named FILE."
;; This is a case where .elc makes a lot of sense.
(interactive (list (let ((completion-ignored-extensions
  (remove ".elc" completion-ignored-extensions)))
     (read-file-name "Load file: "))))
(load (expand-file-name file) nil nil t))
```

  - 勉強になります。load-file で統一しました。–seto
* 2006-03-21 (Tue) 17:48:03 ぜ：僕は (setq mac-allow-anti-aliasing t) をオンにしています。関連して [Macemacsjp-users 791] も参考になります。 /lib/emacs はすべてのユーザーに必要でしょうか？
  - ありがとうございます。anti aliasing を nil にして見ていたものは中途半端なものだったのですね。デフォルトでは t にしておきます。（そして個人的には、閾値を適当に設定しようと思います。）–seto

- 確かに自分で lisp を持ってくる人は、自分で好きな場所に置いてパスを通しますよね。迷っている項目に上げておくので、他の方の意見も頂けたらと思います。–seto
- Mac の流儀に従うと、 /Library/Emacs ではないでしょうか？ –fu7mu4
- 確かにそうですね。取りあえず、この設定に関しては取り除いておきますので、問題があれば言ってください。–seto
- CarbonEmacsPackage では /Library/Application Support/Emacs をサポートしていますが、次版から削除する予定です。暗黙の load-path を増やすと、いろいろ混乱が起きて厄介です。–ぜ

* 2006-03-21 (Tue) 01:06:31 seto： 個人的には私も C-h は help ですから、デフォルトのままがよいと思います。現行バージョンからは変更になりますがよろしいでしょうか？ 長期的に考えて、悩ましいところは Emacs デフォルトのまま手を付けない方針にしたいと思います。
* 2006-03-20 (Mon) 13:39:22 ぜ： elisp 開発者は C-h k, C-h f, C-h v などを頻繁に使います。迷ったときは手をつけないのがいちばんかと思います。
* 2006-03-20 (Mon) 11:32:15 seto： デフォルトで C-h を Backspace にするべきかどうか悩みます。デフォルトでバインドされている help もよく使う機能なので、そのままでも良さそうな気がします。nw モードでも、delete キーで Backspace できます。

13.2 10.4-3 公開迄

- carbon emacs で dired を使用するとメニューのところが文字化けするのですが、ご確認いただけないでしょうか。よろしくお願いいたします。 –きんちゃん 2007 年 7 月 23 日 (月) 22:09 (JST)
  - ご報告有り難う御座います。これは menu-tree.el と menu bar の encode が合っていないからなのですが、menu bar の encode を utf-8 にする方法が今一よく判りません。来週辺りで何らかの処置をします。暫くお待ちください。–tkoba
  - emacs-lisps-1.3-10.4osx2.3 (7/24) にて暫定対応致しました。 –tkoba
  - (setq menu-tree-coding-system 'utf-8) でしょうか？ 試していませんが。今忙しいのでまた報告します。–Fu7mu4
  - 上のでは変化ありませんでした。/Application/Emacs.app/Contents/MacOS/Emacs -nw のように-nw つけて Terminal 上で起動すると正しく表示されますね。–Fu7mu4
- sudo apt-get update と sudo apt-get upgrade した際に保留 package が多いので、sudo apt-get dist-upgrade を実行すると

```
libintl.3.dylib is needed by (installed) imlib-cfgeditor-1.9.14-10.4osx2.fat
E: Transaction set check failed
E: Handler silently failed
```

というエラーが発生しました。どう対処するのが正しいのでしょうか？

```
sudo apt-get --fix-broken remove imlib
```

の結果、W: トランザクション処理中にエラーが発生しましたという警告がでます。(imlib-cfgeditor を remove しても同じ結果です。)

```
sudo apt-get --fix-broken reinstall imlib-cfgeditor
```

と、

```
sudo apt-get upgrade
```

しておきましたが、これで正しいですか？ 大量に保留 package があるのでなにかおかしいのではと思ってます。–Fu7mu4 2007 年 6 月 11 日 (月) 23:00 (JST)

- ご報告を有り難う御座います。Transaction Error は、普通、%pre %post での処理に失敗すると出ます。ですが、imlib-cfgeditor には %pre %post 共にありませんので、多分、rpm の db が何らかの拍子に壊れたのでしょうね。その様な時の対処法は以下になります。

  ```
  $ sudo rpm --rebuilddb
  ```

  –tkoba
- ありがとうございます。rpm の db だったのですか。–Fu7mu4 保留された package:

  ```
  a2ps apt apt-devel aspell aspell-el atk emacs-lisps gawk gdk-pixbuf gettext
  glib2 gnupg gtk+ gtk2 lftp mlterm pango popt rpm rpm-build rpm-devel
  rpm-libs w3m w3m-img
  ```

  –Fu7mu4
- 古い gettext (libintl.3) に依存した imlib を抜けない状況なので、これらの packages が保留されるのは正常です。–tkoba
- 自己レスです。

