MacOS X WorkShop (OSXWS)

— TeX, Emacs, OpenFOAM on MacOS X with apt-rpm —

10.10-1

KOBAYASHI Taizo

2016/2/02

最新情報は **Mac Wiki**[1] をご覧下さい。
過去の各版はこちら
10.9 Maverics[2] **10.8 MountainLion**[3] **10.7 Lion**[4]
10.6 SnowLeopard[5] **10.5 Leopard**[6] **10.4 Tiger**[7] **10.3 Panther**[8]

Emacs, TeXLive, gnuplot,,,
Web を彷徨ってインストール。
ドットファイル群を設定して、、、、
でも、動かなかったり、更新したら動かなくなったりする。。。。

web で普通に目にします。

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？

Web 上に散らばったこれらのノウハウを凝縮したもの
それが OSXWS です。

[1] http://macwiki.sourceforge.jp/wiki/index.php/OSXWS/10.10
[2] ../Maverics/
[3] ../MountainLion/
[4] ../Lion/
[5] ../SnowLeopard/
[6] ../Leopard/
[7] ../Tiger/
[8] ../Panther/

OSXWS は開発メンバーの仕事環境ですから、
インストール後すぐさま仕事に入れます。
OS やパッケージを更新しても環境は維持されます。

ただし、このディストリビューションの成果物を利用して不具合が生じても、
ディストリビューションとしてもディストリビューションに関係する如何なる人間も
一切責任を負いません。
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、
返事も対応もありません。

目 次

図 目 次

1　更新情報

1.1　10.10-1 の変更点

- Maverics 版からの変更は「Maverics 版からの変更点 (Section9 参照) 」をご覧ください。

2　はじめに

　そもそもディストリビューターの本分は、
機能群としてのソフトウェア群の配布ではなくて、
ソフトウェア群を機能させることです。
　このディストリビューションの直接的な目的は、
TeX や emacs を用いて仕事をしている人が、
OSX 上にストレス無く即座に仕事に掛かれる環境を構築し、且つ、計算機のメンテなすから解放される事
と
大学や研究機関等の計算機管理者が、
OSX 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。
　つまり、OSXWS は環境を提供する のであり、
個々のソフトウェアの粒度でのデフォルト設定等は議論の対象外です。
　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**[9] です。
この Vine Linux の中で日々の仕事に必要なパッケージを OSX に合わせて構築し直した物と、
OSX 上の便利なソフトを組み合わせたものが OSXWS の実態です。
現在公開しているパッケージは

- **MaxOS X 10.10.x (Yosemite)** 対応版 [10]
- **MaxOS X 10.9.x (Mavericks)** 対応版 [11]
- **MaxOS X 10.8.x (MountainLion)** 対応版 [12]
- **MaxOS X 10.7.x (Lion)** 対応版 [13]
- **MaxOS X 10.6.x (SnowLeopard)** 対応版 [14]
- **MaxOS X 10.5.x (Leopard)** 対応版 [15]
- **MaxOS X 10.4.x (Tiger)** 対応版 [16]
- **MaxOS X 10.3.x (Panther)** 対応版 [17]

です。
尚、今後 MountainLion 以前の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 (Yosemite)**[18]

[9] http://www.vinelinux.org/
[10] ../Syrah/index.html
[11] ../Mavericks/index.html
[12] ../MountainLion/index.html
[13] ../Lion/index.html
[14] ../SnowLeopard/index.html
[15] ../Leopard/index.html
[16] ../Tiger/index.html
[17] ../Panther/index.html
[18] ../Syrah/rpm2html/

- **RPM 解説データベース (Mavericks)**[19]
- **RPM 解説データベース (MountainLion)**[20]
- **RPM 解説データベース (Lion)**[21]
- **RPM 解説データベース (SnowLeopard)**[22]
- **RPM 解説データベース (Leopard)**[23]
- **RPM 解説データベース (Tiger)**[24]
- **RPM 解説データベース (Panther)**[25]

をご利用ください。

OSXWS 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？

個々のソフトウェアを開発する人達をゼネコンさんとるすと、
apt-rpm は差詰め行政の様なものです。
そして OSXWS が提供する「環境」は都市の様なもの です。
都市を造るのにゼネコンさんだけでできる訳がありませんよね。
ですから「行政」担当の apt-rpm が必要になるのです。
もっと現実的なご利益を言えば
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-texlive
$ 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、

[19] ../Mavericks/rpm2html/
[20] ../MountainLion/rpm2html/
[21] ../Lion/rpm2html/
[22] ../SnowLeopard/rpm2html/
[23] ../Leopard/rpm2html/
[24] ../Tiger/rpm2html/
[25] ../Panther/rpm2html/

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。

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。

OSX 上での UNIX 研究環境を構築するには **Fink**[26] をはじめ、
琉球大学の **EasyPackage**[27] や、**MacPorts**[28]、**Homebrew**[29]、等があり、
それぞれみな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**[30] 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。

他のディストリビューションと OSXWS との違いは発想と目的にあります。
大抵のディストリビューションは、OSX に標準では準備されていない
UNIX アプリを手っ取り早くインストールするのが目的に見えます。
ですが OSXWS は違います。
OSXWS はすぐに仕事ができる環境の提供が目的です。
つまり、
殆どのディストリビューションは「大工さん的な発想」を引きずっている
のに対して、

[26] http://www.finkproject.org/
[27] http://www.ie.u-ryukyu.ac.jp/darwin2/
[28] http://www.macports.org/
[29] http://mxcl.github.com/homebrew/
[30] http://macwiki.sourceforge.jp/cgi-bin/wiki.cgi

OSXWS は徹底して「行政」の発想を貫いています。
OSXWS は、OSX と云う国の中の一省庁の体裁を保っており、
徒に規模を大きくする事はしません。
これが
ごく少数の開発者だけでも開発を続けいける状況
を実現しているのです。

OSXWS は、開発者たちが日々の仕事をする為に作っています。
ですから、
OSXWS は開発者たちが仕事をしている限り続きます。

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[31] は **Red Hat**[32] 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[33] 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、
OSXWS 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[34] は **Debian GNU/Linux**[35] 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、

[31] http://www.rpm.org/
[32] http://www.redhat.com/
[33] http://www.vinelinux.org/
[34] http://www.debian.org/doc/manuals/apt-howto/
[35] http://www.jp.debian.org/

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。
- 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[36] は **Conectiva Linux**[37] 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。

OSXWS 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「OSXWS の使い方 (Section5 参照) 」を御覧下さい。

2.2 ディストリビューションのポリシー

この OSXWS ディストリビューションには、以下のポリシーがあります。

- ソフト開発の都合（大工さん的発想）を持ち込まない
- 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ディストリビューションを立ち上げやすくする

です。

管理者がたった一人でも `MacBook` 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留めます。

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、OSXWS の目的でありポリシーです。

[36] https://moin.conectiva.com.br/AptRpm
[37] https://moin.conectiva.com.br/

2.3 派生ディストリビューションの歓迎

OSXWS 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ディストリビューションをひな形にした
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。

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、
OSXWS 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

OSXWS に関する議論や連絡は **Mac Wiki**[38] を利用させて戴いています。

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。
.emacs.d/init.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 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。

2.5 メンバー

２０１６年２月現在のメンバーです。(順不同)

小林泰三 帝京大学　准教授・九州大学　研究員

瀬戸亮平 OIST, 数理ソフトマターユニット、グループリーダー

山本宗宏 Green Cherry Ltd. 代表, Project Vine

新山友暁 金沢大学理工研究域　博士研究員

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

2.6　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/osxws/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。

3 姉妹 trees !!

OSXWS の具体的な目的は、
TeX や emacs を用いて仕事をしている人が、
OSX 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、
OSX 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。
（那覇が好きな人が居れば、稚内が馴染む人も、京都が一等！ と云う人も居ます。）
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 OSXWS 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、OSXWS の apt-line (/usr/osxws/etc/apt/sources.list.d/ 内に設置) 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**[39]
  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**[40]
  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。

[39] http://www-jlc.kek.jp/ fujiik/macosx/10.9.X/HEPonX/
[40] http://www.bach-phys.ritsumei.ac.jp/OSXWS/

4　OSXWS をインストールする

このセクションでは OSXWS をインストールする手順について説明します。
尚、以前の MacOS X に関する情報は

- **MacOS X 10.9 (Maverics)**[41]
- **MacOS X 10.8 (MountainLion)**[42]
- **MacOS X 10.7 (Lion)**[43]
- **MacOS X 10.6 (SnowLeopard)**[44]
- **MacOS X 10.5 (Leopard)**[45]
- **MacOS X 10.4 (Tiger)**[46]
- **MacOS X 10.3 (Panther)**[47]

をご覧ください。

4.1　インストールする前に…

警告！

OSXWS のインストール環境は、素の OSX 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 や MacPorts, Homebrew との共存は可能かもしれませんがディストリビューションとしてはサポート外です。また、他の方々が配布されている emacs, TeXLive 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。

OSXWS は OSX 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 OSX のインストール条件を満たす必要があります。

