MacOS X WorkShop
— an apt-rpm system on MacOS X —
10.6-1
for i386/x86_64

KOBAYASHI Taizo

2011/08/24

CocoaEmacs を入れ、TeXLive 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、LaTeXiT 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。

目 次

図 目 次

1　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、
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。
　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**[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 10.6.x (SnowLeopard) 対応版と **MaxOS X 10.5.x (Leopard) 対応版**[2] 、**MaxOS X 10.4.x (Tiger) 対応版**[3] 、**MaxOS X 10.3.x (Panther) 対応版**[4] です。
尚、今後 SnowLeopard 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 (SnowLeopard)**[5] **RPM 解説データベース (Leopard)**[6] **RPM 解説データベース (Tiger)**[7] **RPM 解説データベース (Panther)**[8] 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 (Section9 参照) 」をご覧ください。

1.1　何故 apt-rpm か？

　個々のソフトウェアを開発する人達を宮大工さんとるすと、
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。
都市を造るのに大工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。

[1] http://www.vinelinux.org/
[2] ../Leopard/index.html
[3] ../Tiger/index.html
[4] ../Panther/index.html
[5] ../SnowLeopard/rpm2html/
[6] ../Leopard/rpm2html/
[7] ../Tiger/rpm2html/
[8] ../Panther/rpm2html/

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、
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 (Section5 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**[9] をはじめ、
琉球大学の **EasyPackage**[10] や、**DarwinPorts**[11] 等があり、

[9] http://fink.sourceforge.net/
[10] http://www.ie.u-ryukyu.ac.jp/darwin2/
[11] http://darwinports.opendarwin.org/

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**[12] 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、
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[13] は **Red Hat**[14] 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[15] 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。

[12] http://macwiki.sourceforge.jp/cgi-bin/wiki.cgi
[13] http://www.rpm.org/
[14] http://www.redhat.com/
[15] http://www.vinelinux.org/

apt[16] は **Debian GNU/Linux**[17] 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。
- 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[18] は **Conectiva Linux**[19] 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 (Section4 参照) 」を御覧下さい。

1.2 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 `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出来るだけ留めます。

[16] http://www.debian.org/doc/manuals/apt-howto/
[17] http://www.jp.debian.org/
[18] https://moin.conectiva.com.br/AptRpm
[19] https://moin.conectiva.com.br/

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。

1.3 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。

インストーラの作り方は「インストーラを作る (Section6 参照) 」を、
apt-rpm tree の作り方は「apt-rpm tree を作る (Section7 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 !! (Section2 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。

1.4 連絡先とメンバー

MacOS X WorkShop に関する議論や連絡は **Mac Wiki**[20] 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

打田旭宏 立命館大学物理学教室池田研究室 PD

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

瀬戸亮平 ミュンヘン工科大学 PD

山本宗宏 Vine Linux packager

新山友暁 立命館大学物理学教室池田研究室 PD

1.5 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/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。

2　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 (/private/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**[21]
  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**[22]
  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。

[21] http://www-jlc.kek.jp/ fujiik/macosx/10.6.X/HEPonX/
[22] http://www.bach-phys.ritsumei.ac.jp/OSXWS/

3 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、以前の MacOS X に関する情報は **MacOS X 10.5 (Leopard)**[23] **MacOS X 10.4 (Tiger)**[24] **MacOS X 10.3 (Panther)**[25] をご覧ください。

3.1 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 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。

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/X11/bin/Xquartz` がインストールされていれば大丈夫です。

- `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。

3.2 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「環境設定...」を選択し、上部にあるアイコンの『設定』を選択します。

3. 右上に並んだ項目から「詳細」と表示されている選択項目をプレスします。

4. 「非 ASCII 文字をエスケープする」のチェックを外します。

[23] ../Leopard/index.html
[24] ../Tiger/index.html
[25] ../Panther/index.html
[26] http://developer.apple.com/tools/xcode/

3.3 Install

MacOS X WorkShop を始めるには
MacOS X WorkShop start kit MacOSX-WS-10.6.1.dmg[27]
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-10.6.1.tar.bz2**[28] 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 (Section4 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**[29] 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。

[27] MacOSX-WS-10.6.1.dmg
[28] MacOSX-WS-10.6.1.tar.bz2
[29] OSX-system/

- `.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。

- `.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-10.6.tar.bz2**[30] 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。

[30] OSX-Preferences-10.6.tar.bz2

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。

3.4 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[31] をマウントします。

```
$ hdid MacOSX-WS-10.6.1.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-10.6.1/MacOSX\ WorkShop\ start\ kit.pkg -tar
$ hdiutil eject /Volumes/MacOSX-WS-10.6.1
```

3. **再起動**
   再起動します。

```
$ sudo reboot
```

3.5 SnowLeopard 以前の版からの 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。

3.6 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/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。