  ```
  sudo rpm -e --nodeps imlib-cfgeditor
  sudo apt-get upgrade
  sudo apt-get dist-upgrade
  ```

  で解決しました。これでよいかどうかは不明ですが、保留 package はなくなりました。–Fu7mu4
- 大丈夫だと思いますが、念のために rpm db の再構築 ($ sudo rpm –rebuilddb) を掛けておいてください。imlib は gtk-1.x 用の古いライブラリでもあり、Leopard 版では obsolete したいですね。–tkoba
- 了解です。rpm db の再構築をしました。–Fu7mu4

- 不安定な状況にしたくないので update していません。よければ 10.4-2 の stable tree と 10.4-3 の unstable tree にわけてリリースしてくれないでしょうか？ –Fu7mu4 2007 年 5 月 8 日 (火) 13:06 (JST)
  - そうしたいのは山々なのですが、、一人で１００以上のパッケージ環境を面倒見ているのでなかなか厳しいです。(Fu7mu4 さん、stable tree のメンテナーになりません？) とは云うものの、実際には現状は dot.emacs 周辺の変更がされているだけで、更新に気をつけて戴ければ仕事でも十分に使えるのですが、念のために unstable としています。新規インストールの場合には全く問題ありません。TeX や a2ps 等の商売道具の UTF-8 化はまだこれからなので、これらを tree に入れる時に addon unstable tree として切り分けるのもありかも知れませんね。–tkoba

  - TeX と emacs のデフォルト文字コードを UTF-8 にする際に addon の test tree に切り分ける事にします。unstable の宣言を撤回しました。–tkoba

- 2007-05-07 (mon) 01:19:08 きんちゃん： 奥村さんのクラスファイル（例えば jsarticle）を使って、emacs から yatex でコンパイルをしようとするとエラーになります。ターミナルから platex hoge.tex とすればコンパイルできたので、.yatex.el のコマンドを platex-euc から platex に書き換えたところコンパイルが通るようになりました。ただ、そうすると日本語が表示されなくなってしまいます。

  - ご報告を有り難う御座います。私自身 emacs と yatex と jsarticle で毎日仕事をしていますが不具合はありません。エラーメッセージやソース等の条件が不明ですので憶測ですが、ファイルが sjis になっているようです。emacs で生成するファイルのデフォルト文字コードを sjis に変更したり、jsarticle.cls をご自分でダウンロードして sjis として利用されているのではないでしょうか。–tkoba
  - ということは私に固有の事例だということですね。失礼いたしました。特に tex 環境をいじったりはしていないので原因はわからないのですが、再インストールしてみようと思います。–きんちゃん
  - 再インストールをしてみたところ無事に動くようになりました。どうもありがとうございました。–きんちゃん

- 2007-03-12 (Mon) 15:15:08 fu7mu4： gmp ライブラリは clamav-update.pl のドキュメントによると intel Mac で build 可能だそうです。私は intel マシンを所持していないので、試していません。http://clamav-update.sourceforge.jp/docs/FreshClamDaemon/chapter3.html

  - 何時も有り難う御座います。手元で Universal Binary でパッケージングしていたものを tree に上げ忘れていました。これから tree 全体の見直しをぼつぼつと始めます。お気付きの点などを引き続きご指摘戴ければ幸いです。–tkoba

- 2007-03-12 (Mon) 13:52:30 たちゃな：pTeX のデフォルトエンコードの件ですが、ptetex3-20070218 以降のバージョンには nkf でソースを前処理する機能が備わっており、この機能を使えばデフォルトエンコードを utf-8 に変更したとしても、移行時の混乱を最小化できるのではないかと思います。（エレガントさには若干欠けるかも知れませんが。）ちなみに、Fink ではこの機能を活用し、先日 utf-8 への移行を敢行されたようです。MacPorts ではユーザがデフォルトエンコードを指定できるため、特に急いてはおりませんが、upTeX という面白いものが出てきましたので、考慮はしております。

  - 早速のコメントを有り難う御座います。奥村さんが土村さんの ptetex3 を用いて utf-8 で「美文書」を書かれたのを知り、utf-8 への移行を考えるようになりました。あと一月ほどは時間があると思いますので、皆様のご意見を戴きながら模索していきたいと思います。–tkoba

- 2007-01-11 (Thu) 22:02:18 fu7mu4： emacs-22.0.92 の info ディレクトリ/usr/share/info/emacs-22.0.92/に dir ファイルが欠けているために Emacs 上で M-x info ができません。どうしたらいいのでしょうか？ Emacs の Package に入っていないようです。

  - 自己レスです。/usr/share/info/emacs-22.0.50/に dir ファイルがありました。以下のようにコピーすると Info が表示できました。–fu7mu4

    ```
    sudo cp /usr/share/info/emacs-22.0.50/dir /usr/share/info/emacs-22.0.92/dir
    ```

  - 何時もご報告を有り難う御座います。この問題はパッケージの%post セクションでの install-info の扱いに原因がありました。直しておきました。–tkoba

- 2006-12-31 (Sun) 16:59:50 ぜ：↓下記の件、aastex.cls を TEXMF/latex/misc に置くだけのようです。小川版パッケージには追加していただきました。

  - 何時も有り難う御座います。texmacro-aastex として収録しようと思います。–tkoba

- 2006-12-31 (Sun) 12:37:32 ぜ：AASTeX を収録されてはどうでしょうか？ Astrophys. Journal を書くときに必要なので。

  - http://www.journals.uchicago.edu/AAS/AASTeX/
  - http://tug.ctan.org/tex-archive/macros/latex/contrib/aastex/

- 2006-12-24 (Sun) 12:07:45 fu7mu4：LaTeXiT 1.11.0 がリリースされましたので version up をお願いします。

  - お知らせ有り難うございます。早速更新しておきました。–tkoba

- 2006-12-23 (Sat) 05:49:39 ぜ：10.5 を機に、初期設定類に手を加えない（/usr/local/ 以外を改変しない）インストール方法もサポートして下さると嬉しいです。私としては労せず gnuplot, maxima をインストールできるパッケージ集があれば魅力なので。

  - そうですね。。ソフト開発が家作りだとするとディストリ開発は街作りのようなもので、(/usr/local/ 以外を改変しない）とすると「下水道は造ってはいけない」ぐらいの制約になり、なかなか難しいのですが、皆さんの知恵をお借りして独立した Cocoa appli あたりから模索すると突破口が見えるかもしれませんね。これは Leopard が出た暁に改めて議論しましょう。–tkoba

- 2006-11-02 (Thu) 21:00:49 takki：要望：xfig や Tgif の導入について。いつも、Mac OS X Workshop を使わせていただいています。便利に使わせていただいています。さて、これは要望なのですが、Workshop にはドロー系のソフトウェアが入っていないようで、現在は、fink や easypackage を併用して Xfig や Tgif を使用しています。もし、Workshop の apt-get でドロー系のソフトウェアがインストールできると、path の設定などが楽になり、より使いやすいものになるのではないかと思っています。おそらく、メンバーの方は Tgif などを使用していないためパッケージに含まれていないのだと思いますので、無理のない範囲でご検討いただけると幸いです。

  - コメントを有り難うございます。私も Linux の頃は tgif のお世話になりましたが、MacOS X 上では、OmniGraffle, Illustrator 等の優れたソフトがあるので tgif の必要性をあまり感じていません。ですが、どの程度の需要があるのか、一度アンケートを取るのもいいかもしれませんね。–tkoba
  - Inkscape はいかがですか？ たまに使ってます。–fu7mu4
  - お知らせくださり有り難うございます。早速試してみましたところ、一寸した作業であれば十分応えてくれそうですね。それに X11 のソフトなので OSXWS で用意すれば多分１／３くらいのサイズで用意できると思います。私は tgif よりは Inkscape の方がいいとは思いますが、、OmniGraffle がある事を考慮すると OSXWS としてメンテする価値があるかどうか、少々躊躇います。（せめて Aqua(Cocoa) なソフトであれば直ぐに採用するのですけれどね。。）–tkoba

- 2006-11-02 (Thu) 21:29:08 fu7mu4：.emacs.my.el のような、.vimmyrc の導入はないのですか？

- いつも有り難うございます。さて、.vimrc の個人設定ファイルですが、今一必要となる状況を想像出来ません。と云うのは、vi を常用する人であれば、.vimrc の管理くらいはできるものである、と仮定しても差し支えないのではないか、と思うからです。.vimmyrc があると嬉しい状況をご教示くだされば幸いです。(それと、はやく .emacs* の整理をしないと行けませんね。。) –tkoba
- そういえばそうですね。深い考えはなく、.emacs.my.el があるから、.vimmyrc があってもいいのじゃないかと思っただけです。–fu7mu4

- 2006-10-26 (Thu) 16:06:47 kaji : emacs の mew を使っています．メッセージ上で URL が書かれているところをクリックして firefox を起動させたいのです．.mew.el で mozilla となっている部分を firefox などに置き換えてもうまく起動しません．どなたかうまくいっていらっしゃる方がいれば，助言をお願い致します．(読みやすくするため質問文を fu7mu4 がまとめ直しました。)
  - /.mew.el に URL をマウス中クリックでブラウザ起動するようにするという項目がありませんか？ 当方の環境ではブラウザに Mozilla を使用する項目はコメントアウトしたままですが、何事もなく、デフォルトに指定している Firefox が起動します。–fu7mu4
  - kaji さん、fu7mu4 さんコメント有り難うございます。fu7mu4 さんが仰言る通り .mew.el で設定する必要はなく、Safari の「環境設定...」→「一般」→「デフォルト Web ブラウザ」でお好きなブラウザを指定すれば、Mew のメッセージにある URL を中ボタンでクリックすると指定したブラウザが起動します。–tkoba
  - fu7mu4 さん, tkoba さん, 助言ありがとうございます．.mew.el の該当部分をデフォルトの状態に戻し，その後おっしゃる通りにしてみましたが，私の環境 (Mac Pro, OSX 10.4.8, OSXWS/10.4) ではなぜかダメです．中ボタンをクリックすると emacs のウインドウはアクティブでなくなるのですが，それで何も起こりません．どこらあたりを疑えばよいでしょうか？ –kaji
  - うまくいかなった理由がわかりました．.mew.el でも emacs の問題でもなく，マウスの設定の問題でした．実はマウスに Microsoft IntelliMouse Explorer を使っているのですが，その設定で中ボタンの機能の割当が間違っていました．正しく直したところ，無事に firefox が起動するようになりました．お恥ずかしい．ご助言ありがとうございました．–kaji

- 2006-10-05 (Thu) 06:20:54 fu7mu4 : rpm -qf /usr/local/bin すると、Mxdvi-fonts と emacs が出ますが、どうしてこうなっているのでしょうか？
  - いつも有り難うございます。Mxdvi-fonts では files section で _bindir をそのまま指定していて、emacs では %dir として明示してあるから、と云うのが直接的な答えです。emacs の方は別に問題はありませんが Mxdvi-fonts はキチンと%dir 指定するかファイルを記述するか、に改めるておいたほうがいいですね。現状で不具合が発生する事はまずありませんが、対処しておきます。–tkoba
  - 10/7 日に apt で upgrade すると rpm -qf /usr/local/bin は emacs のみになりました。ありがとうございました。–fu7mu4

- 2006-10-03 (Tue) 15:08:47 rokko : OSXWS/10.4 の sylpheed を使おうとしています。既にあった日本語のファイル名のディレクトリをメールボックスとして読み込ませるとディレクトリ名 (utf-8) は表示されるのですが、クリックしてそこに行こうとすると chdir: Invalid argument とのメッセージが出てきます。また sylpheed のメニューから日本語名のメールボックスを kinput を使ってつくろ

うとしたときも mkdir: Invalid argument とのメッセージがでてきます。おそらくコードの問題だと思いますが。。

  - ご報告有り難うございます。10.4.8 リリースに合わせて rpm, apt, gtk2 などの更新作業をしているのですが、apt-get remove が動かなかったり（現状でもこの問題あり）gtk2 アプリが bus error を吐いたり、と難渋しています。お訪ねの件は、gtk2 と sylpheed のどちらの問題かを切り分ける必要がありますので、少々時間を頂戴します。–tkoba

- 2006-09-04 (Mon) 11:15:32 kaji : emacs の color-mode の件は apt-get upgrade で解決しました！ 素早い対応ありがとうございました．おかげさまで本当に楽に Linux から MacOS X に移行できて感謝しております．

  - こちらこそ、ご報告ありがとうございます。不具合の報告や、改善の提案はとてもありがたいものです。今後とも宜しくお願い致します。–tkoba

- 2006-08-28 (Mon) 20:02:10 kaji : Mac Pro(OS 10.4.7) 上で OSXWS/10.4 の emacs を使っていますが，color-theme を使用時に C-x 5 2 でフレームを開くと Bus Error で落ちます．color-theme を読み込まなければよいようです．また最新の carbon emacs package でも大丈夫なようです．可能ならパッケージを新しくしていただければ幸いです．

  - ご報告有り難うございます。今週末までに銭谷さんのビルドにキャッチアップ出来るように頑張ってみます。–tkoba
  - 銭谷さんの 20060901 版を基にして emacs とその周辺を更新しました。私の手元には現在 Intel Mac が無いので動作確認をお願いいたします。また、今回から mac-print-mode を使えるようになりました。M-x mac-print-buffer でお使いください。–tkoba
  - 私も Intel iMac で問題を確認していましたが、先ほど更新したところ、C-x 5 2 で落ちる問題は解決しました。更新の際に、

    これらを解決するためには `'apt-get --fix-broken install'` を実行する必要があるかもしれません。
    以下のパッケージは解決できない依存関係を持っています:
    （省略）

    といったメッセージがでましたが、指示に従うことで依存関係の問題を解決して更新できたようです。–seto

- 2006-08-05 (Sat) 08:32:57 fu7mu4 : ~/.mew.el にて mew-print-command-format で mpage コマンドが指定されています。a2ps に置き換えたほうがいいのでは？

  - fu7mu4 さんご指摘ありがとうございます。そして、一月ほどもご指摘を見逃しており、大変失礼いたしました。さて、対処法ですが、mac-print-mode の出来がよいので、内部で mac-print-buffer を呼ぶように書き換えました。詳細は「おしらせ→パッケージについて→その他」をごらん下さい –tkoba
  - mac-print-buffer ですか。わかりました。–fu7mu4

- 2006-07-20 (Thu) 14:57:42 ぜ : 作者の方の発言が→にあります。http://www.mew.org/pipermail/mew-dist/2005-July/026192.html ただ、配布物に README ／ライセンス条項が同梱されていないところが気になります。

- 何時も有り難うございます。ご教示下さりとても助かりました。私も銭谷さんが仰言る通り「配布物に README ／ライセンス条項が同梱されていないところ」が気がかりですが、GPL として packaging しようとおもいます。ただ、暫く忙しいので来週迄お時間をください。–tkoba
- いえ、「一言 ”Free” と書いてある外部コードを用いている」そうなので、GPL ではダメです。– ぜ
- おいかけてみました。一言”Free”と書かれている外部コードは http://www.isd.net/dsl03002/CocoaProgramming で配布されており、確かに Free と書いてあります。Free とあるのでダウンロードしてみたところ、利用されているという、GSNSDataExtensions.m のコメント部分に以下の記載があります。

```
 Created by khammond on Mon Oct 29 2001.
Copyright (c) 2001, 2005 Kyle Hammond. All rights reserved.

Original development comments by Dave Winer
```

それで、Original(C source code for base 64 encoder/decoder) 部分にはこうあります。

```
 Anyway, so I wrote an encoder/decoder. Docs are being maintained
on the web, and updates at:

http://www.scripting.com/midas/base64/

If you port this code to another platform please put the result up
on a website, and send me a pointer. Also send email if you think this
isn't a compatible implementation of Base 64 encoding.

BTW, I made it easy to port -- layering out the handle access routines.
Of course there's a small performance penalty for this, and if you don't
like it, change it. Thanks!
```

オリジナル (C) は自由に改変してよいということはわかりましたが、GSNSDataExtensions.m は利用が自由なのか改変が自由なのかは書かれていないことがわかりました。GPL ではありませんね。作者の khammond さん聞くしかないようです。fu7mu4

- 銭谷さん fu7mu4 さん、お調べ下さり有り難うございます。このところ昼休みに一寸時間が取れるだけなので大変助かります。スクラッチから自分で書いたコードであればライセンスはそれ程込み入ったりはしませんが、複数の開発者が階層を連ねていたりすると手間取りますね。兎に角私が状況を正しく把握する迄 packaging は一旦保留します。–tkoba

- 2006-07-19 (Wed) 14:27:51 fu7mu4 : mew-5.1 がリリースされましたが、http://www.mew.org/feature/spotlight.html.j にある、MewImporter.mdimporter は WorkShop に含めないのでしょうか？ Mew 用の mdimporter です。
  - ご指摘有り難うございます。私も package にしたいのですが license が不明で躊躇しています。どうしましょうか？ –tkoba
- 2006-06-28 (Wed) 22:59:14 uchida : a2ps の件自己解決しました。a2ps hoge.txt ¿ hoge.ps とリダイレクションで使うと一行目に [hoge.txt (プレーン): 1 ページ, 1 シート] が含まれてしまうのが原因でした。a2ps hoge.txt -o hoge.ps では上手くいきました。

- 2006-06-22 (Thu) 19:13:12 uchida : a2ps で変換したテキストファイルが ghostview で開くことができません。a2pdf の途中経過で作成される ps ファイルはエラーもなく開けます。

```
 Error: /undefined in test.txtAFPL Ghostscript 8.53: Unrecoverable error, exit code 1
Operand stack:
   --nostringval--
Execution stack:
   %interp_exit   .runexec2   --nostringval--   --nostringval--   --nostringval--   2   %stoppe
Dictionary stack:
   --dict:1124/1686(ro)(G)--   --dict:0/20(G)--   --dict:70/20
```

- こちらでは再現しませんでした。変換に失敗したテキストファイルを bz2 圧縮して私宛にメールで送ってください。–tkoba
- 2006-05-26 (Fri) 22:48:12 hsatoshi? : 了解しました。ありがとうございます。
- 2006-05-26 (Fri) 10:27:48 tkoba : fu7mu4 さんの認識で大丈夫です。私自身は最新の Xcode で開発しています。
- 2006-05-26 (Fri) 08:50:24 fu7mu4 : Xcode2.2 がリリースされた際に質問したところ、package の開発をしなければ影響ないとの返事がもらえました。
- 2006-05-24 (Wed) 22:13:50 hsatoshi? : Xcode2.3 がリリースされましたが、インストールしても OSXWS には影響はありませんか？ 一般に、Xcode がバージョンアップされればインストールしても問題ないのでしょうか？
- 2006-05-24 (Wed) 08:08:18 matsuda? : 数式の背景がグレーになる件，わかりました．お騒がせして申し訳ありません．.latex2html-init に$LATEX_COLOR=””; を追加しました.
- 2006-05-23 (Tue) 21:04:43 matsuda? : permission なんですね．ありがとうございました．とにかく動いて感動しました．で，latex2html 初心者ですが，`\{equation}{数式}` がグレーで網かけされてしまいます．image1.png も網かけされているのですが…．このレベルの質問に適当な相談場所を教えて下されば幸いです.
- 2006-05-23 (Tue) 19:25:25 matsuda? : 申し訳ありません．改行のつもりで, つい…． latex2htlm が動きません．error message は [japanese]Error: ’/private/var/local/tmp’ not usable as temporary directory. です．教えて頂ければ幸いです.
  - ご報告有り難うございます。設定が悪いですね。私の環境で /private/var/local/tmp の permission を調べてみたところ、755 でオーナーが OSXWS をインストールしたユーザーになっていました。近いうちに然るべきパッケージで管理するようにしますが、当面以下の作業でしのいで下さい。–tkoba

    ```
    $ sudo chmod 777 /private/var/local/tmp
    ```

- 2006-05-23 (Tue) 19:20:35 matsuda? : latex2htlm が動きません.
- 2006-05-18 (Thu) 21:35:39 tkoba : fu7mu4 さん銭谷さん、ご報告有り難うございます。もう pkfont 自体を使う事が殆ど無い筈なので tetex.cron は外せそうですね。となると、/private/etc 以下も整理できそうです。ToDo? に入れておきます。

- 2006-05-17 (Wed) 15:39:53 ぜ : 私の環境（10.4.6 PPC）でも/usr/sbin/tmpwatch は存在しません。
- 2006-05-17 (Wed) 06:23:13 fu7mu4 : tetex-3.0-10.4osx4 に含まれる、/etc/cron.daily/tetex.cron から /usr/sbin/tmpwatch を呼び出していますが、私の環境では存在しません。そのため、/var/log/daily.out にエラーが記載されます。MacOSX/10.4.6 には標準で存在するのでしょうか？
- 2006-05-10 (Wed) 14:14:13 seto : yahtml-mode の件、遅くなりましたが yatex の ML に聞いてみたら、早速回答をいただきましたので報告します。新しい Emacs 22 では、ファイルの内容のパターンをみてモードを起動する方法が導入されたようで、magic-mode-alist の html-mode を yahtml-mode に置換すれば良いようです。