- `X11, (Xquartz)`[48]
  mlterm, gnuplot, xgraph, yaplot.... 等の X11 のソフトを利用するのに必要です。
  version 2.7.7 以上をインストールしてください。

- `Xcode`
  AppStore 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。インストールするの後で `Command Line Tools` を必ずインストールしてください。
  terminal で以下のコマンドを実行して、その後の指示に従ってください。

```
$ xcode-select --install
```

[41] ../Maverics/index.html
[42] ../MountainLion/index.html
[43] ../Lion/index.html
[44] ../SnowLeopard/index.html
[45] ../Leopard/index.html
[46] ../Tiger/index.html
[47] ../Panther/index.html
[48] http://xquartz.macosforge.org/

また、必須事項ではないものの、
以下の様に「大文字／小文字を区別する」様にディスクをフォーマットし直す事をお勧めします。(OpenFOAM を利用する場合には必須です。)

- Mac OS 拡張 大文字／小文字を区別する、ジャーナリング
  OSX をインストールする際にディスクユーティリティを呼び出して、
  ディスクフォーマットを大文字と小文字を区別するように変更してください。

4.2 Install

OSXWS を始めるには
MacOS X WorkShop start kit MacOSX-WS-10.10.1.dmg[49]
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-10.10.1.tar.bz2**[50] です。)

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「OSXWS 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以下の処理を内部で自動で行います。

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 や rpm 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。
   /usr/osxws/etc/{csh.login-osxws,profile-osxws,zprofile} が加えられ、/usr/osxws/bin 等にパスを通します。

   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**[51] にてご確認ください。

[49] MacOSX-WS-10.10.1.dmg
[50] MacOSX-WS-10.10.1.tar.bz2
[51] OSX-system/

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。

- .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。
- .custom_osxws.el
  emacs の設定ファイルです。

- .bibdesk.d
  BibDesk で利用するディレクトリです。
  journal title abbreviation file は .bibdesk.d/jtabbrv.d/ に*_abbrv.txt の形でおいてください。
- .emacs.d
  emacs で利用するディレクトリです。
  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。
- .vimrc
  vi の設定ファイルです。
- .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。
- 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-10.10.tar.bz2**[52] 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
`/usr/osxws/share/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。

4.3 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 [53] をマウントします。

```
$ hdid MacOSX-WS-10.10.1.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-10.10.1/MacOSX\ WorkShop\ start\ kit.pkg -ta
$ hdiutil eject /Volumes/MacOSX-WS-10.10.1
```

3. 再起動
   再起動します。

```
$ sudo reboot
```

4.4 Maverics 以前の版からの Upgrade

警告！
新規インストールを推奨します。

[52] OSX-Preferences-10.10.tar.bz2
[53] MacOSX-WS-10.10.1.dmg

4.5 Uninstall

警告！

OSXWS のみをインストールしている状況を想定しています。
/usr/osxws 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
```

4.6 Install 後にユーザーを追加したとき

追加したユーザーで以下を実行します。

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

5　OSXWS 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、
`/usr/osxws/etc/apt/apt.conf` 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は
OSX-base パッケージを apt でインストールします。

OSXWS 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 dist-upgrade
$ 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。
Emacs 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 Cocoa 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/OSXWS/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 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 UTF-8 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-texlive
$ 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。
> TeX は ptexlive/UTF-8 で make されています。
> terminal 上で emacs で作成したファイルをコンパイルする時には
> platex2pdf 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 版以降には apt の GUI frontend である **Synaptic**[54] を用意しました。

```
$ 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 [55] 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 [56] 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

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

[54] http://www.nongnu.org/synaptic/
[55] file:///usr/osxws/share/synaptic/html/index.html
[56] 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

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
aspell-el - Emacs lisp for aspell
ctags - A C programming language indexing and/or cross-reference tool.
emacs - GNU Emacs エディタ
emacs-git - Emacs の Git サポート
gnuplot-lisps - gnuplot mode lisp files for emacsen
mercurial-el - Mercurial バージョン管理システム用 Emacs サポート
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.
apel - Emacs 用の 基礎的な関数を提供するライブラリ
autoconf265-mode - Emacs-lisp autoconf-mode for autoconf/autotest
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。
rst-el - reStructuredText の Emacs サポート
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 - 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
1116kB のアーカイブを取得する必要があります。
展開後に 4826kB 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://liberty.cc.kyushu-u.ac.jp MountainLion/x86_64/main psutils 1.17-12osx10.8 [67.3kB]
取得:2 http://liberty.cc.kyushu-u.ac.jp MountainLion/x86_64/main a2ps 4.14-2osx10.8 [1049kB]
1116kB を 0s 秒で取得しました (12.3MB/s)
変更を適用しています...
準備中                                  ############################## [100%]
更新/インストール中
  psutils-1.17-12osx10.8.x86_64          ############################## [100%]
  a2ps-4.14-2osx10.8.x86_64              ############################## [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、eb を削除したい場合

```
$ sudo apt-get remove eb
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:
  eb eb-devel kotonoko
アップグレード: 0 個, 新規インストール: 0 個, 削除: 3 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 2887kB 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%]
クリーニング/削除中
  eb-devel-4.4.3-2osx10.8.x86_64         ############################## [100%]
  kotonoko-2.1-1osx10.8.x86_64           ############################## [100%]
  eb-4.4.3-2osx10.8.x86_64               ############################## [100%]
完了
```

の様になります。

ここで eb に依存している eb-devel, kotonoko が存在する場合、それら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。
OSXWS 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
```

開発に携わるには、常に apt-get dist-upgrade して最新の環境にしなければなりません。

5.1.6 後片付け

apt-get で取得したパッケージは
`/usr/osxws/var/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
Version     : 4.14
Release     : 2osx10.8
Architecture: x86_64
Install Date: 月  5/13 18:29:47 2013
Group       : Applications/Publishing
Size        : 4575209
License     : GPL
Signature   : DSA/SHA1, 月  5/13 12:50:16 2013, Key ID f367e1515c69cada
Source RPM  : a2ps-4.14-2osx10.8.src.rpm
Build Date  : 月  5/13 12:50:10 2013
Build Host  : macprol2010
Relocations : (not relocatable)
Packager    : KOBAYASHI Taizo <tkoba965@mac.com>
Vendor      : MacOS X WorkShop
URL         : http://www.inf.enst.fr/~demaille/a2ps/
Summary     : Converts text and other types of files to PostScript(TM).
Description :
The a2ps filter converts text and other types of files to PostScript(TM).
A2ps has pretty-printing capabilities and includes support for a wide
```

```
number of programming languages, encodings (ISO Latins, Cyrillic, etc.),
and medias.
```

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
/usr/osxws/bin/a2pdf
/usr/osxws/bin/a2ps
/usr/osxws/bin/a2ps.bin
/usr/osxws/bin/card
/usr/osxws/bin/composeglyphs
...................
/usr/osxws/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-1.23-1osx10.9.x86_64.rpm をインストールするには

```
$ sudo rpm -ivh hoge-1.23-1osx10.9.x86_64.rpm
```

或は

```
$ sudo rpm -Uvh hoge-1.23-1osx10.9.x86_64.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
```

を実行します。

6 rpm パッケージを開発する

お願い！

ディストリビューションとしてのパッケージ開発は、
「環境」という都市を開発する様なものです。
ライブラリの依存関係から OSXWS としてのデフォルト設定まで
ディストリビューションとしての整合性・一貫性に気を配って下さい。

ここでは OSXWS 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。

OSXWS 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**[57] や、
Momonga Linux の **Specfile-Guidance**[58] 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-1.0-1osx10.10.src.rpm
```

を実行すると、
spec file は `~/rpm/SPECS` に、
source files は `~/rpm/SOURCES` 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。
パッケージを作るには

```
$ cd ~/rpm/SPEC
$ rpmbuild -ba hoge-osx.spec
```

[57] http://www.vinelinux.org/docs/vine6/making-rpm/vine-making-rpm.html
[58] http://www.momonga-linux.org/docs/Specfile-Guidance/ja/index.html

すると、hoge の source package が ~/rpm/SRPMS に作成され、binary package が ~/rpm/RPMS 以下の適当なディレクトリに作成されます。

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 /my/home/dir/rpm
%packager KOBAYASHI Taizo    <xxxxxxx@xxxx.xxx>

%_tmppath %{_topdir}/temp
%_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。
OSXWS の開発に参加を希望される方は、gnupg の public key をご連絡ください。
ご相談のうえ参加戴ける場合には OSX-keyring に登録いたします。

6.2 spec file

ここでは OSXWS 固有の spec file に関する方針を述べます。
spec file は OSXWS のものと判別し易くする為に、
ファイル名を **(Name)-osx.spec** にして下さい。

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。
OSXWS では VineLinux に倣い

MountainLion (release number)osx10.10

と付ける事にします。

(architecture) は特に指定しなければ x86_64 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)`
を標準にします。

6.3 rpm macro

ここでは OSXWS 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/osxws/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 OSXWS 固有のものを列挙します。

_prefix `/usr/osxws`
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/osxws` 以下にインストールします。

_var `/usr/osxws/var`

_sysconfdir `/usr/osxws/etc`

次に、OSXWS のみに存在するマクロを列挙します。