[31] MacOSX-WS-10.6.1.dmg

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　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 と /usr/osxws/etc/apt/apt.conf 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。
CocoaEmacs 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
> eplatex 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。

4.1 apt の「いろは」

Tiger 版には apt の GUI frontend である **Synaptic**[32] 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 [33] 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。

4.1.1 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 [34] 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

[32] http://www.nongnu.org/synaptic/
[33] file:///usr/osxws/share/synaptic/html/index.html
[34] 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。

4.1.2 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。

4.1.3 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
1752kB のアーカイブを取得する必要があります。
展開後に 4825kB 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 SnowLeopard/fat/main psutils 1.17-11osx10.6 [138kB]
取得:2 http://www.bach-phys.ritsumei.ac.jp SnowLeopard/fat/main a2ps 4.13b-12osx10.6 [1614kB]
1752kB を 0s 秒で取得しました (2669kB/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-1.17-11osx10.6.fat             ############################## [100%]
  a2ps-4.13b-12osx10.6.fat               ############################## [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。

4.1.4 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 4825kB 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
  a2ps-4.13b-12osx10.6.fat               ############################## [100%]
```

```
  psutils-1.17-11osx10.6.fat              ############################## [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。

4.1.5 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
```

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。

4.1.6 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
```

4.2 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 (Section5) 」をご覧ください。

4.2.1 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     : 4.13b
Release     : 12osx10.6
Architecture: fat
Install Date: 火  8/16 18:14:28 2011
Group       : Applications/Publishing
Size        : 4398628
License     : GPL
Signature   : DSA/SHA1, 木  4/21 12:31:44 2011, Key ID f367e1515c69cada
Source RPM  : a2ps-4.13b-12osx10.6.src.rpm
Build Date  : 火  6/ 8 18:13:28 2010
Build Host  : xxxxxxx.cc.kyushu-u.ac.jp
Relocations : (not relocatable)
Packager    : KOBAYASHI Taizo <tkoba965@mac.com>
Vendor      : MacOS X WorkShop
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
/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
```

4.2.2 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-1.23-1osx10.6.ppc.rpm をインストールするには

```
$ sudo rpm -ivh hoge-1.23-1osx10.6.ppc.rpm
```

或は

```
$ sudo rpm -Uvh hoge-1.23-1osx10.6.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。

4.2.3　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。

4.3　OpenFOAM の使い方

　MacOS X WorkShop では、複数のバージョンの OpenFOAM を提供しています。
alternatives に対応してありますので、
複数のバージョンを同時にインストールしておき、
適宜切り替えて利用できます。

　OpenFOAM に関する情報は **OpenFOAM**[35] をご覧ください。

4.3.1　install OpenFOAM-1.7.1

　OpenFOAM-1.7.1 をインストールするには

```
$ sudo apt-get update
$ sudo apt-get install OpenFOAM171
$ sudo apt-get clean
```

を実行してください。
ドキュメントやチュートリアルをインストールするには

[35] http://www.openfoam.com/

```
$ sudo apt-get update
$ sudo apt-get install OpenFOAM171-doc
$ sudo apt-get install OpenFOAM171-tutorials
$ sudo apt-get install OpenFOAM171-source
$ sudo apt-get clean
```

などを適宜実行してください。
再ログインすれば OpenFOAM-1.7.1 の利用が可能になります。

4.3.2 install OpenFOAM-2.0.0

OpenFOAM-2.0.0 をインストールするには

```
$ sudo apt-get update
$ sudo apt-get install OpenFOAM200
$ sudo apt-get clean
```

を実行してください。
後は OpenFOAM-1.7.1 のインストールと同様です。
OpenFOAM-1.7.1 がインストールされている場合は OpenFOAM-2.0.0 が優先されます。

4.3.3 version switching of OpenFOAM

複数のバージョンの OpenFOAM を切り替えるには alternatives を利用します。
インストールされている OpenFOAM を確認するには以下を実行します。

```
$ update-alternatives --display OpenFOAM
OpenFOAM - status is auto.
 link currently points to /usr/osxws/OpenFOAM/OpenFOAM-2.0.0/etc/bashrc
/usr/osxws/OpenFOAM/OpenFOAM-1.7.1/etc/bashrc - priority 20
 slave OpenFOAM-csh: /usr/osxws/OpenFOAM/OpenFOAM-1.7.1/etc/cshrc
/usr/osxws/OpenFOAM/OpenFOAM-2.0.0/etc/bashrc - priority 30
 slave OpenFOAM-csh: /usr/osxws/OpenFOAM/OpenFOAM-2.0.0/etc/cshrc
Current ‘best’ version is /usr/osxws/OpenFOAM/OpenFOAM-2.0.0/etc/bashrc.
```