```
;; Emacs-22 provides magic-mode-alist...
(if (boundp 'magic-mode-alist)
    (or (rassq 'yahtml-mode magic-mode-alist)
(setq magic-mode-alist
      (cons
       (cons (car (rassq 'html-mode magic-mode-alist)) 'yahtml-mode)
       magic-mode-alist))))
```

- 2006-04-25 (Tue) 10:47:02 tkoba : ご報告有り難うございます。ghostscript の PreReq? 指定のミスのようです。出来るだけ早く直しておきます。（連休までずれ込みそうです…）
- 2006-04-19 (Wed) 02:13:25 uchi? : 不具合報告です。OSXWS の Ghostscript が PostScript を gv で開いたり、ps2pdf を使うと

```
AFPL Ghostscript 8.53: Unrecoverable error, exit code 1
```

というエラーを吐きます。Ghostscript の cidfmap に指定されているヒラギノが

```
/usr/local/share/ghostscript/Resource/CIDFont
```

にないためのようなので、とりあえずこのディレクトリにヒラギノフォントのシンボリックリンクを作成すればうまくいきます。

- 2006-04-16 (Sun) 18:36:17 seto : 確認有り難うございます。私も同じ状況です。YaTeX の ML で聞こうと思います。
- 2006-04-16 (Sun) 16:56:55 ぜ : 既存のファイルを開くと html-mode、新しいファイルを開くと yahtml-mode になりました。yatex か macemacs の ML で問い合わせてみてはどうでしょうか？
- 2006-04-16 (Sun) 16:20:39 seto : `("\\.s?html?\\(\\.[a-zA-Z_]+\\)?\\'" . yahtml-mode)` を auto-mode-alist に cons しています。でも、銭谷さんのところでは再現しませんか？
- 2006-04-16 (Sun) 12:50:31 ぜ : *.html ファイルに yahtml-mode をアサインしてないからではないでしょうか？ auto-mode-alist です。
- 2006-04-16 (Sun) 07:40:44 seto : ちょっと前にも報告を頂いていましたが、現在も html ファイルを開くとき、yahtml-mode ではなく html-mode が立ち上がるようです。CarbonEmacsPackage でも同じ問題があり、/usr/bin/emacs ではちゃんと yahtml-mode になるので、Carbon Emacs が原因のようです。

- 2006-03-16 (Thu) 19:16:06 kryo? : 今月の 13 日に update と dist-upgrade を実行し、次いで osxws-upgrade を実行したところ次のメッセージが出力されました。

```
 cp: /Library/Application Support/OSXWS/en/.Xclients: No such file or directory
cp: /Library/Application Support/OSXWS/en/.Xresources: No such file or directory
cp: /Library/Application Support/OSXWS/en/.bash_logout: No such file or directory
cp: /Library/Application Support/OSXWS/en/.bash_profile: No such file or directory
cp: /Library/Application Support/OSXWS/en/.bashrc: No such file or directory
cp: /Library/Application Support/OSXWS/en/.cshrc: No such file or directory
cp: /Library/Application Support/OSXWS/en/.cshmyrc: No such file or directory
cp: /Library/Application Support/OSXWS/en/.rpmmacros: No such file or directory
cp: /Library/Application Support/OSXWS/en/.vimrc: No such file or directory
install: /Library/Application Support/OSXWS/en/.yatex.el: No such file or directory
install: /Library/Application Support/OSXWS/en/.inputrc: No such file or directory
install: /Library/Application Support/OSXWS/en/.xinitrc: No such file or directory
```

  おそらく tkoba さんが en ディレクトリを用意されていないことと、私が「システム環境設定」の言語環境で English を最優先にしているためだと思われます。今のところ問題はありませんが、一応報告しておきます。

  - kryo さんご報告ありがとうございます。お察しの通り jp 以外の環境は準備中です。環境の整備は実際に使っている方の設定を参考にしたく思っています。/Library/Application Support/OSXWS/jp 以下のファイルを参考にして英語環境に適した設定に調整して戴けると助かるのですが、、kryo さん、如何でしょう？ –tkoba
  - えーと、私の理解では「該当するファイルを/Library/Application Support/OSXWS/jp から手でホームディレクトリにコピーして、不都合があれば英語環境に合わせて手を加える」と取ったのですが、それでよろしいでしょうか？ それから、私が英語環境で作業しているのは、/Applications や Dock がなんとなくごちゃっとした印象があり嫌だったからです。私個人の見解ですが、日本語環境と英語環境での OSXWS の設定はそれほど変えなくても良いと思います。参考になれば幸いです。–kryo?

- 2006-03-07 (Tue) 07:55:55 seto : 現在の新しい仕様では、共通の設定ファイルという位置づけで /usr/local/share/emacs/site-lisp/emacs-lisps/dot.emacs.el を読み込み、そこで `(setq user-init-file "~/.emacs.m` と指定されているので、ユーザーごとのローカルな設定を読み込むようになりました。私は、そもそも設定ファイルの類いはあくまで最初のサンプルを提供して、その後はユーザーに任せるという立場で良いと思っています。しかし、CarbonEmacs の場合、本体そのものや、インラインパッチなどといった開発が急ピッチに続けられているものが含まれており、一部ユーザーの設定ファイルを変更する必要性が生じることがあるので、その部分を OSXWS の管理下にしてしまおうということだと理解しています。(または、どのような設定ファイルを提供するべきかということが定まっていなかったので、とりあえずパッケージ扱いで提供しておいて、あとからよりよいものに更新しようという OSXWS 側の事情だった。) いずれにせよ、こういった管理は、最終的には打ち切るべき物だと思います。また、OSXWS で提供したサンプルが、CarbonEmacs パッケージや OS 標準の Emacs を想定して書かれた物でなかったとしても、それはしょうがないように思います。(サンプルで提供する設定ファイルを吟味して、OSXWS に依存する設定を Emacs のバージョンなどをみて選択的に読み込むようにする、と

いった方法で対処するのがいいです) 今回/usr/local/share/emacs/site-lisp/emacs-lisps/dot.emacs.el に移動した内容は、マルチユーザ環境で、全員が共通に使う設定としては、少しふさわしくないように思います。もしかすると、このレベルで必要な設定は一つもないように思うのですがどうでしょうか？ また、しばらくこの仕様でいくとしても、ここで (expand-file-name "~/.emacs.my.el") をすると、オプション-q をしても、設定ファイルが読み込まれるようなので消した方が良いと思います。(この意見がでたときに、すぐに反応できず、後から意見をいう形になり申し訳ないです)

- OSXWS ユーザ以外にも分かるように、MacOSX_WorkShop/dot_emacs_el に内容をあげました。この議論が終わるとすぎに消します。
- ~/.emacs.el の件は、私が CarbonEmacs の Intel 版をお手伝いする際に悪影響を与えない様な環境を作る事が目的の一つでした。~/.emacs.el をキチンと整備するのが本来である事は認めます。ですが、もともと ~/.emacs.el の改変自体を原則認めていない OSXWS の現状では、~/.emacs.el そのものを site-start.d 内に置く方が、現実的には混乱は少なくなる様に感じています。ここでの設定を変更したければ、~/.emacs.my.el 内で over write すれば良い訳ですから。seto さんが「マルチユーザ環境で、全員が共通に使う設定としては、少しふさわしくない」と思う気持ちは解らないでもないです。ですが、少なくともこれまでと実質的には状況は変わりません。むしろ、emacs の標準的な環境をどの様に整備するかを議論した方が建設的でしょうね。-q オプションの件は考えないと行けませんね。–tkoba
  - しばらくは現行の共通／個別の切り分けで行くということは了解しました。実際に使用される状況で、可能な限りトラブルを少なくするためには必要なことだと思います。将来のバージョン (10.5 以降) では、ホームディレクトリ外の設定を最小に切り詰めて、ローカルな.emacs.el としてサンプルを提供するという方針になれば良いと思っています。また、今回共通部分がユーザ権限で書き換えられないところに移動したことで (後で設定の上塗りはできるとはいえ)、Emacs の設定ファイルに関して再検討するチャンスと思いました。(小林さんも多忙な時期ですので) 作業を急ぐ必要はないと思いますが、他の方の意見やアイデアも頂いて再編成できないかと思っています。–seto
  - 同感です。MacOSX_WorkShop/dot_emacs_el も出来た事ですし、この機会に OSXWS の emacs 環境を総合的に改善していきましょう。–tkoba
- fu7mu4 さんも下で仰言っている「CarbonEmacs がよくできているのでシステム標準を使うメリットがよくわかりません。」は、私も同感です。実際に OSXWS を使って日々の仕事をこなすのであれば、現状で顕在化している問題は無い訳ですからね。ただ、ユーザーの利用形態は実に様々ですから、一寸した手直しで、システム標準の emacs や CarbonEmacs Package を併用する必要があるユーザーにも対応できるのであれば、プロジェクトとしては対応するに越した事はないでしょう。今回はこのケースです。ですので、今回の変更後でも、殆ど全てのユーザーさんに取って、実質的な変更はない筈ですし、煩う必要もありません。–tkoba
- 僕はリモートから ssh でログインしたときに、システム標準の emacs を使うことがあります。(エイリアスを設定すれば CarbonEmacs を使うこともできるんですけどね) – ぜ
- そうですね OSXWS では、/usr/local/bin/openemacs を叩く様にエイリアスが組まれています。ssh で入った時もそのまま emacs コマンドを叩けば自動で-nw オプションが付いて起動されるように調整されています。–tkoba

- 2006-03-06 (Mon) 23:19:10 ぜ : 詳細はよくわからないですが、初期ファイル名は .emacs.el のままの方が望ましいと思います。外部ライブラリの有無などは、elisp で判断させれば良いでしょう。

– 確かに初期設定ファイルが~/.emacs.el のほうがユーザーにはわかりやすいのですが、ディストリビューター側のチェックが省けるというメンテナンス上の利点を述べられると、そうなのかなと思ってしまいます。別の質問なのですが、WorkShop でシステム標準の Emacs を使う方法があるのですか？ CarbonEmacs がよくできているのでシステム標準を使うメリットがよくわかりません。 – fu7mu4

- 2006-03-01 (Wed) 13:41:03 ぜ： .emacs.el の内容はできるだけ、site-init.el/site-start.el に移されてはいかがでしょうか？　　CarbonEmacsPackage やシステム標準の emacs を使おうとしたとき、OSXWS の .emacs.el ではエラーを起こしてしまいませんか？

  – そうですね。各ユーザーのドットファイルの整理はずっと考えている事の一つです。銭谷さんの３月パッケージに合わせて整理しようと思います。–tkoba

  – ~/.emacs.el の内容を site-start.d 内に移動しました。–tkoba

  – 99dot.emacs.el 内で~/.emacs.my.el を読む設定なのですね。~/.emacs.el が~/dot.emacs.el.osxws に移動したわけですね。ところで site-start.d 内の設定ファイルにて user-init-file を~/.emacs.el に指定してますが、~/.emacs.el が存在しないのは良くないのではないでしょうか。例えば Emacs の custmize を利用した場合。このときにファイルが存在しなかったらどうなるのでしょうか。ここは~/.emacs.my.el を廃止して~/.emacs.el にコピーした方がいいのではないでしょか。私の環境で異常動作が出た訳ではありませんけれども。–fu7mu4

  – fu7mu4 さん、ご指摘有り難うございます。仰言る通り user-init-file が ~/.emacs.el であるのはまずいですね。—verb— /.emacs.my.el— に直しました。~/.emacs.el を置くと、銭谷さんご指摘の通り、CarbonEmacsPackage を利用する際（まぁ、現実的には OSXWS のものを利用するので問題はありませんが、避けられる不都合は予め避けておくべきだと考えます。）に支障が出る可能性があるのでコピーはしないで下さい。–tkoba

- 2006-02-24 (Fri) 10:21:15 demura?：なるほど、了解しました。ありがとうございました。分かってすっきりしました。

  – 納得して戴けて嬉しいです。今後も何か気付かれましたらご遠慮なくご報告ください。–tkoba

- 2006-02-23 (Thu) 11:17:55 demura?：いつもお世話になります。updmap-otf の挙動が私の環境では、どうも想定と違うような気がします。OTF-Hiragino をインストールした状態で、updmap-otf auto としても、hiragino の設定になりません。試した手順は、sudo updmap-otf nofont ¿ sudo updmap-otf auto です。sudo updmap-otf hiragino で設定した後、pdf をつくったものは、きちんと hiragino が埋め込まれています。とくに現状で困っているという状況ではないのですが、ご報告まで。

  – 上記の手順では noEmbeddedFont が設定されます。これは正しい動作です。auto で hiragino が設定されるのはインストール時だけです。もしも上記の手順で hiragino に設定が変更されてしまうと、noEmbeddedFont を設定している環境で texmacro-otf package が更新されると hiragino に設定が勝手に変わってしまいます。noEmbeddedFont から hiragino に設定を変更したい場合は正しく hiragino を以下のように指定してください。 –tkoba

    ```
    $ sudo updmap-otf hiragino
    ```

- 2006-02-16 (Thu) 17:49:11 えんどう？：Q.apt-cache はどこにインストールされるのでしょうか?

– インストーラを入れただけでは OSXWS は不完全です。http://www.bach-phys.ritsumei.ac.jp/OSXWS/node15.h
を参照してください。–tkoba

- 2006-02-13 (Mon) 04:55:24 fu7mu4 : CarbonEmacs での shell-mode でエスケープシーケンスを処理する方法が `http://www.namazu.org/~tsuchiya/elisp/#ansi-color` に掲載されています。alias 以外の解決方法としていかがでしょうか

  – fu7mu4 さん、何時も有り難うございます。早速試してみましところ良好な結果が得られましたので OSX-Preferences を更新しました。–tkoba

- 2006-02-06 (Mon) 19:05:45 kryo? : 二月六日の昼の時点で試したところエラーなく更新が行なわれました。`/Library/Application\ Support/OSXWS/jp` から.emacs.el と.emacs.my.el をコピーして home のものに上書きをしました。(.emacs.my.el の古いものは別の名前でバックアップを取りました。) その後で.emacs.my.el を前と同じ環境で作業するために書き替えました。このとき、`/Library/Application\ Support/OSXWS/jp` から持ってきたそのままの.emacs.my.el に対して 83 行目に次のものを加えました。