```
%_dist_release osx%(sw_vers | grep ProductVersion | cut -f2  | cut -f1,2 -d.)
%_rpm_platform32 i686-apple-darwin%(uname -r | cut -f1 -d.)
%_rpm_platform64 x86_64-apple-darwin%(uname -r | cut -f1 -d.)
```

6.4 その他

ライブラリに関する問題
出来るだけ OSX 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 10.7.4 + Xcode 4.4 以降、libtool, autotools は用意されていません。

OSXWS では、glibtool, automake-1.14.1, autoconf-2.69 を用意しています。
これらは必要に応じて、`aclocal-1.14 -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。

X11

Apple からは配布されなくなりました。
(Xquartz)[59] を利用します。

[59] http://xquartz.macosforge.org/

7　インストーラを作る

OSXWS 10.10 のインストーラを Mavericks 上の Xcode-7.1 で開発した際の備忘録です。
productbuild/pkgbuild はまだ利用していません。
総合的且つ正確な情報は Apple の **PackageMaker User Guide**[60] 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 (Section4.2 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
Auxiliary Tools for Xcode[61] に含まれる `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/OSXWS`」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。

[60] http://developer.apple.com/library/mac/documentation/DeveloperTools/Conceptual/PackageMakerUserGuide/Introduction/Introductio
[61] https://developer.apple.com/downloads/

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[62] 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= を適宜書き換えた上で、「OSX-WS/OSXWS」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/OSXWS」以下にインストールされるファイル類が準備されます。

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

[62] MacOSX-WS-10.10.1.tar.bz2

OSXWS 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 「Distribution」の「Requirements」タグで全て行います。

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 のヴァージョンが適切か？
- 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？
- apt をチェックして OSXWS が既にインストールされているか？
- gcc 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。

注意！

“Pass if” の “false” は機能しません！！ 。

“Pass if” で “true” にして hoge.pmdoc/index.xml を編集して operator=”eq” を operator=”ne” にして対処します。

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。

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、左側カラムの「Distribution」が選択されたウィンドが現れます。
右側カラムの「Configuration」タグを選択して必要事項を記述します。

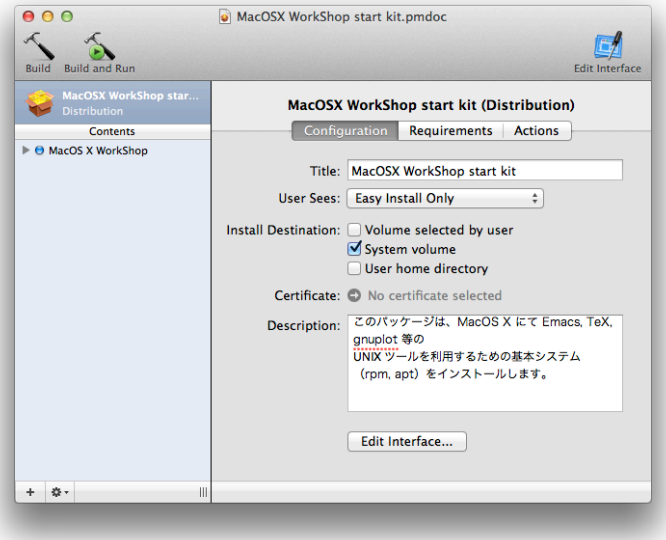

図 1: 「PackageMaker-Distribution-Configuratio」

● 「Requirements」タグではインストールに必要な条件のチェックを指示します。

新たな項目は左下の「＋」を押して作ります。既存の項目の編集は、項目をダブルクリックします。

● 次に、左側カラムの「Contents」にある項目を選択して、右側カラムの「Configuration」タグ以下を記述します。

● 左側カラムの「Contents」にある項目を開いて、右側カラムの「Configuration」タグ以下を記述します。

● 右側カラムの「Contents」タグ以下でファイルやディレクトリのオーナーやパーミッションをチェックします。

● 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。

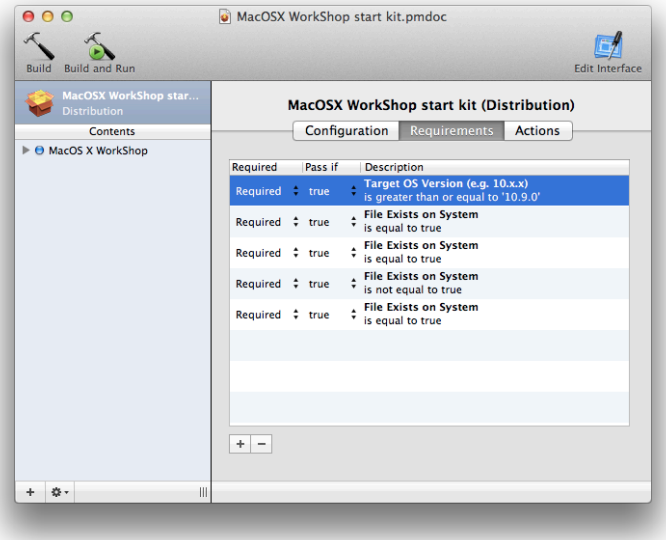

図 2: 「PackageMaker-Distribution-Requirements」

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 8MB 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-10.10.1.tar.bz2**[63] の中にある make-pkg.sh をご覧ください。

[63] MacOSX-WS-10.10.1.tar.bz2

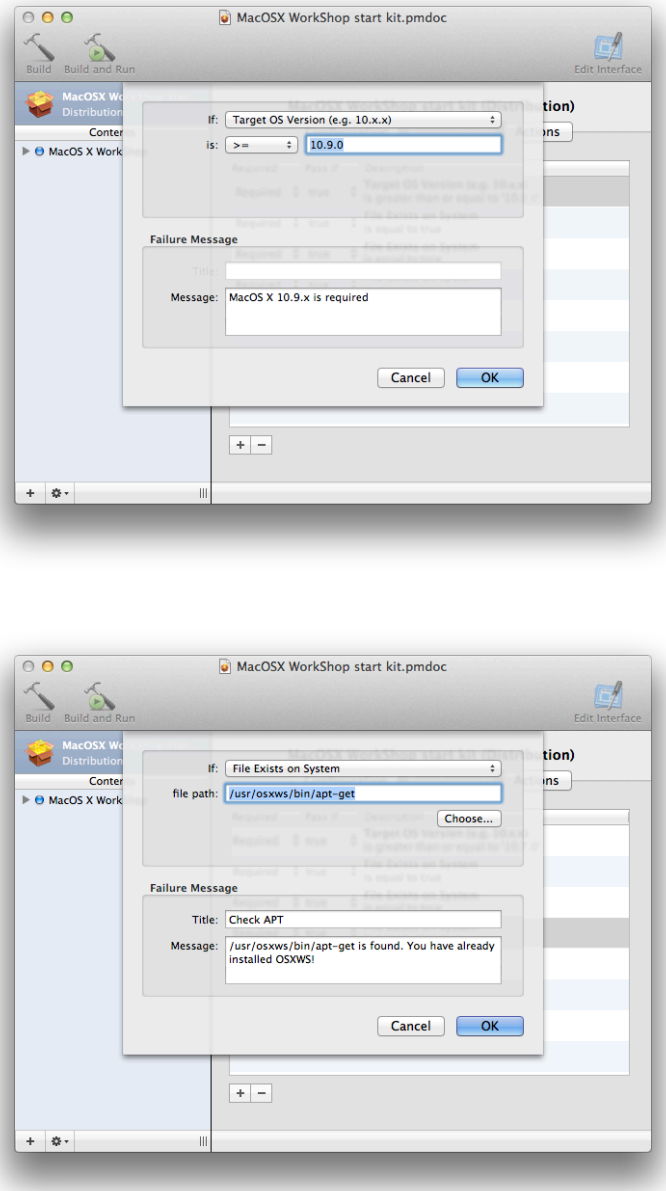

図 3: 「PackageMaker-Distribution-Requirements Describe」

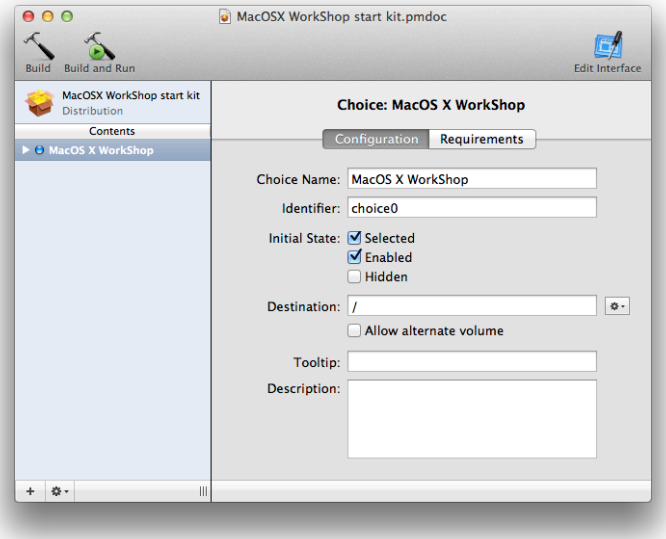

図 4: 「PackageMaker-Contents-Configuration」

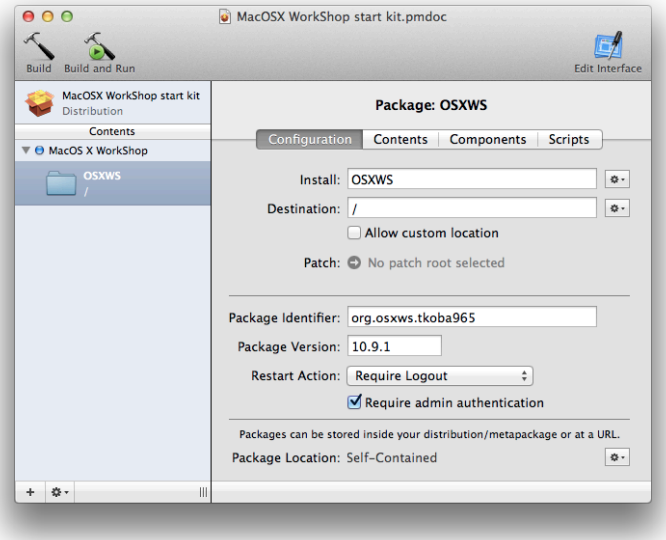

図 5: 「PackageMaker-Package-Configuration」

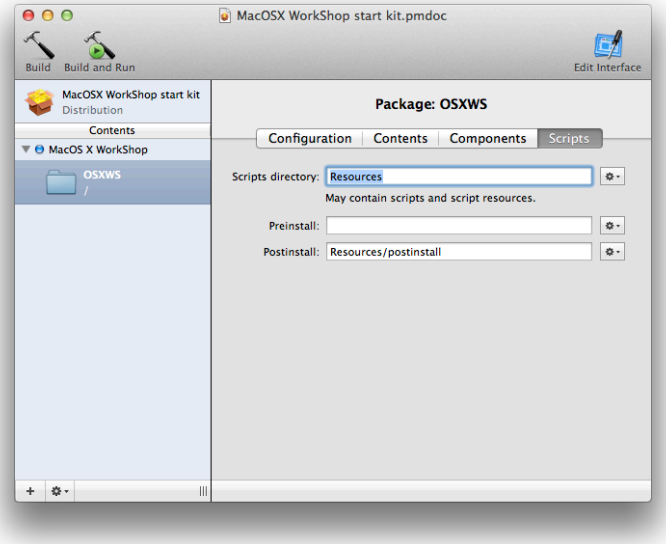

図 6: 「PackageMaker-Package-Scripts」

8　apt-rpm tree を作る

OSXWS tree を用意した際の備忘録です。

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 GB のファイルを置く容量を確保出来るか？
- サーバ／ネットワークの性能は十分か？

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

OSXWS は現時点では 10.3 から 10.10 版までがあり、
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/{Syrah,Mavericks,MountainLion,Lion,SnowLeopard,Leopard,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/バージョン/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、minor release 迄の小さな更新用に updates カテゴリーをもうけています。
- バイナリー・パッケージ
  バイナリー・パッケージ（*.{fat,noarch}.rpm）は「OSXWS/バージョン/fat/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。
OSXWS の場合は