```
(setq default-frame-alist
    (append
      '(
        ;;character color
        (foreground-color . "#eee9e9") ;snow1
        ;;background color
        (background-color . "#1a1a1a") ;gray10
        ;;cursor color
        (cursor-color . "#a1a1a1") ;gray63
        )
default-frame-alist))
```

これはフレームの色を変えるもので、文字を白に背景を黒にします。カーソルは灰色にします。ただし、この時 (color-theme-ld-dark) はコメントアウトしました。このとき何故かカーソルの色だけが変わりません。`(background-color . "#1a1a1a")` を `(background-color . "#ff0000")` とすればきちんと赤くなるのですが。何故か `(cursor-color . "#a1a1a1")` だけが受け付けられません。emacs ではエラーメッセージは表示されませんでした。ただし、`"C-x 52"`を実行してもう一つウィンドウを表示すればカーソルもきちんと灰色になります。OSXWS が 10.4-1 の頃は上のスクリプトで何の問題もありませんでした。この問題は 10.4-2 になってから生じたものです。お手間をおかけして申し訳ありませんが、何か心当たりありますでしょうか？

  – あら、現在の inline_patch では修正された、と Macemacsjp-users ML で報告されていたのですが。。一寸調べてみます。–tkoba
  – お手間をおかけします。どうやらローマ字入力では黒いカーソルを、日本語入力では赤いカーソルを表示するようです。昔はそんな設定ではありませんでしたよね？ –kryo?
  – そうですね。inline_patch の仕様がその様に変わっています。以下を参考になさってください。–tkoba
    * http://lists.sourceforge.jp/mailman/archives/macemacsjp-users/200.../000919.html

– 今ちょっと遊んでみたところ、(mac-set-input-method-parameter 'japanese 'cursor-color "Color-Name"
で日本語入力のときのカーソルの色を、(mac-set-input-method-parameter '0 'cursor-color "Color-Name"
で英字入力のときのカーソルの色を設定出来るようですね。便利な設定だと思います。これで
やっと昔と同じ環境を築けました。どうもありがとうございました！ –kryo?

13.3 10.4-2 公開迄

- 2006-02-05 (Sun) 21:05:49 demura? : 昨夜は遅くまでおつきあいいただきありがとうございました。今、試しましたところ、エラーもなく、すべての更新がうまくいったようです。これからもよろしくお願いいたします。いつも仕事を進める上で、大変重宝しています。ありがとうございます。ご報告まで。
  – ご報告有り難うございます。皆さんのお声はこの OSXWS をより良いものにする原動力の一つです。今後ともどうぞ宜しくお願い致します。–tkoba
- 2006-02-04 (Sat) 23:02:33 demura? : お手間を取らせます。rpm -q apt では「apt-0.5.15.cnc.7-10.4osx2.1」でした。
  – demura さん、深夜迄おつきあい戴き有り難うございます。パッケージの依存関係がらみである事が判明しました。現在では普段通りの更新作業でそのまま更新可能な状態になったと思います。ご確認戴けますでしょうか？ –tkoba
- 2006-02-04 (Sat) 22:46:46 demura? : sudo apt-get apt を行ったところ、「* apt は既に最新バージョンがインストールされています。」というメッセージでした。sudo apt-get dist-upgrade では、アップグレード他、すべての項目で 0 個というコメントでした。いかがでしょうか？
  – 早速お試しくださり有り難うございます。rpm -q apt で apt-0.5.15.cnc.7-10.4osx7 と出ますでしょうか？ –tkoba
- 2006-02-04 (Sat) 22:20:55 金田? : CarbonEmacs において、M-x shell として Emacs から bash を使おうとすると文字化けしてしまいます。自分でも色々と試してみたのですが、うまくいきません。どなたかご教授頂ければ幸いです。
  – 金田さん、ご報告有り難うございます。こちらでも確認しました。対策にあたります。–tkoba
  – これは、素の状態の CarbonEmacs Package 12 月版でも起こりますね。銭谷さん心当たりあります？ –tkoba
  – このプロジェクトにはとても期待しておりますし、できることがあれば貢献していきたいと思っております。よろしくお願いします。金田?
  – ご支持を戴き有り難うございます。日本語ファイルの表示はまだうまく行きませんが、ls コマンドは判りました。~/.bashmyrc に alias ls='\ls' を加えてみてください。–tkoba
  – やってみたらうまくいきました。どうもありがとうございました。個人的には日本語ファイルは使わないのでこれで問題ないのですが、どうにかできないものかといろいろといじってみても私の能力では解決しそうにありません。とりあえず日本語を扱うときは eshell で済ませようと思います。–金田?

- 2006-02-04 (Sat) 21:58:44 demura? : sudo apt-get install –reinstall apt では、パッケージリストを読みこんでいます... 完了；依存情報ツリーを作成しています... 完了；* apt をダウンロードすることができないため、再インストールは不可能です。；アップグレード: 0 個, 新規インストール: 0 個, 削除: 0 個, 保留: 0 個となりました。その後の update で出てくるコメントは、一回目の update でのコメントと一緒です。
  - demura さん、詳細なご報告有り難うございます。そのまま sudo apt-get install apt か sudo apt-get dist-upgrade するとどうなりますでしょうか？ お手数をおかけしますが宜しくお願い致します。–tkoba
- 2006-02-04 (Sat) 21:56:57 demura? : お手間を取らせます。まず、sudo apt-get update で現れるログは、取得:1 http://www.bach-phys.ritsumei.ac.jp Tiger/ppc release [1087B]；1087B を 15s 秒で取得しました (69B/s)；取得:1 http://www.bach-phys.ritsumei.ac.jp Tiger/ppc/main pkglist [22.9kB]；取得:2 http://www.bach-phys.ritsumei.ac.jp Tiger/ppc/main release [157B]；取得:3 http://www.bach-phys.ritsumei.ac.jp Tiger/ppc/main srclist [39.0kB]；62.0kB を 2s 秒で取得しました (28.1kB/s)；パッケージリストを読みこんでいます... 完了；依存情報ツリーを作成しています... 完了 以上です。次に続きます。
- 2006-02-04 (Sat) 21:18:12 demura? : 試してみました。install –reinstall apt で「apt をダウンロードすることができないため、再インストールは不可能です。」とでました。2 回目の update では、Tiger/ppc/main とでました。どうも、うまくいっていない様子です。今回は、自宅から試したので、週明けに、職場から試してみます。取り急ぎご報告まで。
  - エラーメッセージが不可解ですね。。log を見てみると 4 日 18 時時点の tree で正常に apt の更新が出来ている人が複数人いるのですが。。。キャッシュが残って悪さをしているのかもしれません。一度 apt が返すエラーをそのままここにコピーして戴けますでしょうか？ –tkoba
- 2006-02-03 (Fri) 16:24:00 demura? : お手数をかけ、すみません。次のお知らせをお待ちしています。
  - こちらこそテスト不足で申し訳ありません。apt を作り直してみましたのでお試し戴けますでしょうか。２回目の apt-get update で http://www.bach-phys.ritsumei.ac.jp Tiger/fat/main の行が表示されれば成功です。

    ```
    $ sudo apt-get update
    $ sudo apt-get install --reinstall apt
    $ sudo apt-get update
    ```
- 2006-02-03 (Fri) 16:07:47 demura? : お手数をおかけします。教えていただいた方法の結果は、「apt をダウンロードすることができないため、再インストールは不可能です。」というものでした。わたしの職場では、ダウンロードの際にチェックをかけるため、時々、ダウンロードがうまくいかないことがあります。その場合、再度ダウンロードをかけると、すでにチェックの終わったものを吸い出せるので、次にはうまくいきます。今回のもその成果と思い、何回か install-apr –reinstall apt してみましたが、結果を同じでした。取り急ぎ、ご報告いたします。
  - 早速のご報告有り難うございます。状況が良く判りました。別の方法を試してみます。用意が整いましたらご連絡致します。–tkoba

- 2006-02-03 (Fri) 15:15:23 demura? : 早速ありがとうございます。install-apt をかけたところ、今度は昨日と同じく、「既に最新バージョンがインストールされています。」というメッセージでした。その後、dist-upgrade したところ、readline readline-devel の二つが保留項目でした。以上の状態で OK でしょうか？

  - いえ、期待される動作とは違います。。server の log を観てみたのですが demura さんは正しく操作していらっしゃいますね。多分現状では apt の version は 10.4osx2.1 or 10.4osx5 だと思います。正しくは 10.4osx7 にならなければいけません。試しに以下の操作をして頂けますでしょうか？ お手数をおかけ致しますが宜しくお願い致します。

    ```
    $ sudo apt-get update
    $ sudo apt-get install --reinstall apt
    ```

- 2006-02-03 (Fri) 13:05:17 tkoba : demura さんご報告有り難うございます。apt tree を整備し直しましたので改めてテストして戴けますでしょうか。

- 2006-02-03 (Fri) 10:46:13 demura? : つづきです。依存:OSX-keyring それをインストールすることができません、librpm-4.4.dylib、librpmdb-4.4.dylib、librpmio-4.4.dylib/usr/bin/python それをインストールすることができません。以上です。

- 2006-02-03 (Fri) 10:44:00 demura? : 長くなりますが、install-apt をかけたときのエラーメッセージの一部です。以下のパッケージは解決できない依存関係を持っています

- 2006-02-03 (Fri) 10:41:45 demura? : いつもお世話になっています。私も昨日 install apt に失敗しました。今朝、ここを読みまして、再度上記手順を試みたのですが、install apt で、（昨日とは違う）エラーメッセージが現れます。その後、ひとまず install-apt はあきらめて、dist-upgrade をかけてみました。この際、apt は保留項目の一つとなっていました、その後再び、install apt をやってみたのですが、うまくいきません。このまま、ほっておいてよいものでしょうか？

- 2006-02-03 (Fri) 09:34:45 tkoba : kryo さん fu7mu4 さんご報告有り難うございます。apt package が ppc から fat に変更になったのが不具合の原因です。ご指摘を戴き漸く気づきました。上記の方法で更新できる様に apt-*ppc.rpm を整備し直しました。お二人を人柱にしてしまった事になり大変申し訳ありません。今後もこれまで通りご意見を戴ければ幸いです。

- 2006-02-03 (Fri) 06:03:23 kryo? : 分かり難くてごめんなさい。「`~/.emacs.my.el` を編集していた」というのは vi などでファイルをいじっている最中であったという意味ではなく、`~/.emacs.my.el` を自分で書き換えたものを使用していたということです。

- 2006-02-02 (Thu) 23:01:36 fu7mu4? : 私も emacs を実行中に apt-get をかけてしまい、おかしくなったことがありました。これはほかのこともあって、OS から再インストールしたのでいいのですが、apt-get update で package を delete するのではなく、apt-get upgrade 時に y を答えてから、package のアンインストールとインストールするようにできないものでしょうか？ VineLinux などではそのようになってますよね。

  - apt の動作は VineLinux? と全く同じですよ。apt-get update で package を delete することは絶対にありません。Del.... と出るのは apt tree に package の実態がなくなった事を示しているだけです。今回は ppc から fat へと見かけ上の arch が変更になったのが原因です。–tkoba

  - そうだったのですか、わかりました。よく人柱になってますので、あまり気にしてません。–fu7mu4

- 2006-02-02 (Thu) 15:18:52 kryo? : 二月二日の朝の時点で更新情報に示されているようにアップグレードを行ないました。その際、sudo apt-get install apt を実行したところ「apt は既に最新バージョンがインストールされています。」というメッセージが表示されました。そのまま続けて sudo apt-get update と sudo apt-get dist-upgrade を行なったところ、emacs がアップグレードされました。`~/.emacs.my.el` を編集していたため、/Library/Application Support/OSXWS/jp/を参照しようとしたところ、存在しませんでした。ただし、/System/Library/User Template/Japanese.lproj/は見つけることが出来ました。

- 2005-12-25 (Sun) 18:25:31 fu7mu4? : apt-get update をしたところ、以下の警告がでました。checksum の情報が足りないそうです。警告に従って update しましたが、改善されません。

```
 W: Relese file did not contain checksum information for
http://www.bach-phys.ritsumei.ac.jp/OSXWS/Tiger/ppc/base/pkglist.test

W: Relese file did not contain checksum information for
http://www.bach-phys.ritsumei.ac.jp/OSXWS/Tiger/ppc/base/release.test

W: Relese file did not contain checksum information for
http://www.bach-phys.ritsumei.ac.jp/OSXWS/Tiger/ppc/base/srclist.test

W: この問題を解決するには'apt-get update' を実行する必要があるかもしれません。
```

  - ご報告ありがとうございます。test tree が source list に含まれているのは好ましくありません。/etc/apt/source.list を /etc/apt/source.list.rpmnew で置き直してください。–tkoba

```
$ sudo mv -f /etc/apt/sources.list.rpmnew /etc/apt/sources.list
```

  - 確かに test tree が含まれていました。置き直して update で警告が出なくなりました。ありがとうございました。–fu7mu4

- 2005-12-18 (Sun) 00:58:45 ぜ : CarbonEmacs の elisp をパッケージ化される際は、以下の点にご留意いただければ幸いです。

  - mac-key-mode ... 2005111x 以降の Emacs CVS では、動作しないかもしれません
  - mac-drag-N-drop ... smart-dnd にリプレース予定です
  - `http://homepage.mac.com/zenitani/elisp-j.html#smart-dnd`
  - お知らせくださり有り難うございます。本業が忙しくてなかなか作業が進みませんが、年末年始に時間でも作ろうと思います。–tkoba

- 2005-12-02 (Fri) 13:04:57 demura? : 早速の対応をありがとうございました。なるほど確かに、ヒラギノを埋め込むときは otf パッケージを明示しておくというのは、埋め込むかどうかをソース上で制御できるので、良いアイデアですね。他の方の意向もあるでしょうが・・・。

– VineLinux 等のデフォルトではフォントは埋め込まない設定になっています。また、OTF package はヒラギノ等の OTF が持っているグリフをフルに活用する為に作られたもので、フォントを埋め込むかどうかは本来別の問題です。ですが、OSX では折角ヒラギノが付いているのでデフォルトでヒラギノが埋め込まれた PDF を作成できた方が便利であろう、との判断でその様な設定にしています。OTF package を使用してもフォントを埋め込まない設定にするのも簡単で以下の様にしてください。–tkoba

```
$ sudo updmap-otf  nofont
```

– 将来的には、これに相当する作業を管理者権限なしで行えるようになるといいですね。–seto

– 実は teTeX-3.0 にはその機能が付いていて、各ユーザー事に自分好みの設定を置く事ができます。ただ、問題になるのは package の更新時にそれら各ユーザ個別の設定まで面倒を看られない事にあります。別の解決法として土村さんと議論しているのですが、使用するフォントと埋め込みの有無を tex source に明示する方法も検討しています。それには OTF package そのものを弄らなければならないので齋藤さんにも作業をお願いする必要がありますね。–tkoba

- 2005-12-01 (Thu) 18:55:33 demura? : いつもお世話になっています。OSXWS/10.4 を利用させていただいています。otf がらみで挙動がおかしいようで、報告します。otf パッケージを読み込まない場合、明朝体でヒラギノが埋め込まれません。otf-hiragino auto で設定し直してみましたが、同様のようです。どうも、最近の更新以降のような気がします。

  – ご報告有り難うございます。texmf/fonts/map/dvipdfm/cid-x.map の内容を Vine の package と同一のモノにしたのが原因でした。私個人としては PDF にフォントを埋め込むときは `\usepackage{otf}` を明示した方が気持ちがいいのですが、OSX 上ではデフォルトでヒラギノが埋め込まれた方が便利ですね。修正しておきます。–tkoba

- .bashrc に記した cp, mv, rm の alias が効かないなど bash-completion の side effects が幾つか見つかりました。原因を探っていますが、問題が解決するまで OSX-base package で Requires するのをストップしました。もしも bash-completion 由来の不具合でお困りの場合は以下を実行して bash-completion を抜いてください。

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get remove bash-completion
```

  – 自分で設定したエイリアスが効かなくなるんですか？ 僕の手元では次の設定が効いているようですが... – ぜ

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

  – ご報告有り難うございます。MacOS X 10.4.3 と現在の 20050721 の組み合わせでは大丈夫である事を確認しました。復帰させておきます。–tkoba

- 2005-11-11 (Fri) 23:31:58 fu7mu4? : Apple より、開発環境 Xcode2.2 がリリースされましたが、OSXWS との互換性はどのようになるのでしょうか？

  - package の開発をしないのであれば特に互換性を気にする必要はありません。今後の package build は Xcode2.2 base になります。もしも不具合がでましたらご報告ください。対処致します。–tkoba

- 2005-09-01 (Thu) 04:04:38 ぜ：ClamXav のベースになっているアンチウイルスエンジン、Clam AV http://www.clamav.net/ はいかがでしょうか？ ウイルス対策はこれからますます重要になっていくと思います。

  - 何時もお知らせ戴き有り難うございます。仰言る通りウイルス対策は必要ですね。早速作業しようと思います。–tkoba

- 2005-07-31 (Sun) 08:36:54 tkoba：a2ps, a2pdf が ja_JP.UTF-8 に対応していない事が判りました。現在ハック中です。暫くは以下の様にして凌いでください。

```
$ LC_ALL=ja_JP.eucJP  a2pdf  hoge.txt
```

  - 非常に ad hoc ですが対処しました。PS file への書き出しはリダイレクトではなく -o オプションで指定してください。–tkoba

- 2005-07-26 (Tue) 11:04:41 demura?：小林さん、さらにお返事ありがとうございます。おはずかしい、update と upgrade を勘違いしていました。とほほ。お時間を取らせて、申し訳ありませんでした。これからもよろしくお願いします。

- 2005-07-25 (Mon) 11:49:49 demura?：小林さん、お返事ありがとうございました。今回の更新は、パッケージの削除を伴うものだったということなのですね。なるほど、よくわかりました。ありがとうございました！

  - update と upgrade を混同されているようです。upgrade, dist-upgrade のどちらを用いるにせよ、最初に update してデータベースを更新しておく必要があります。パッケージ更新の様子をご覧戴ければ判りますが今回はパッケージの削除はありませんから upgrade でも同じです。upgrade と dist-upgrade の違いが出るのは、例えば kterm が廃止されて mlterm へ移行する時に、upgrade では保留されて dist-upgrade では kterm が mlterm に置き換えられます。–tkoba

- 2005-07-25 (Mon) 09:32:51 tkoba：demura さん、ご報告有り難うございます。パッケージの更新については http://www.bach-phys.ritsumei.ac.jp/OSXWS/node14.html に纏めてありますのでご覧ください。簡単に説明しますと apt-get update で apt のデータベースを更新し、その後そのデータベースに基づいてパッケージを更新する時に upgrade or dist-upgrade を用います。

- 2005-07-24 (Sun) 22:26:30 demura?：小林さん、ありがとうございました。パッケージを更新して、otf 無しでも動くことを確認しました。ありがとうございました。ところで、今回の task-tetex の更新は、update でやったときは行われず、dist-upgrade で反映されました。パッケージの更新は、種類によっては、dist-upgrade を使用する必要があるのでしょうか？

- OTF パッケージを利用しない時に dvipdfmx が働く様に map files を調整してみました。パケージを更新してご確認ください。–tkoba

- 2005-07-22 (Fri) 22:56:49 demura?：小林さん、遅くに煩わせてすみません。助かります。いつもお世話になってばかりで恐縮です。

- 2005-07-22 (Fri) 22:46:49 tkoba : demura さん、OTF パッケージを使わずにヒラギノを埋め込む為の設定が抜け落ちていました。明日一番で早速修正しておきます。ご報告有り難うございました。
- 2005-07-22 (Fri) 22:30:10 demura? : 興奮のあまり、お礼を忘れていました。とりあえず、otf パッケージを使うことで、切り抜けたいと思います。小林さん、瀬戸さん、ありがとうございました。
- 2005-07-22 (Fri) 22:26:31 demura? : おおっと。seto さんのご助言に従って、`\usepackege{otf}` をいれたところ、ちゃんと dvipdfmx で変換できました。ヒラギノを埋め込む場合、otf パッケージが必須ということでしょうか？
- 2005-07-22 (Fri) 22:21:44 demura? : インストールの手順は、6/23 に MacOSX-WS-10.4.0 にてインストール後、task-tetex で tex 環境をインストールしました。この時点では、問題なく変換できていました。その後、apt-get update で 7.15 バージョンにアップデートして、不具合が発生しました。その後、MacOSX-WS-10.4.1 でのアップグレード、task-tetex の削除、再インストールを試みましたが、不具合は解消していません。また、updmap-otf auto も試してみました。とりあえず、わかるところを報告しました。必要な情報があれば、追加しますので、教えてください。
- 2005-07-22 (Fri) 22:13:56 demura? : まずは、updmap-otf status の結果を報告します。Standby map file : otf-hiraginox、Standby map file : otf-noOpenTypeFont、CURRENT map file : otf-hiraginox でした。
- 2005-07-22 (Fri) 19:49:06 demura? : dvipdfmx 不具合の続きです。task-tetex を入れ直してみましたが、やはり日本語を含む dvi は変換できません。dvips でつくった ps ファイルをプレビューでみたところ、日本語の部分が文字化けしています。
  - 私の所でも変換できませんでした。とりあえず、`\usepackage{otf}` としてみて動くでしょうか？ –seto
  - たった今確認してみましたが、Hiragino, Morisawa 共に埋め込めています。インストールされた手順と以下の結果をお教えください。