```
[Yosemite] genbasedir --progress --bz2only [OSXWS]/Syrah/fat main updates
```

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-sourceslist-yourDistr パッケージを作成します。

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

して、apt のソースパッケージを展開します。
次に、~/rpm/SOPURCES/sources.list-yosemite-yourDist を作り、貴方が作った apt-line を記述します。

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-sourceslist-yourDistr-osx.spec
```

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

9 Maverics 版からの変更点

Yosemite 版は Mavericsk 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。

- TEXEmacs 関連
  - Emacs 24.5
- OpenFOAM-2.2.2, 2.3.1 (alternatives で切り替えて利用)
- XQuartz-2.7.7 以降

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 (Mavericks)**[64] をご利用ください。

10.1　Emacs 関連

OSXWS では、今のところ CocoaEmacs のみを提供しています。
CocoaEmacs に関する情報は **MacEmacs**[65] をご覧ください。
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 (Cocoa 版)

emacs-lisps 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 をもとに纏めたものです。

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。

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

[64] MountainLion/rpm2html/
[65] http://macemacsjp.sourceforge.jp/

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

10.2 TeX 関連

パッケージの内容は、山本さんが主にメンテナンスしている Vine Linux の TeX Live package 群と基本的に同一です。

pTeXLive 本体 UTF8 で作成しています。
texlive-20140524 base です。

TEXmacros texlive-macros TEX 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, jlisting

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 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。

OTF-Morisawa-RmSgSmg6 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-RyuminPr6-{Regular,Heavy}.otf,
A-OTF-ShinGoPr6-{Regular,Heavy}.otf,
A-OTF-ShinMGoPr6-{Regular,Bold}.otf
を利用する為の設定パッケージです。

OTF-Morisawa-RmSgSmg6N 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-RyuminPr6N-{Regular,Heavy}.otf,
A-OTF-ShinGoPr6N-{Regular,Heavy}.otf,
A-OTF-ShinMGoPr6N-{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」[66] をインストールします。

その他 **BibDesk BibDesk**[67]
文献管理ソフトです。

LaTeXiT LaTeXiT[68]
Apple の Keynote 等で数式を扱う際に利用出来ます。

TeXShop TeXShop[69]
TeX の統合環境です。

task-texlive 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。

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、その場合は

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

等と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

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.

[66] http://sourceforge.jp/projects/efont/files/
[67] http://bibdesk.sourceforge.net
[68] http://www.chachatelier.fr/latexit/latexit-home.php?lang=en
[69] http://pages.uoregon.edu/koch/texshop/

openMotif The Open Motif runtime components.

ttfonts-ja Free Japanese TrueType fonts

内容はさざなみフォント [70] です。

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

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

yaplot yaplot - an easy 3D modeller and animator
GTK2 上で動きます。

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 gcc-4.8.2 A GNU Compiler Collection.
gfortran を提供します。

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。

`/usr/osxws/etc/{profile-osxws,csh.login-osxws,zprofile}` が加えられ `/usr/osxws/bin, /usr/X11/bin` 等にパスを通します。

OSXWS 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。

OSXWS 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 (.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`
の中に記述して下さい。

[70] http://sourceforge.jp/projects/efont/files/

`/usr/osxws/share/OSXWS/jp/` 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。