    ```
    $ updmap-otf status
    ```
- 2005-07-22 (Fri) 19:16:19 demura? : dvipdfmx で日本語を含む dvi を変換しようとすると、ビットマップフォントを作りにいこうとして失敗し、Could not locate a virtual/physical font for TFM "gbm" という警告が出て止まります。フォントマップのせいかなと（勝手に）想像しているのですが。。。とりあえず、もう一度インストールし直してみます。
- 2005-07-22 (Fri) 19:02:57 demura? : 2005.7.15 バージョンにしたところ、dvipdfmx で不具合が出ました。

— — — — — — — — — — — — — — — — — — —

- システム初期設定の変更ぶん
- ユーザー初期設定の変更ぶんをオプションパッケージとして分離することは難しいでしょうか？ パッケージシステムの動作に必要なのは apt-get と (apt-get を動かすためのパス設定) ぐらいで、それ以外のカスタマイズを必須にする必要はないのでは... と感じています。 – ぜ

- ある程度カスタマイズされた状態からスタートしてもらうということが OSXWS の方針と理解しています。OSX を UNIX として利用しようとしたときに、デフォルトではほとんど白紙に近い状態です。その状態から環境構築をするということは、OSX 固有のノウハウというのも必要になるでしょうし時間と労力が必要になります。個々のソフトウェアを提供する単なるパッケージシステムとしてではなく、「インストールの直後から快適な日本語環境で作業ができるように」という Vine のような方向性を OSXWS が実現できれば良いと思っています。(初期設定の内容については、みなさんから意見をもらって改良されていくだろうと期待しています。) ただ、設定ファイルパッケージをどのように実装するかについては、議論できるかもしれません。現在の実装は/System/Library 以下にある、User Template にドットファイルを追加することを行っています。したがって、すべてのアカウントは作成時から自動的に OSXWS ユーザということになります。さらに、将来もし Apple が User Template の中に設定ファイルを置くようなことがあるとしたら、それと干渉する可能性がでてきます (そういうことは起こりえないのかもしれませんが)。良い手段か分かりませんが、OSXWS では初期設定ファイル群がアップデートされたときに、各アカウントで osxws-upgrade というスクリプトを実行する必要がありますが、最初に初期設定ファイルを持ってくることもこれに統合できないでしょうか？ ¿ tkoba さん。 –seto
- 先ず、銭谷さんのご質問にお答えします。何時もご指摘くださり有り難うございます。さて、OSXWS のスタンスは上で瀬戸さんが書かれた通りです。それと、特に私が想定しているのは、大学や研究機関で数台から数百台規模で計算機環境の面倒を見る立場の人たちです。例えば、研究室に入ってきた院生諸君の計算機環境を仕事ができる様にする状況を考えてください。必要なパッケージ群を入れただけでは作業の半分ほどでしかなく、その後本当に面倒くさい各種設定をこまごまとしないと行けませんし、全て手で作業するのでどうしても一台一台微妙に環境が異なってきます。これはその後のメンテナンスを複雑にする原因になります。こう云った細々した作業は非常に煩わしく、自分の時間をどんどん食い潰して行きます。この問題を解決する為に一念発起して造ったのが OSXWS です。おかげで「OSXWS で hogehoge 入れといてね」で全て済むので管理者としてとても楽をしています。そして、ここが Fink と大きく異なるところでもあります。この様な訳で、システムと各ユーザーの設定ファイル群を予め用意するのは OSXWS としては必須です。–tkoba
- 次に (先ほどお話しした事の備忘録として) 瀬戸さんのご質問ですが、osxws-upgrade は terminal や mlterm を起動するたびに自動で実行されています。また、ユーザー設定ファイル群の扱いも、問題点が明らかになるまで、或はその可能性が出てきた時点で改めて議論しようと思います。– tkoba
- ユーザー初期設定 について：UNIX ツールの入れ方は知っているが自分で入れるのは手間なのでパッケージで楽をしたい、という「手間が惜しいだけの中級者」ユーザーも少なくないと思います。こうしたユーザーにとっては、パスを通す等の設定は自前で済んでいると思いますし、お仕着せの設定を使いたくない気持ちの方が強いと思います。
  - OSXWS を設計した私からすると、その様な状況にも配慮しているつもりです。イストールする前からどの様な設定ファイルが何処に入るか http://www.bach-phys.ritsumei.ac.jp/OSXWS/node11.html で完全に知る事が出来ます。それを見て戴ければ、基本的に殆どの人がする、或はせざるを得ない設定である事をお解り戴けると思います。もしも『これはマズイよ！』と云う設定がありましたら具体的にご教示戴けるとあり難いです。また、瀬戸さんが解説してくださっている様に、これらの設定ファイルがインストール後に編集された場合には OSXWS

が更新時に勝手に上書きする事はしませんから、スキルのあるユーザーがお仕着せの設定を使い続ける必要はありません。この設定ファイルの扱いは VineLinux とほぼ同等で、問題になるとは思えません。–tkoba

- システム初期設定 について：僕は、/etc ファイルのようなシステムのメインの設定を変えてしまうことには抵抗があります。こういう変更は予期せぬ不具合を引き起こしても原因に気づきにくいことが多いですし、現在問題がないにせよ、OS のマイナーアップグレード等で微妙に環境が変わったときに不具合が生じる可能性があります。
  - この問題にも配慮しています。特に Tiger 版からは直接 /private/etc 以下のファイルをいじる事はせず、OSXWS 固有の設定が書かれたファイルを置き、それを各ユーザーの設定ファイルから読み込ませています。これもインストール前に http://www.bach-phys.ritsumei.ac.jp/OSXWS/node11 で完全に知る事が出来ます。ですから、ユーザーが明示的に terminal を叩かない限り不具合が起きる事は無いはずです。もしも『これはマズイよ！』と云う設定がありましたら具体的にご教示戴けるとあり難いです。いずれにしても、VineLinux で叩かれて枯れた方法を用い、web 上に散らばった MacOSX 特有の Tips を集めた形になっていますから、たとえ問題が起きても対応が泥沼化する事は防げるはずです。–tkoba
- 僕にとって、現状の OSXWS は、UNIX ツールを手っ取り早くインストールできるメリットよりも、設定の上書き／お仕着せが怖くてインストールしたくない気持ちの方が強いです。このあたりの線引き（どこまでディストリビューション側で面倒を見るか？）は難しく、最終的には配布者の判断に委ねられるものだと思いますが、私ならここまで踏み込んだ設定変更は行わないと思います。ともあれ、このような優れたディストリビューションを整備して下さり、また、その過程で MacWiki を活用してくださっていることについては、心から感謝しております。– ぜ
- 銭谷さんが危惧されていることを回避することも、これからの方針の１つになる (なってほしい) と考えています。設定ファイルの提供は、あくまでスタート地点と用意するというだけで、自分流の環境構築ができる人に設定を強制するものではないはずです。設定ファイルの扱いについて簡単に解説しますと、まず共通項 (.emacs.el など) とカスタマイズの一例 (.emacs.my.el など) という構成になっています。また、設定ファイルを編集した形跡がなければ、更新があったときに最新のものに書き換えられますが、編集されている場合は上書きはされません (スクリプト osxws-upgrade を見てください)。いずれも、ユーザのカスタマイズについては意識されています。設定について何をデフォルトにするかということは確かに難しく、近いうちに再検討することになっていましたので、今意見をもらえることは有益だと思います。
- 私が VineLinux から OSX に移行したとき、OSX は UNIX を利用する立場からはまさに更地だと感じました。自分で最小限の設定から環境構築できることは、楽しかったし勉強になりましたが、Linux ディストリビューション的な、細かい知識を抜きに即座に利用できる環境も OSX には必要だと思いました。それは OSXWS の目指すところです。–seto
- すいません。そもそものご質問にまだ答えていませんでしたね。『システム初期設定の変更ぶんユーザー初期設定の変更ぶんをオプションパッケージとして分離することは難しいでしょうか？ 』できます。と云うか、最初からそうなっています。インストーラーを作り直す必要があります（プロジェクトのページに方法は解説済み）が、OSX-Preferences パッケージを自分好みの内容に作り替えてしまえばおしまいです。具体的な手順としては、例えば、 My-Preferences の様な名前の rpm package を作り、spec 中で以下の様に 宣言しておけば、OSXWS の中で OSX-Preferences として振る舞ってくれますし、勝手に更新される事も無くなります。–tkoba

```
Conflicts: OSX-Preferences
Provides: OSX-Preferences
```

  - ただし、OSXWS としてはその様な構成まで対応できませんから、各自、或は各団体で行っていただく事になります。そうして姉妹 tree が出来てくれたら本当に嬉しいです。–tkoba
  - OSXWS は OSX の上で、Mac の Cocoa, Carbon 等で書かれた優れたツールと便利な UNIX ツールを同居させて、トータルで現役の物理屋がすぐさま仕事ができる環境を作る事を念頭に置いています。個々のパッケージの寄せ集めではなく、それらを束ねたメタなレベルのパッケージだと言えるかもしれませんし、その様なトータルの環境を提供するのが OSXWS の目的です。ですから、銭谷さんの様に CarbonEmacs package などの単独のパッケージ開発者向きには残念乍らなっていません。しかし、望めばその様なディストリにする事も apt-rpm の柔軟性を利用すれば十分可能です。その場合、Fink との差別化が難しくなるかもしれませんが、必要と感じた方々がいらしゃって、自分専用でも良いので姉妹 tree を作ってくださったら、 本当に望外の喜びです。–tkoba
  - OSXWS の作成段階から銭谷さんにはとても建設的なご意見を戴き、またこの MacWiki に場所を提供してくださったからこそ、現在の OSXWS がある事は明らかです。改めて感謝いたします。 これまで同様、今後ともどうぞ忌憚のないご助言をお願いいたします。–tkoba

- ざっとファイルを眺めてみました。見当違いな点もあるかもしれませんが、例えば次のような構成にはできないでしょうか？ – ぜ
- 1.「OSXWS システム設定」の廃止/private/etc/csh.login-osxws,profile-osxws,zprofile などは/usr/local/bin 等にパスを通すだけのようですが、これは不要では？ – .cshrc などにも同じパス設定が書いてあるようです。
  - 必要です。よく読んでいただければ判りますがパスだけの設定をしてるのではありません。/etc/profile.d/* を読み込ませてもいます。
  - /etc/profile.d/* の下には何が入るのですか？ どこかにその説明は...？
  - 利用するパッケージに依存します。私の環境では以下のファイルが存在します。ですが、、スキルと経験をお持ちで /etc の存在理由を理解している中級ユーザー以上の方でしたら、わざわざ説明するまでもない事だと思っています。–tkoba

```
$ ls /etc/profile.d/
glib2.csh*  glib2.sh*   pgplot.csh* pgplot.sh*  w3m.csh*    w3m.sh*
```

- cron の設定は必要なんですか？
  - teTeX package が使います。
- xinitrc は .xinitrc に置いても良いのでは？
  - これは検討する価値がありますね。
- /private/etc 以下の設定ファイルの存在意義は、一つのパッケージの都合だけを考えていては決して理解できないでしょう。繰り返しますが OSXWS が提供するものはあくまで「総合環境」であって、単なる完