OSXWS 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。

OSXWS 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。

11　スクリーンショット

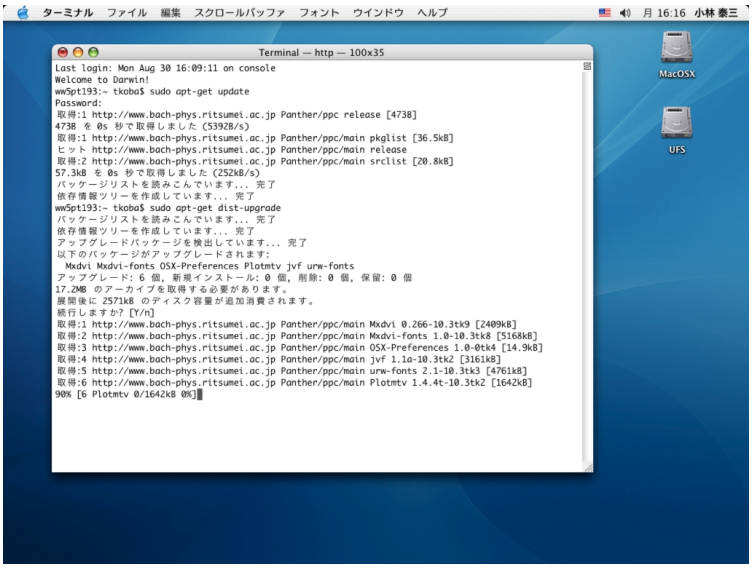

図 7: apt-get でアップデートパッケージをダウンロード中。

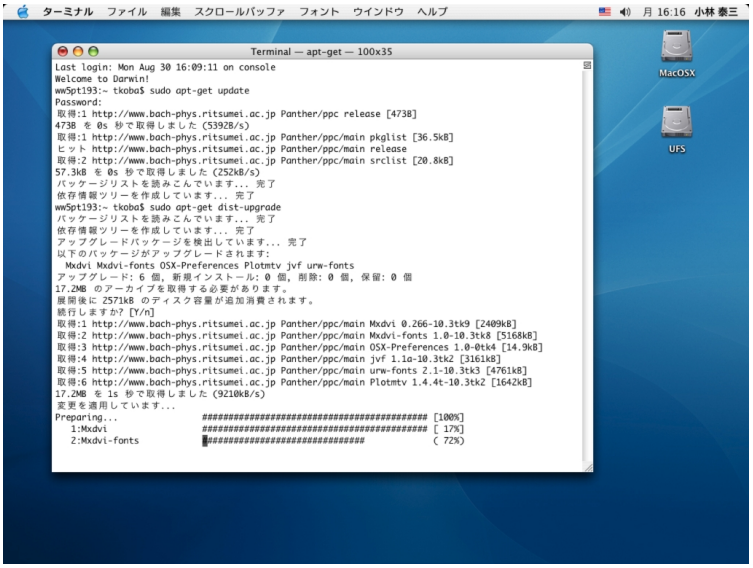

図 8: apt-get でパッケージを更新中。

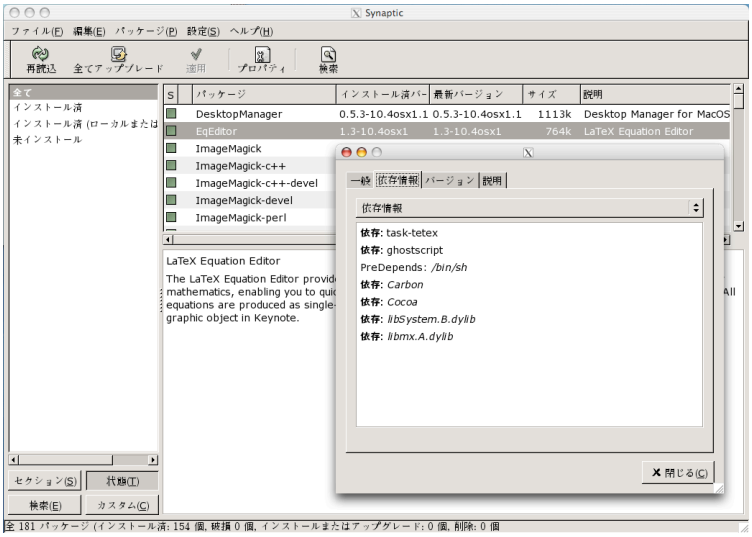

図 9: X11 上の apt-rpm frontend である synaptic

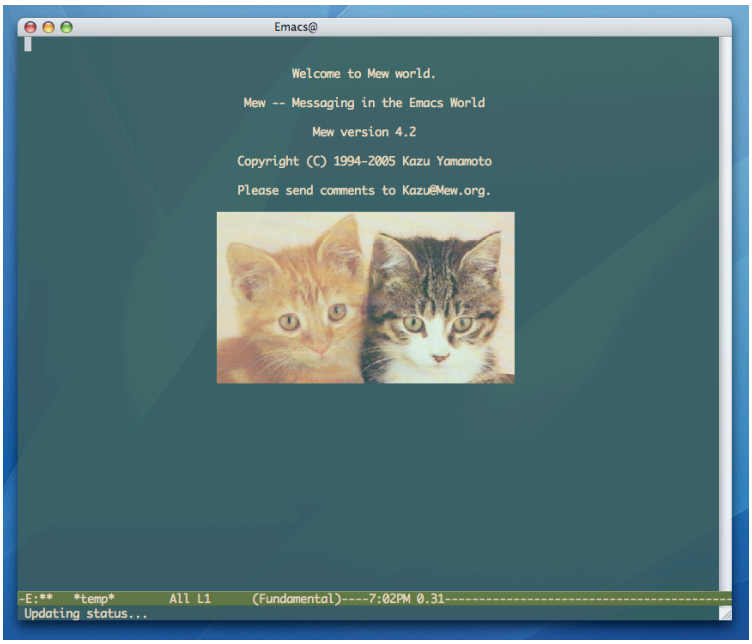

図 10: Emacs で mew を立ち上げているところ。

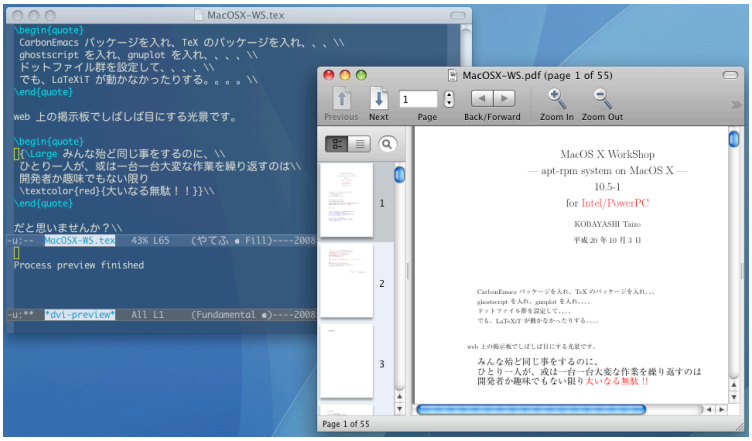

図 11: Emacs で yatex を用いて LaTeX の文章を書き Skim でプレビューしているところ。Synctex に対応しています。

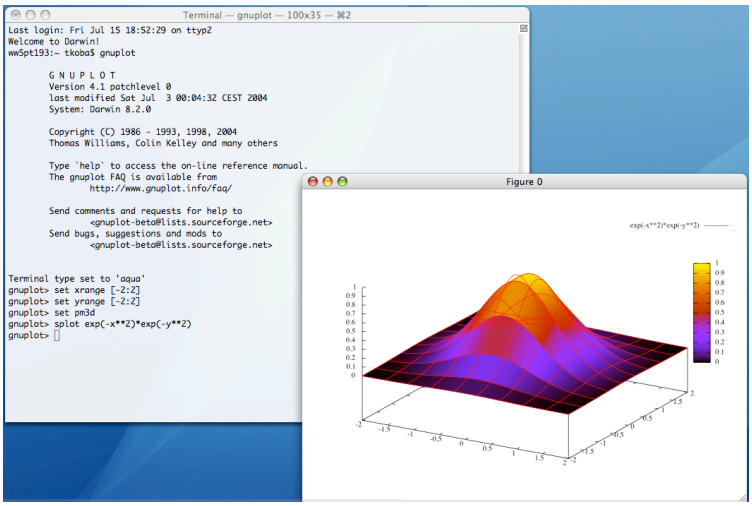

図 12: お馴染み gnuplot です。aquaterm, PDFlib-Lite も同時にインストールされます。

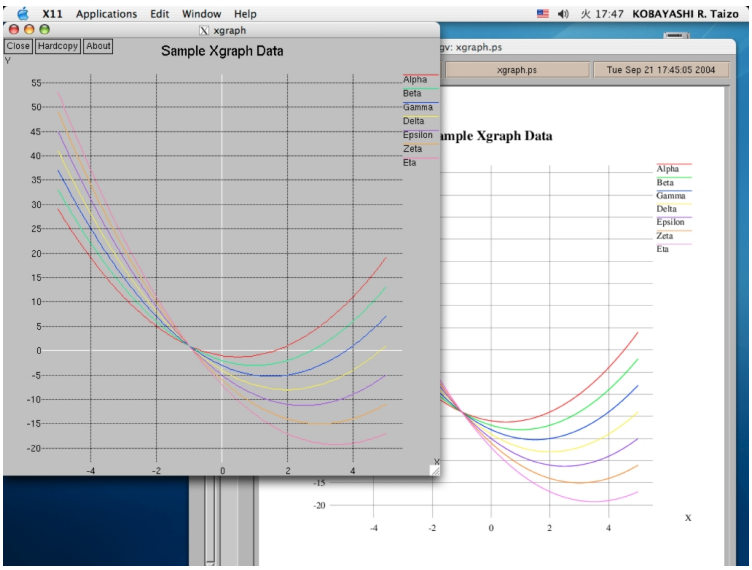

図 13: Vine Linux と同じパッチを適用してあります。Color PS を生成できます。PS ファイルをダブルクリックすれば大抵の場合 PDF に変換でき、プレビューで確認と印刷ができます。

12　既知の問題点と注意点

2016 年 2 月 2 日現在、OSXWS には特に不具合は見つかっていません。

etc. その他、把握し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**[71] にて行われた過去の議論の書庫です。

13.1　10.10-1 公開迄

議論と要望

- 小林先生，お返事頂き誠にありがとうございます．まさにご指摘の通り頂きましたように 23 日以前にインストールしてしまったので，emacs が yatex を呼ぶのに失敗していたようです．再度インストールさせて頂きましたらうまく起動致しました –133.42.2.86 2014 年 10 月 30 日 (木) 16:21 (JST)
  - 解決のご連絡をありがとうございます。些細なことでも結構ですのでこれからもご指摘ご提案をよろしくお願い致します。–利用者:Tkoba
- いつも大変お世話になっております．10.10 をクリーンインストールして OSXWS10.9 を導入してみましたが，emacs が yatex を呼ぶことに失敗してしましますね．動作環境は，XQuartz2.7.7，Xcode6.1（command tool は別途インストール）を使用しております．どうぞ宜しくお願い致します．–122.130.230.90 2014 年 10 月 25 日 (土) 22:31 (JST)
  - ご連絡をありがとうございます。この１０月２３日に emacs を 24.4 に更新したのに合わせて emacs lisps も整理を行いました。その狭間に引っかかったのかもしれません。パッケージの更新をかけて戴いてご確認下さい。まだエラーがでるようでしたら emacs の Messages buffer のログをご提供下さい。よろしくお願い致します。–利用者:Tkoba
- お忙しい中開発ありがとうございます．OSXWS 10.9 の TeX と emacs 周りはとりあえず 10.10 でも動くようです．たくさん使ったわけではありませんが．–118.237.27.3 2014 年 10 月 25 日 (土) 16:05 (JST)

13.2　dot.emacs 10.10-1 公開迄

OSXWS-10.10-1 の変更点

OSXWS-10.9 からの変更はありません。

OSXWS-10.10-1 の emacs 設定ファイル

- ユーザーの初期設定ファイルを読む直前に osxws.el がロードされる：”/usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws.el”

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  osxws.el for MacOS X WorkShop
;;            KOBAYASHI Taizo <xxxxxxx@xxxxxxx>
```

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

```
;; Time-stamp:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

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

(setq emacs-build-system
      (concat
       emacs-build-system
       " - MacOS X WorkShop - 10.10 "))

(setq report-emacs-bug-address "osxws@xxxxxxxxxxx")

(defcustom osxws-default t
  "A boolean for all OSX Workshop default settings"
  :type 'boolean)

(defcustom osxws-default-base t
  "A boolean for loading osxws-setting section 0 (fundamental configurations)"
  :type 'boolean)

(defcustom osxws-default-language-auto t
  "A boolean for loading osxws-setting section 1 (language auto detect)"
  :type 'boolean)

(defcustom osxws-default-appearance t
  "A boolean for loading osxws-setting section 2 (appearance)"
  :type 'boolean)

(defcustom osxws-default-keyboard t
  "A boolean for loading osxws-setting section 3 (keyboard/keybinding)"
  :type 'boolean)

(defcustom osxws-default-shell t
  "A boolean for loading osxws-setting section 4 (shell-command)"
  :type 'boolean)

(defcustom osxws-default-inlinepatch t
  "A boolean for loading osxws-setting section 5 (inline patch)"
  :type 'boolean)

(defcustom osxws-default-cocoaemacs t
```

```
  "A boolean for loading osxws-setting section 6 (Cocoa Emacs)"
  :type 'boolean)

(defcustom osxws-default-else t
  "A boolean for loading osxws-setting section 7 (anything else)"
  :type 'boolean)

(defcustom osxws-default-mew t
  "A boolean for loading osxws-setting for Mew"
  :type 'boolean)

(defcustom osxws-default-yatex t
  "A boolean for loading osxws-setting for YaTeX"
  :type 'boolean)

(if (file-exists-p (concat user-emacs-directory "setup_osxws_default.el"))
    (load-file (concat user-emacs-directory "setup_osxws_default.el")))

(when osxws-default
    (message "Starting osxws-default ...")
    (load-file "/usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-default.el")
    (if osxws-default-yatex
        (load-file "/usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-default-yatex.el")
    )
  )

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

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

- デフォルト設定を使用しない場合、ホームディレクトリのファイルから各変数を nil にする:" /.emacs.d/setup_osxws_defa

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; ~/.emacs.d/setup_osxws_default.el
;;  .emacs for MacOS X WorkShop
;; Time-stamp:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; MacOS X WorkShop provides the default setting for Emacs.
;; The setting is written in the following file:
```

```
;; /usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-default.el
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; You can inactivate all by setting the variable osxws-defaul to nil.
;; (setq osxws-default nil)
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; You can inactivate each section of the setting by setting
;; the variables osxws-defaul-* to nil.
;;(setq osxws-default-base nil)
;;(setq osxws-default-language-auto nil)
;;(setq osxws-default-appearance nil)
;;(setq osxws-default-keyboard nil)
;;(setq osxws-default-shell nil)
;;(setq osxws-default-inlinepatch nil)
;;(setq osxws-default-cocoaemacs  nil)
;;(setq osxws-default-else nil)
;;(setq osxws-default-mew nil)
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; You can inactivate the default setting for YaTeX by
;;(setq osxws-default-yatex nil)
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Local Variables:
;; mode: emacs-lisp
;; buffer-file-coding-system: utf-8-unix
;; End:
```

- init.el の例を兼ねて、最も変更する可能性の高いウィンドウ設定のみ記述：” /.emacs.d/init.el”

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; ~/.emacs.d/init.el
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; You can edit this file as you like!
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;; no start up message
;;(setq inhibit-startup-screen t)

;; window mode setting
(when (eq window-system 'ns)
  ;; window size
  ;;(setq default-frame-alist
  ;;   (append
  ;;     '((width . 100) (height . 40))
  ;;      default-frame-alist))
  ;; Color-thema
  (require 'color-theme)
  (color-theme-dark-blue2)

  ;; Transparency3
  (add-to-list
   'default-frame-alist
   '(alpha . (100 80))) ;; (alpha . (<active frame> <non active frame>))

  ;; Save GUI Emacs frame size and position.
  ;; Restore the size and position when you launch Emacs.
  (require 'save-frame-posize)
  )

;; input special and control characters by "Option"
(setq ns-option-modifier 'none)

;; cursor mode
(setq blink-cursor-interval 0.5)
(setq blink-cursor-delay 5.0)
(blink-cursor-mode 1)   <--- 点滅を止める時は -1 に設定

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Local Variables:
;; mode: emacs-lisp
;; buffer-file-coding-system: utf-8-unix
;; End:
```

- デフォルト設定のファイル：”/usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-default.el”

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; osxws-default.el for MacOS X WorkShop
;; Time-stamp:
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

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

(when osxws-default-base
  ;;;; path setting
  (setq exec-path
(append
 (list "/usr/osxws/bin" "~/bin") exec-path))
  (setenv "PATH"
  (concat '"/usr/osxws/bin:~/bin:" (getenv "PATH")))

  ;;; save the position before you editing.
  (require 'saveplace)
  (setq-default save-place t)
  (setq save-place-file "~/Library/Application Support/OSXWS/emacs-places.txt")

  ;;; copy foo to foo~ as a backup file
  (setq backup-by-copying t)

  ;;; yank text with mouse
  (setq mouse-drag-copy-region t)

  ;;; deleting files goes to OS's trash folder
  ;;(setq delete-by-moving-to-trash t)
  ;;(setq trash-directory "~/.Trash")

  ;;; start emacsclient server if in window mode
  (if window-system
      (progn
        (require 'server)
        (unless (server-running-p) (server-start))))
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; Section 1 language configurations (auto detect)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq appleLang
      (shell-command-to-string
        "/bin/echo -n `/usr/bin/defaults read -g AppleLocale | cut -f 1 -d _`"))
```

```
(when osxws-default-language-auto
  ;;; japanese settings for Cocoa Emacs
  (cond ((string-match appleLang "ja") (set-language-environment 'Japanese))
        ((string-match appleLang "en") (set-language-environment 'English))
        ((string-match appleLang "fr") (set-language-environment 'French))
        ((string-match appleLang "de") (set-language-environment 'German))
        ((string-match appleLang "es") (set-language-environment 'Spanish))
        ((string-match appleLang "it") (set-language-environment 'Italian))
        ((string-match appleLang "nl") (set-language-environment 'Dutch))
        ((string-match appleLang "sv") (set-language-environment 'Swedish))
        (t (set-language-environment 'English)))
  (prefer-coding-system  'utf-8-unix)
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; Section 2 appearance setting
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(when osxws-default-appearance
  ;;; hide tool-bar and menu-bar
  (if window-system
      (tool-bar-mode 0)
    (menu-bar-mode 0))

  ;;; show the corresponding paren
  (show-paren-mode)

  ;;; do not font scaling
  (setq scalable-fonts-allowed nil)

  ;;; show the present time on status bar
  (when (equal current-language-environment "Japanese")
    (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)
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; Section 3 keyboard/keybinding
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(when osxws-default-keyboard
  ;;; emulation of the standard CUA key bindings (Mac GUI)
  (cua-selection-mode t)

  ;;; behavior of "Command + Cursor" to the default of MacOS X
  ;;; default : ns-next-frame in ns-win.el
  (define-key global-map [s-left] 'move-beginning-of-line)
  ;;; default : ns-prev-frame in ns-win.el
  (define-key global-map [s-right] 'move-end-of-line)
  (define-key global-map [s-up] 'backward-page)
  (define-key global-map [s-down] 'forward-page)

  ;;; font resize short cut (Command +/-/0)
  (global-set-key [(s ?+)] (lambda () (interactive) (text-scale-increase 1)))
  (global-set-key [(s ?-)] (lambda () (interactive) (text-scale-decrease 1)))
  (global-set-key [(s ?0)] (lambda () (interactive) (text-scale-increase 0)))

  ;;; revert [Home] Key and [End] Key
  (define-key global-map [home] 'beginning-of-buffer)
  (define-key global-map [end] 'end-of-buffer)

  ;;; Delete the following character by fn + delete
  (define-key global-map [kp-delete] 'delete-char)