結したパッケージの寄せ集めではないのです。http://www.bach-phys.ritsumei.ac.jp/OSXWS/node7.html に開発当初から書いている内容と今一度照らし合わせてお考え戴ければ幸いです。–tkoba

一口に「楽をする」と云っても 計算機環境に求められるものも好みも千差万別です。
その様な状況で管理者とユーザーの双方が楽をする為には、
それぞれの環境に合わせた `apt-rpm tree` を構築するのが一等です。
`apt-rpm tree` を零から新たに作るのは結構な作業に成りますが、
この `MacOS X WorkShop` をひな形にすれば、数日で実現可能です。
例えば、
デフォルトのログイン環境や `.emacs.el` を変えたければ、
`OSX-Preferences` パッケージを弄るだけで済みますし、
`emacs` に `lisp file` を加えたければ、
`emacs-lisps` パッケージに加えればおしまい です。

- 2.「OSXWS ユーザー設定」のオプションインストール化
  - インストーラーのカスタムインストールオプションで,「OSXWS 推奨設定」のチェックボックスをオン／オフ
    オンにすると「OSXWS ユーザー設定」をインストールする
    オフにすると「OSXWS ユーザー設定」をインストールしない
    apt-get で後から 「OSXWS ユーザー設定」をインストール可能
  - 前にも書きましたが、この様にすると複数の環境をサポートせねばならなくなり、各パッケージをそれぞれの環境をサポートする様に作り替えなければならず、私一人の手には負えなくなります。それは OSXWS の運営としては本末転倒です。上の二点のご指摘からは根本的な不具合が発生する状況は全く想定できません。VineLinux に倣ってユーザーが設定する余地を広く残している現状を、そして、それで全く不都合無く仕事をこなせている現状を捨てて、より複雑化するメリットは見つかりません。–tkoba
  - 上の二点の問題は、それを必要とするユーザーが自分用の My-Prefereces package (中身が空っぽのダミーパッケージでも良いのですから) を作り、それを入れたインストーラーを作れば済む事です。その実作業は 1 時間も掛かりません。OSXWS 内の各パッケージ間には複雑な依存関係がありますから、勿論 OSX-Preferences を弄った結果満足に動作しないパッケージが出る可能性があります。逆に言えば、これまで様々な不具合を克服し、便利な環境を提供するための様々な努力の結果が OSX-Preferences に結実しているとも云える訳です。つまり、OSX-Preferences こそが OSXWS の要のパッケージで、rpm packages を OSXWS たらしめている核なのです。–tkoba
  - ですから、要望としては「こう云う設定をするともっと便利になりますよ」といった内容の方が建設的だと思っています。–tkoba
  - ありがとうございます。現状では自分用のパッケージを作ってまで OSXWS ベースの環境を構築しようとは思いませんが、他のパッケージと使い比べて考えてみたいと思います。– ぜ
  - （補足）今のところ、統合環境を使いたいというより、自分に必要なツールだけをつまみ食いしたい気持ちが強いです。長期的には、このようなニーズにも応えてくれるような方向で OSXWS を発展させて下さることを希望します。– ぜ

    * 仰言る様に独立して利用できるパッケージだけでもつまみ食いできた方が良いのは当然の事だと思います。私自身リソースさえあれば今すぐにでも取り掛かりたいのです。しかし選択肢を増やす事に因るメンテナンスコスト激増の危険は冒せません。本業の片手間で OSXWS をリリースできるのは、この環境を私自身が毎日利用しており常にバグ潰しと改良を続けられる事が確実だからです。OSXWS は実質上私一人でメンテナンスしており、とても本業の片手間に複数の環境をメンテナンスする（即ち手元に複数の環境を用意する事）余裕は無いのです。どうぞご理解ください。今後もしも私自身が手がけるとしても、それは OSXWS の外で姉妹 tree として行う事になるでしょう。多分、つまみ食い版 OSXWS はそれ程手間を掛けずに構築できるとは思いますが、もしもそうでなかった場合に今の私には責任が持てないのです。現在の OSXWS は銭谷さんの様なデファクトスタンダードパッケージの開発者は利用してはならないディストリです。それは私も悔しい程心得ています。しかし同時に Fink や EasyPackage 等も、中立且つ独立のパッケージを作成している銭谷さんはお使いになってはならない筈です。お世話になっている銭谷さんが OSXWS に興味を示してくださり私たちもとても嬉しく、どうにか応えたいのですが、どうぞ私の立場を、そして、何故私が姉妹 tree を欲しがるのかをご理解ください。–tkoba

  – そうですね。開発当初から OSXWS は銭谷さんの様な独立したデファクトスタンダードパッケージの開発者向けではない事は承知していました。それと、私たちはパッケージではなくて、総合環境のディストリビューションを作っている心算です。これは ptetex の土村さんともお話しした事ですが、銭谷さんの様に CarbonEmacs の世界を纏めてくださる開発者と私たちの様に総合環境としてのディストリビューションを構築する開発者とでは、その仕事の階層が異なります。総合環境は個々のパッケージを纏めてくださる開発者が居ないと構築する事が格段に難しくなります。私が MacOS X に移住しようと決意したのは、銭谷さんの CarbonEmacs パッケージの完成度を知ってからの事です。ですから、OSXWS は銭谷さんの CarbonEmacs パッケージが無ければ生まれなかったでしょうし、今でもソースを全て公開してくださっているので本当に簡単に rpm 化できて助かっています。–tkoba

  – また、銭谷さんが環境を弄るのを嫌う理由も解らないではないです。私自身 VineLinux での開発経験が無ければ総合環境としての全体像をイメージできたか怪しかったでしょうし、OSXWS の設計は出来なかったと思います。瀬戸さんも言っていますが、OSXWS は、所謂 Mac のソフトとして Apple が推奨している形から外れていて、『お行儀が悪い』と思われても仕方がない面は確かに有ります。これは OSXWS がそもそも単体のパッケージの概念で括れる存在ではないので仕方が無い事ではあります。Fink は最大限 MacOS X 本体に触れない方針を貫いていて、だからこそ多くのユーザーの要望に最大公約数的にではあるけれども応えられている、と私は考えています。一方で、ある OSXWS ユーザーから以下の様なご意見を戴きました。–tkoba

    `fink` を使うぐらいなら `Intel debian` 機をもう一台用意した方が良い。
    `OSXWS` はその点よく纏まっている。

  – Mac に限らず計算機それ自体にはさして興味は無いけれど、実際に目の前にある Mac を自分の研究の為に使い倒さなければならない人たちにとって、OSXWS は一つの解答を与えている、と自負しています。一方で、この様な議論が展開される事も、問題提議の側面を含めて有意義であると思っています。–tkoba

- UNIX 設定ファイルの問題は、現状の様に、Cocoa, Carbon, UNIX(terminal, X11) と仕事で使うツールの出自がバラバラである事が原因です。瀬戸さんとも話していますが、将来全て hoge.app だ

けで仕事が済むのであれば、このような問題から大幅に解放され、上の『「OSXWS ユーザー設定」のオプションインストール化』も簡単に実現できるでしょう。それまではやはり VineLinux で実績のある現在の方法が最も確実だと思います。–tkoba

— — — — — — — — — — — — — — — — — —

- 2005-07-16 (Sat) 00:01:40 okamonn? : ありがとうございます。すべて日本語で表示できました。お手数かけました
- 2005-07-15 (Fri) 13:21:57 tkoba : Ngraph, gtk+ を修正して ja_JP.UTF-8 locale 下で日本語表示できるようにしました。
- 2005-07-15 (Fri) 11:45:23 tkoba : mdview も文字化けしていますね。直しておきます。
- 2005-07-15 (Fri) 11:23:31 okamonn? : すみません。以前 scigraphica の要望したものです。今回 ngraph をいれ使用を始めたのですが、文字化けを起こしてしまいます。OS ごと再インストールしても文字化けを起こします。ただなぜか WINDOW のところだけが文字化けしないのです。OS は 10.4.2 です。何かし忘れているのでしょうか?
  - ご報告ありがとうございます。locale を ja_JP.UTF-8 に変更してから Ngraph の対応をしていませんでした。早速修正します。(scigraphica もあれこれ弄っていますが、どうも Python-numarray との間がうまくいきません) – tkoba

13.4 10.4-1 公開迄

- 2005-06-29 (Wed) 02:15:10 fu7mu4? : MacOSX-WS-10.4.0.dmg をダウンロードして、インストールをこころみたところ、以下のメッセージがでて、インストールできません。どうすればいいのでしょうか。フォーマット拡張（大文字小文字区別、ジャーナリング）です。MacOSX WorkShop? start kit がこのコンピューターにインストールできません。インデックス 18 に対してしていされたメッセージがみつかりませんでした。
  - うーーん。。。　PackageMaker どうなってるんでしょうねぇ。一応各メッセージの対応表を載せておきますが、ReadMe を読んでいれば回避できる内容の筈です。–tkoba

    ”16” = ”既に apt がインストールされていますので、インストールをする必要はありません。”;
    ”17” = ”X11 と X11 開発環境を先にインストールしてください。”;
    ”18” = ”Xcode 開発環境を先にインストールしてください。”;
    ”19” = ”Xcode, X11 と X11 開発環境を先にインストールしてください。”;
    ”20” = ”このパッケージは MacOS X 10.4 (Tiger) が必要です。”;

  - PackageMaker は今までスクリプトで処理をしてきたが GUI でビルドすれば動く事が判った！！全く勘弁してくれょ。–tkoba
  - 確かに Xcode のインストールをとばしておりました。Xcode をインストール後は WorkShop を無事インストールできました。ありがとうございました。–fu7mu4
- 2005-06-24 (Fri) 08:56:25 demura? : アドバイスをありがとうございます。早速、試してみます。

- 2005-06-23 (Thu) 16:15:34 demura? : 質問です。ターミナルで、日本語ファイル名を表示することは可能でしょうか？ 以前は、文字コードエンコーディングを Unicode にすることで、対応していました。MacOSX_WorkShop を利用するために、今は EUC にしています。
  - そうですね。。ターミナルで UNIX のコマンド類を殆ど使わないのであれば、文字コードエンコーディングを UTF-8 にしても構いません。そのうえで環境変数を LC_ALL=C としてコマンド使用時のメッセージを英語にすれば、文字化けも減ると思います。いろいろ試してみてください。–tkoba
- 2005-06-23 (Thu) 16:12:06 demura? : 先日より利用を始めました。tex 等の環境が一発で構築でき、本当に感激、感謝しています。
  - ご感想を有り難うございます！ 楽に仕事の環境が構築されたとしたならば、私も骨を折った甲斐があります。–tkoba
- 2005-06-16 (Thu) 16:25:41 okamon? : 要望です。gtk+,pango 等パッケージ化されているのでしたら scigraphica をパッケージ化してほしいです。複数軸グラフかけるソフトが、市販の IGOR Pro しかないので頼みます。インストールしようとがんばりましたが、挫折しました。
  - コメント有り難うございます。Ngraph と似ていますが、なかなか有用そうなので簡単に入れられそうならパッケージングしてみます。 –tkoba
  - 一応パッケージングしました。$ sudo apt-get install scigraphica して試してみてください。web を見て python-numeric を利用するのかと思ったら numarray を要求されたり、と、なかなか複雑で、キチンと動くかどうか判りません。ので、動作確認して戴いて使い物にならないようならリリースからは外そうと思っています。–tkoba
  - 駄目ですね。時間が取れたら一寸弄ってみます。–tkoba
- 2005-06-04 (Sat) 03:11:45 fu7mu4? : 要望です。checkinstall を追加してくれると助かります。一般的な tarball 等から rpm を作成する package 管理ソフトです。
  - コメント有り難うございます。早速調べてみました。確かに、自分でパッケージングする敷居を下げられそうなので、追加しようと思います。 –tkoba
  - コードを見ていますが、Linux 決め打ちで書かれていますね。。時間が取れたら作業します。–tkoba
- インストーラを 10.3.9 上で作り直し、インストールできる事を確認しました。不具合でインストールできなかった方はダウンロードし直して試してみてください。
- テスト版を公開します。バグ報告は歓迎しますが、迅速な対応は期待しないでください。
- iMac G5 を研究室で買って貰ってから、仕事の半分以上は Tiger 上でしている。いい加減に TeX 環境を構築しないと能率が上がらない。と云う訳で、土村さんにお世話になり乍ら何とか teTeX3 で構築。できれば今週末にテスト tree を公開したいものである。
  → 0612 一応公開
- ここ暫く本業が忙しくて開発が停滞気味。先週一年生の iBook G4 に Tiger 版のテスト tree を入れてみたが、どうも Xcode 2.0 の PackageMaker の出来が今ひとつで、ドキュメント通りに用意したメッセージを探し出せなかったり、管理者のパスワードを要求せずにインストールを始めてしまって