  ;;; 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 4 shell-command
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(when osxws-default-shell
  ;;; hide password
  (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 5 inline-patch by Hashimoto-san
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(when osxws-default-inlinepatch
  (when window-system
    (setq default-input-method "MacOSX")
    (add-hook 'minibuffer-setup-hook 'mac-change-language-to-us)
    ;;(mac-add-ignore-shortcut '(control))
    (mac-add-key-passed-to-system 'shift)
    (mac-set-input-method-parameter "com.apple.inputmethod.Kotoeri.Roman" `title "
あ")
    (mac-set-input-method-parameter "com.apple.inputmethod.Kotoeri.Roman" `cursor-type 'box)
    (mac-set-input-method-parameter "com.apple.inputmethod.Kotoeri.Japanese" `cursor-color "red

  ;;; start up by Command-Space
  (global-set-key [(s \ )] 'toggle-input-method)
  ;;; start up by Shift-Space
  (global-set-key [?\S-\ ] 'toggle-input-method)
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; Section 6 CocoaEmacs window mode
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(when osxws-default-cocoaemacs
```

```
  (when window-system
    ;;;; Font setting
    (set-frame-font "Inconsolata 16")
    (set-fontset-font (frame-parameter nil 'font)
      'unicode
      '("ヒラギノ丸ゴ ProN" . "unicode-bmp") nil 'append)
    (add-to-list 'default-frame-alist '(font . "Inconsolata 16"))

    ;;;; smart-dnd
    (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 7 anything else
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(when osxws-default-else
  ;;; The number of lines to scroll a window by when point moves out.
  (setq scroll-step 1)

  ;;;; Time Stamp
  ;;;   If you put 'Time-stamp: <>' or 'Time-stamp: ""' on
  ;;;   top 8 lines of the file, the '<>' or '""' are filled with the date
  ;;;   at saving the file.
  (require 'time-stamp)
  (if (not (memq 'time-stamp write-file-functions))
      (setq write-file-functions
    (cons 'time-stamp write-file-functions)))

  ;;;; Auto Complete Mode
  (require 'auto-complete-config)
  (ac-config-default)
  (add-to-list 'ac-dictionary-directories "/usr/osxws/share/emacs/site-lisp/emacs-lisps/ac-dict"

  ;;; text-to-speech by say: sentence
  (defun osxws-speech (b e)
    "Convert text to audible speech by /usr/bin/say of OSX."
    (interactive "r")
    (let (str)
      (if (eq mark-active 'nil)
          (progn (setq b1 (save-excursion (re-search-backward "^\\ *%")
                                          (point)))
                 (setq b2 (save-excursion (re-search-backward "^\n")
                                          (point)))
                 (setq e1 (save-excursion (re-search-forward "^\\ *%")
                                          (point)))
                 (setq e2 (save-excursion (re-search-forward "^\n")
                                          (point)))
                 (setq b (if (< b1 b2) b2 t b1))
                 (setq e (if (> e1 e2) e2 t e1))))
```

```
    (setq str (buffer-substring-no-properties b e))
    (setq str (replace-regexp-in-string "\\\\%" "percent" str))
    (setq str (replace-regexp-in-string "%[^\n]*" "" str))
    (setq str (replace-regexp-in-string "\\\\[a-zA-Z]+{" " " str))
    (setq str (replace-regexp-in-string "[$_^\n()~{}|;‘]" " " str))
    (setq str (replace-regexp-in-string "\\ +" " " str))
    (message str)
    (if (eq (process-status "speech") 'run)
        (delete-process "speech")
      (process-kill-without-query
       (start-process-shell-command "speech" nil
                                    "/usr/bin/say " (concat "\"" str "\"" ))))))

;;; text-to-speech by say: word
(defun osxws-speech-word()
  "Speak words."
  (interactive)
  (let* ((str (url-hexify-string (string-word-or-region))))
    (process-kill-without-query
     (start-process-shell-command "speech" nil
                                  "/usr/bin/say -r 150" (concat "\"" str "\"" )))))

;;; Search marked region by google
(defun osxws-search-google()
  "Search marked region by google"
  (interactive)
  (let* ((str (string-word-or-region)))
    (browse-url
     (concat "http://google.com/search?q=\"" str "\""))))

;;; Search marked region by google scholar
(defun osxws-search-googlescholar()
  "Search string by google scholar"
  (interactive)
  (let* ((str (string-word-or-region)))
    (browse-url
     (concat "http://scholar.google.com/scholar?q=\"" str "\""))))

;;; "Look up the word by Dictionary.app of Mac OS X"
;;; http://sakito.jp/mac/dictionary.html
(defun osxws-lookup-dictionary-osx()
  "Look up the word by Dictionary.app of Mac OS X"
  (interactive)
```

```
  (let* ((str (url-hexify-string (string-word-or-region))))
    (browse-url (concat "dict://" str ))))

(defun string-word-or-region ()
  "If a region is selected, the text string of the region is returned.
Otherwise, the text string of the word is returnd."
  (let ((editable (not buffer-read-only))
        (pt (save-excursion (mouse-set-point last-nonmenu-event)))
        beg end)
    (if (and mark-active
             (<= (region-beginning) pt) (<= pt (region-end)) )
        (setq beg (region-beginning)
              end (region-end))
      (save-excursion
        (goto-char pt)
        (backward-char 1)
        (setq end (progn (forward-word) (point)))
        (setq beg (progn (backward-word) (point)))))
    (buffer-substring-no-properties beg end)))


;;; Show in Finder
(defun osxws-show-in-finder ()
  (interactive)
  (process-kill-without-query
   (start-process-shell-command "Show in Finder" nil "open -R" (buffer-file-name))))

;;; Open current working directory by Finder
(defun osxws-open-folder-in-finder ()
  "open -a Finder.app CURRENT-DIRECTORY"
  (interactive)
  (process-query-on-exit-flag
   (start-process-shell-command "open folder in Finder" nil "open .")))

;;; Open current working directory by Terminal/iTerm
(defun osxws-open-Terminal()
  "open -a Terminal.app CURRENT-DIRECTORY"
  (interactive)
  (let* (;;(cmd "open -a Terminal.app")
         (cmd "open -a iTerm.app"))
    (process-kill-without-query
     (start-process-shell-command
      "Open directory" nil cmd default-directory))))
```

```
  ;;; keybindings
  (define-key global-map [?\s-r] 'osxws-speech)
  ;; keybinding: ^^e2^^8c^^98-option-r
  (define-key global-map [?\s-^^c2^^ae] 'osxws-speech-word)
  ;; keybinding: ^^e2^^8c^^98-option-shift-F
  (define-key global-map [?\s-^^c3^^8f] 'osxws-open-folder-in-finder)
  ;; keybinding: ^^e2^^8c^^98-option-shift-T
  (define-key global-map [?\s-^^cb^^87] 'osxws-open-Terminal)

  (define-key global-map (kbd "C-;") 'ispell-word)
  (define-key global-map (kbd "C-c w") 'osxws-lookup-dictionary-osx)
  (define-key global-map (kbd "C-c g") 'osxws-search-google)
  (define-key global-map (kbd "C-c G") 'osxws-search-googlescholar)
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; Mew 6.6  -  Messaging in the Emacs World
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(when osxws-default-mew
  (autoload 'mew "mew" nil t)
  (autoload 'mew-send "mew" nil t)
  (if (file-exists-p "~/.emacs.d/mew.el")
        (load-file "~/.emacs.d/mew.el"))
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Local Variables:
;; mode: emacs-lisp
;; buffer-file-coding-system: utf-8-unix
;; End:
```

- YaTeX のデフォルト設定：”/usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-default-yatex.el”

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; osxws-default-yatex.el for MacOS X WorkShop
;; Time-stamp:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(autoload 'yatex-mode "yatex" "Yet Another LaTeX mode" t)

;; use \C-c \C- instead of \C-c [yatex:04567]
(unless (boundp 'YaTeX-inhibit-prefix-letter)
```

```
      (setq YaTeX-inhibit-prefix-letter t))

;; use our own scripts for typesetting
(setq YaTeX-typeset-auto-rerun nil)

(setq auto-mode-alist
      (append
'(("\\.\\(tex\\|sty\\|cls\\|fd\\|ind\\|idx\\|ltx\\|clo\\|bbl\\)$" .
   yatex-mode)) auto-mode-alist))

(setq YaTeX-latex-message-code 'utf-8
      YaTeX-use-LaTeX2e t ; AMS-LaTeX
      YaTeX-use-AMS-LaTeX t ; AMS-LaTeX
      YaTeX-use-font-lock t
      YaTeX-skip-default-reader  t)

(if (equal current-language-environment "Japanese")
  (setq YaTeX-kanji-code nil ; (1 JIS, 2 SJIS, 3 EUC, 4 UTF-8)
        tex-command "platex2pdf"
        dvi2-command "open -a Skim"
        makeindex-command "mendex2pdf"
        bibtex-command "pbibtex2pdf"
        dviprint-command-format "dvipdfmx %s -o - | lpr")
  (setq YaTeX-kanji-code nil ; (1 JIS, 2 SJIS, 3 EUC, 4 UTF-8)
        tex-command "latex2pdf"
        dvi2-command "open -a Skim"
        makeindex-command "makeindex2pdf"
        bibtex-command "bibtex2pdf"
        dviprint-command-format "dvipdfmx %s -o - | lpr")
  )


(add-hook 'skk-mode-hook
  (lambda ()
    (if (eq major-mode 'yatex-mode)
      (progn
(define-key skk-j-mode-map "\\" 'self-insert-command)
(define-key skk-j-mode-map "$" 'YaTeX-insert-dollar)
))
))

;;; Indent
;;; http://www.hit.ac.jp/~wachi/misc/latexindent.html
```

```
(autoload 'latex-indent-command "latex-indent"
  "Indent current line accroding to LaTeX block structure.")
(autoload 'latex-indent-region-command "latex-indent"
  "Indent each line in the region according to LaTeX block structure.")

;;;; Skim PDF indicating cursor
;;; pdflatex/platex -synctex=1
;;; switch from Emacs to Skim: C-c s
(defun skim-forward-search ()
  (interactive)
  (process-kill-without-query
    (start-process
      "displayline"
      nil
      "/Applications/OSXWS/Skim.app/Contents/SharedSupport/displayline"
      (number-to-string (save-restriction
                          (widen)
                          (count-lines (point-min) (point))))
      (expand-file-name
        (concat (file-name-sans-extension (or YaTeX-parent-file
                                              (save-excursion
                                                (YaTeX-visit-main t)
                                                buffer-file-name)))
                ".pdf"))
      buffer-file-name)))

;;; insert subscript with roman font
(defun osxws-TeX-insert-subscript_rm ()
  "insert subscript with roman font"
  (interactive)
  (insert "_{\\mathrm{}}")
  (backward-char 2))

;;; switch to previous buffer
(defun osxws-TeX-switch-to-previousbuffer ()
  "switch to previous buffer"
  (interactive)
  (switch-to-buffer nil))

;;; Open BibDesk with the cite key
(defun osxws-TeX-open-item-BibDesk ()
  "Open BibDesk with the cite key"
  (interactive)
```

```
  (let* ((s (save-excursion
              (skip-chars-backward "^{,")
              (point)))
         (e (save-excursion
              (skip-chars-forward "^},")
              (point))))
    (browse-url (concat "x-bdsk://" (buffer-substring-no-properties s e)))))

;;; Open BibDesk
(defun osxws-open-bibdesk ()
  "Open BibDesk"
  (interactive)
  (process-kill-without-query
   (start-process-shell-command "Open BibDesk.app" nil "open -a BibDesk" )))

;;; \cite{foo} -> open foo.pdf
;;; citekey == PDF file name
(defun osxws-TeX-open-article ()
  "Open PDF file by the cite key: \cite{foo} -> open foo.pdf"
  (interactive)
  (let* ((s (save-excursion
              (skip-chars-backward "^{,")
              (point)))
         (e (save-excursion
              (skip-chars-forward "^},")
              (point)))
        ;(pdfdir "~/Documents/Dropbox/Papers/pdf/")
         (pdfdir (interactive "DPDF file directory name: "))
         (pdffile (concat pdfdir
                          (buffer-substring-no-properties s e) ".pdf"))
         (process-kill-without-query
          (start-process-shell-command "Open the article" nil "open" pdffile)))))

;;; Convert text to audible speech by /usr/bin/say (Emacs + OSX)
(defun osxws-TeX-speech ()
  "Convert text to audible speech by /usr/bin/say of OSX."
  (interactive)
  (let (b e s
          (block-sep-str "^\*\\|^\\ *%\\|^\\ *\n\\|\\\\item\\|\\\\begin\\|\\\\end"))
    (if (eq (process-status "speech") 'run)
        (delete-process "speech")
      (progn
        (if mark-active
```

```
            (setq b (mark) e (point))
          (setq b (save-excursion
                    (progn
                      (re-search-backward block-sep-str (point-min) 1)
                      (point)))
                e (save-excursion
                    (progn
                      (re-search-forward block-sep-str (point-max) 1)
                      (point)))))
        (setq s (buffer-substring-no-properties b e)
              s (replace-regexp-in-string "\\\\%" "percent" s)
              s (replace-regexp-in-string "$\\\\mu$m" "micrometer" s)
              s (replace-regexp-in-string "\\\\item" " " s)
              s (replace-regexp-in-string "\\\\begin" " " s)
              s (replace-regexp-in-string "\\\\end" " " s)
              s (replace-regexp-in-string "\\\\," " " s)
              s (replace-regexp-in-string "%[^\n]*" "" s)
              s (replace-regexp-in-string
                 "\\\\[a-zA-Z]+{\\|[$\\\\_^~{}`\"*\n]" " " s)
              s (replace-regexp-in-string "\\ +" " " s))
        (message s)
        (process-kill-without-query
         (start-process-shell-command "speech" nil
                                      "/usr/bin/say" (concat "\"" s "\"" )))))))


;;; YaTeX key bindings
(add-hook 'yatex-mode-hook
          '(lambda ()
             (require 'yatexprc)
             (turn-off-auto-fill) ; no auto fill
             (define-key YaTeX-mode-map [?\s-t]
               (lambda ()
                 (interactive)
                 (YaTeX-typeset-menu nil ?j)))
             (define-key YaTeX-mode-map [?\s-b]
               (lambda ()
                 (interactive)
                 (YaTeX-typeset-menu nil ?j)))
             (define-key YaTeX-mode-map [?\s-P]
               (lambda (arg)
                 (interactive "P")
                 (let ((current-prefix-arg (not arg)))
```

```
      (YaTeX-typeset-menu 'dummy ?p))))
  (define-key YaTeX-mode-map [?\s-B]
    (lambda ()
      (interactive)
      (YaTeX-typeset-menu nil ?b)))
  (define-key YaTeX-mode-map [?\s-I]
    (lambda ()
      (interactive)
      (YaTeX-typeset-menu nil ?i)))
  (define-key YaTeX-mode-map [?\s-J] 'osxws-TeX-open-item-BibDesk)
  (define-key YaTeX-mode-map [?\s-R] 'skim-forward-search)
  (define-key YaTeX-mode-map (kbd "C-c s") 'skim-forward-search)
  (define-key YaTeX-mode-map "\t" 'latex-indent-command)
  (define-key YaTeX-mode-map (kbd "C-c TAB") 'latex-indent-region-command)
  (define-key YaTeX-mode-map [?\s-_] 'osxws-TeX-insert-subscript_rm)
  (define-key YaTeX-mode-map [?\C-\s-J] 'YaTeX-goto-corresponding-*)
  (define-key YaTeX-mode-map [?\s-H] 'YaTeX-display-hierarchy)
  (define-key YaTeX-mode-map [?\s-1] 'YaTeX-visit-main)
  (define-key YaTeX-mode-map [?\s-2] 'osxws-TeX-switch-to-previousbuffer)
  (define-key YaTeX-mode-map [?\M-\s-B] 'osxws-open-bibdesk)
  (define-key YaTeX-mode-map [?\s-^^c4^^b1] 'osxws-open-bibdesk)
  (define-key YaTeX-mode-map [?\s-\)] 'osxws-TeX-speech-region)
  (define-key YaTeX-mode-map [?\s-C] 'osxws-TeX-open-article)
  ))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; yahtml
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq auto-mode-alist
      (cons (cons "\\.html$" 'yahtml-mode) auto-mode-alist))
(autoload 'yahtml-mode "yahtml" "Yet Another HTML mode" t)
(add-to-list 'auto-mode-alist '("\\.htm\\'" . yahtml-mode))

(setq yahtml-www-browser "open"
      yahtml-lint-program "htmllint"
      yahtml-kanji-code 4)

(add-hook 'yahtml-mode-hook
     '(lambda ()
       (auto-fill-mode -1)
       ))
```

```
;;;; <p> </p>
(setq yahtml-always-/p t)
;;;; <li> </li>
(setq yahtml-always-/li t)

;; End:
```

コメント

14 謝辞

OSXWS 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。
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
MacWiki	-
Project PINEAPPLE の皆さん	-
Vine Linux の皆さん	-

更新履歴

- Tue Feb 02 2016 KOBAYASHI Taizo
  - Version 10.10-1
- Tue Dec 02 2014 KOBAYASHI Taizo
  - Version 10.9-1
- Tue Nov 05 2013 KOBAYASHI Taizo
  - Version 10.8-1
- Mon Sep 03 2012 KOBAYASHI Taizo
  - Version 10.7-1
- Wed Aug 17 2011 KOBAYASHI Taizo
  - Version 10.6-1
- Mon Aug 09 2010 KOBAYASHI Taizo
  - Version 10.5-3
  - 「過去の議論」追加記入
- Thu Dec 10 2009 KOBAYASHI Taizo
  - Version 10.5-2
  - 「過去の議論」追加記入
- Wed Oct 01 2008 KOBAYASHI Taizo
  - Version 10.5-1
- 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 [72] 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

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

- 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。
  - 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