すぐにコケたりした。その同じインストーラーでキチンとインストールできる事もある。近々 10.4.2 が出るそうだけれど、Xcode 周辺の更新にも期待したい。
→ インストーラは当面 10.3.9 上で作成することにした。

- 取り敢えず Emacs 周辺までのパッケージとインストーラーをテストとして作りましたが、試してみたい方いらっしゃいますか？
- 銭谷さんの 20050507 版を基に Carbon Emacs が動き出した。後は TeX 周辺だ。
→ 0615 TeX 関連全面解決
- rpm-4.4.1 には MacOS X 向けのパッチが取り入れられ、ソースの管理は楽になった。が、neon や libselinux を要求したりと、これはこれで面倒くさい。一応これで暫くパッケージングしてみて不安定であれば rpm-4.3.2 で行こう。
→ rpm-4.3.2 で行くことにした。gnupg の問題は解決。

14 謝辞

MacOS X WorkShop は、以下の個人/団体 (順不同) に多大な御指南/御協力を戴いたり、
公開されているパッケージや議論を参考にさせて戴きました。
この場を借りて関係各位に感謝の意を表します。

藤井恵介さん	MacOS X WorkShop の下地を築いてくださいました。
土村展之さん	ptetex3 パッケージ
内山孝憲さん	Mxdvi とそのフォントパッケージ
齋藤修三郎さん	OTF パッケージ
銭谷誠司さん	CarbonEmacs パッケージ、 Mac Wiki
kenichi kikuchi さん	kinput2 ことえりパッチ
MacWiki	-
Project PINEAPPLE の皆さん	-
Vine Linux の皆さん	-

更新履歴

- Wed Jul 03 2008 KOBAYASHI Taizo
  - 「過去の議論」に dot emacs 関連を追加記入
- Mon Dec 31 2007 KOBAYASHI Taizo
  - Version 10.4-3
  - 「過去の議論」追加記入
- Fri Sep 01 2006 KOBAYASHI Taizo
  - 「過去の議論」追加記入
  - 00-News を追加
- Fri Mar 03 2006 KOBAYASHI Taizo
  - `~/.emacs.el` の内容を `site-start.d` 内に移動
- Thr Feb 16 2006 KOBAYASHI Taizo
  - 「Remote Install」追加
- Mon Feb 06 2006 KOBAYASHI Taizo
  - 「過去の議論」追加記入

  Wed Feb 01 2006 KOBAYASHI Taizo
  - Version 10.4-2 for PowerPC/Intel
  - パッケージの大部分を Universal Binary 化
  - Intel Mac に対応
    binary package は i386, fat, ppc, noarch の組み合わせで行きます。
  - アンインストールをサポート
    以下のコマンドとそれに続く確認に了承すればアンインストールできます。

    ```
    $ sudo apt-get remove OSX-system
    ```

  - 英語環境を睨んで
    各ユーザーの dot files を/System/Library/User Template/Japanese.lproj/ から /Library/Application Support/OSXWS/jp/ へ移動。この結果 OSXWS インストール後に新規ユーザーを作成しても OSXWS とは切り離された素のユーザー環境が作られます。その新規ユーザーが OSXWS を利用したい場合は以下のコマンドを実行して dot files を整えてください。

    ```
    $ /usr/local/bin/osxws-upgrade
    ```

  - ## パッケージ追加情報

    * clamav, gmp
      最近迄猛烈に忙しく大変に遅くなりましたが ClamAV を packaging しました。daemon の扱いを MacOSX に準拠させ/Library/StartupItems?/clamav/ 以下に起動と停止のスクリプトをおきました。自動で clamd, freshclam が daemon として動きます。
    * cmucl, Maxima, Imaxima, clisp(test tree)
      デフォルトの lisp を cmucl に変更して Maxima を復活させました。 test tree に clisp と maxim-exec-clisp を置いておきますが clisp はメンテナンス対象外です。
    * fugu
      Cocoa で書かれた sftp client
    * fftw3
      研究で必要になったから
    * Desktop Manager
      一年以上利用しているのと source が tar ball で配布されたので packaging しました。
    * ImageMagick
      やはり無いと不便であるから。
    * synaptic
      これで GUI でパッケージ管理出来ます！ 関連して gtk2 も用意しました。起動 (mlterm 上) とマニュアルの表示は以下で行ってください。

      ```
      $ sudo synaptic
      $ open /usr/local/share/synaptic/html/index.html
      ```

    - gcc-g95
      gcc-g77 と排他利用になりますが用意しました。

  - 変更したパッケージ
    - ispell から aspell
    - LatexEquationEditor から LaTeXiT
      今後の発展を見込んで移行。ただし LatexEquationEditor のサポートも続けます。お好みに応じて使い分けてください。
    - kterm から mlterm
      locale を ja_JP.UTF-8 へ変更するに伴い移行。
    - eTeX-3 ベースに更新
      dvipdfmx と齋藤さんの OTF パッケージを自動で組み込む updmap-otf の調整に手間取ったが、漸く仕事で使える様になった。TeX 関連では、昨日瀬戸さんと議論の上、.emacs.el から yatex に関する記述を .yatex.el へ移した。
    - ghostscript の version は 8.51 で組んでみることにした。ヒラギノをデフォルトにしました。
    - less から lv

  - 削除したパッケージ
    - vim
      vim は multi_byte でコンパイルされている。`~/.vimrc` を弄って利用可
    - bizp2
    - freetype

- Wed Jul 20 2005 KOBAYASHI Taizo
  - 各ページの文章の誤りを訂正。
- Fri Jul 15 2005 KOBAYASHI Taizo
  - Version 10.4-1

  - Tiger 版リリース
- Wed Jan 19 2005 KOBAYASHI Taizo
  - 各ページの文章の誤りを訂正。
- Thu Nov 25 2004 KOBAYASHI Taizo
  - Version 10.3-7

  - Installer ver. 10.3f
    rpm-4.3.2 をはじめとする全パッケージの更新。

  - パッケージ追加情報
    - Ngraph-6.3.30-10.3tk1

    - xgraph-12.1-10.3tk1
      - xgraph11 から修正パッチを移植しました。百害あって一利無しのアニメーション機能は削除してあります。

  - パッケージ更新情報
    - OSX-Preferences-10.3-1tk12
      - .bashmyrc, .cshmyrc, .emacs.my.el, .zshmyrc の追加。.*myrc や .emacs.my.el を既に書いている人は以下のディレクトリから該当するファイルを参照して書き直して下さい。
      /System/Library/User Template/Japanese.lproj/
    - emacs-21.3.50-10.3tk11.5
      - CVS 20041123, inline_patch-20041101
    - OSX-Preferences-10.3-1tk16
      - fix osxws-upgrade script
    - kterm-6.2.0-10.3tk5
      - background:wheat, foreground:black に設定
    - kinput2-v3.1-10.3tk2
      - Cmd+Space で日英切り替え出来るように設定。modeLocation を kterm の左下に出るように設定。
- Thu Nov 11 2004 KOBAYASHI Taizo
  - rpm2html による RPM データベースのページを追加
  - 各ページの文章の誤りを訂正。

- Tue Oct 26 2004 KOBAYASHI Taizo
  - Installer の ReadMe.rtf, License.rtf を書き換え GPLv2 である事を明示。
  - Subsection 「ライセンス」追加
- Sun Oct 24 2004 KOBAYASHI Taizo
  - Version 10.3-6
  - Installer ver. 10.3d
    gettext, beecrypt, bzip2, OSX-Preferences 更新に伴う更新。
  - Section 「過去の議論」を追加。
  - パッケージ追加情報
    * w3m, w3m-el
      kterm 上で画像を表示する場合は w3m-img をインストールして下さい。
    * gtk+, glib, gdk-pixbuf, imlib, libungif
      w3m-img の為に導入。
    * OSX-keyring
      パッケージに gpg 署名をする為の鍵束。
    * kotonoko
      コトノコ [47] ver 1.0-beta26
    * vim
      vim-6.3.31 (huge,big,normal)
      kterm 上で利用する vim
      terminal での日本語入力はダメ。
  - パッケージ更新情報
    * emacs-21.3-10.3tk10
      - CVS 20041024, inline_patch 20041015
    * tetex-2.0.2-10.3tk5
      - remove TEXMF/dvips/base/config.ps
    * OSX-Preferences-10.3-1tk10
      - added rpm/BUILD dir
- Wed Oct 13 2004 KOBAYASHI Taizo
  - Version 10.3-5
  - Installer ver. 10.3c
    carbon-font.el の改訂に伴い Ayuthaya.ttf に関する記述を変更。
  - dot files の更新
    OSX-Preferences 更新の際に各ユーザーの設定ファイルを更新する osxws-upgrade scriput を同梱。
- Tue Oct 12 2004 KOBAYASHI Taizo
  - Version 10.3-4
  - .emacs.el の更新
    font-lock の導入と YaTeX 使用時の skk 環境の整備
  - urw-fonts をインストールする際の warning についてを「10 既知の問題点」に追加
- Sun Oct 10 2004 KOBAYASHI Taizo
  - Version 10.3-3
  - Installer ver. 10.3b
    postintall script で無駄な .*.rpmorig を作らない様に修正
  - .emacs.el の更新
    bibtex-command ”jbibtex -kanji=euc” 追加（坂田君）
    ”set-terminal-coding-system” を ’utf-8 から ’euc-jp-unix へ変更
    terminal や kterm で -nw mode を利用できる様にしてみました。
    ただし、ことえりではなく SKK を利用して下さい。

[47] http://www.afternooncafe.jp/kotonoko/

- Mxdvi-fonts の更新
  オリジナルの *.hqx を *.sitx で作り直しました。

- Thu Oct 07 2004 KOBAYASHI Taizo
  - Version 10.3-2
  - Installer ver. 10.3a
  - skk, skkdic, skktools 追加
  - OSX-Preferences-10.3-1tk5
    fixed typo in .bashrc
  - emacs-21.3.50-10.3tk7
    cvs-20041005
  - texmacro-otf
    updmap-otf ver. 0.5
    利用可能な font map のみを status で表示する様に修正

  ToDo
  - .emacs.el 関連の更なる調整。
    これは Mac Wiki で議論し乍ら発展させよう。

- Wed Sep 29 2004 KOBAYASHI Taizo
  - Version 10.3-1
  - 設定ファイル {/private/etc/something, $HOME/.something} の内容を追加。(Thanks. 銭谷さん)

  ToDo
  - .emacs.el 関連の更なる調整。
    これは Mac Wiki で議論し乍ら発展させよう。

- Thu Sep 23 2004 KOBAYASHI Taizo
  - Version 10.3
    公開版
  - apt-rpm tree の作成方法を追加

  ToDo
  - .emacs.el 関連の更なる調整。
    これは Mac Wiki で議論し乍ら発展させよう。

- Wed Sep 22 2004 KOBAYASHI Taizo
  - Version 1.0
  - installer の作成方法を追加

  ToDo
  - .emacs.el 関連の更なる調整。
    これは Mac Wiki で議論し乍ら発展させよう。

- Mon Sep 20 2004 KOBAYASHI Taizo
  - Version 0.99
  - installer の version を 10.3 へ変更。
  - zsh の設定ファイルを追加 (新山君)
  - pTEX3.1.4, mendex-2.5a, etc..
  - emacs Sep 19 CVS

  ToDo
  - .emacs.el 関連の更なる調整。

- Tue Sep 07 2004 KOBAYASHI Taizo

- Version 0.9
- スクリーンショット追加。
- パッケージメモ以外はほぼ完成？

ToDo

- installer の version を 10.3 へ変更。
- .emacs.el 関連の更なる調整。

- Mon Aug 23 2004 KOBAYASHI Taizo
  - 最初の版

索 引