これは OpenFOAM-1.7.1 と 2.0.0 がインストールされていて、auto に設定されているので 2.0.0 が選択されていることを示しています。
この状況で blockMesh の位置を確認すると以下になります。

```
$ which blockMesh
/usr/osxws/OpenFOAM/OpenFOAM-2.0.0/platforms/darwinIntelDPOpt/bin/blockMesh
```

OpenFOAM のバージョンを切り替えるには以下を実行します。ここでは 1.7.1 への変更を例にします。

```
$ sudo update-alternatives --config OpenFOAM
Password:

There are 2 alternatives which provide ‘OpenFOAM’.

  Selection    Alternative
-----------------------------------------------
      1        /usr/osxws/OpenFOAM/OpenFOAM-1.7.1/etc/bashrc
*+    2        /usr/osxws/OpenFOAM/OpenFOAM-2.0.0/etc/bashrc

Press enter to keep the default[*], or type selection number: 1
Using ‘/usr/osxws/OpenFOAM/OpenFOAM-1.7.1/etc/bashrc’ to provide ‘OpenFOAM’.
$ <re-login>
$ which blockMesh
/usr/osxws/OpenFOAM/OpenFOAM-1.7.1/applications/bin/darwinIntel64DPOpt/blockMesh
```

切り替えた後に再ログインが必要であることに注意してください。

5 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 MacOS X WorkShop 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 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**[36] や、
Momonga Linux の **Specfile-Guidance**[37] 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 (Section9 参照) を参照して下さい.
尚、

```
$ rpm -i hoge-1.0-1osx10.6.src.rpm
```

とすると、
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
```

[36] http://www.vinelinux.org/docs/vine6/making-rpm/vine-making-rpm.html
[37] 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。

5.1 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。
MacOS X WorkShop 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。

5.2 spec file

ここでは MacOS X WorkShop 固有の spec file に関する方針を述べます。
spec file は MacOS X WorkShop 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。
MacOS X WorkShop では VineLinux に倣い

SnowLeopard (release number)osx10.6

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。

尚、noarch 以外の殆どパッケージは fat になっています。

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。

5.3 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/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 MacOS X WorkShop 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`

__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。

```
%_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.)
```

5.4 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。
MacWiki[38] にも UniversalBinary の項目がありますので、
合わせてご覧ください。

はじめに

原則として i386, x86_64 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。
これは、特に library 類を作成する際に i386 と x86_64 ではヘッダファイルを別にするもの（例えば libtiff）が有るからです。

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

spec file の基本形

`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。

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

mv %{name}-%{version} INTEL
cp -a INTEL X86_64
```

%build section

以下の様にして i386, x86_64 それぞれ独立にビルドします。

```
pushd INTEL
CFLAGS="-O3 -arch i386 -mtune=pentium-m" \
CXXFLAGS="$CFLAGS" \
%configure --disable-dependency-tracking \
--host=%{_rpm_platform32} \
--build=%{_rpm_platform32} \
--target=%{_rpm_platform32}

make
popd
pushd X86_64
CFLAGS="-O3 -arch x86_64 -mtune=core2" \
CXXFLAGS="$CFLAGS" \
%configure --disable-dependency-tracking \
--host=%{_rpm_platform64} \
--build=%{_rpm_platform64} \
--target=%{_rpm_platform64}

make
popd
```

%install section

以下の様にして Universal Binary を作成します。

```
pushd INTEL
mkdir -p ${PWD}-root%{_bindir}
make install DESTDIR=${PWD}-root
popd
pushd X86_64
mkdir -p ${PWD}-root%{_bindir}
make install DESTDIR=${PWD}-root
cp -fRP COPYRIGHT README VERSION TODO html ..
popd

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

for i in $filelist
do
/usr/bin/lipo -create INTEL-root/$i X86_64-root/$i -output `basename $i` && \
(cp -f `basename $i` INTEL-root/$i) || :
done

# check header files
for i in `find INTEL-root -name "*.h" -type f`
do
        TARGET=`echo $i | sed -e "s,.*INTEL-root,,"`
        TEMP=`diff -u INTEL-root/$TARGET X86_64-root/$TARGET > /dev/null || echo different`
        if [ -n "$TEMP" ]; then
                mv X86_64-root/$TARGET INTEL-root/${TARGET%.*}-x86_64.h
                mv INTEL-root/$TARGET INTEL-root/${TARGET%.*}-i386.h
FILE=${TARGET##*/}
FILE=${FILE%.*}
cat <<EOF > INTEL-root/$TARGET
#if defined (__i386__)
#include "${FILE}-i386.h"
#elif defined( __x86_64__ )
#include "${FILE}-x86_64.h"
#endif
EOF
        fi
done

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

5.5 その他

ライブラリに関する問題

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 10.6.8 + Xcode 3.2.3 では、

`libtool`	2.2.4 (glibtool, glibtoolize)
`aclocal, automake`	1.10
`autoheader, autoconf`	2.61

が用意されています。
MacOS X WorkShop では、automake1.4, autoconf-2.{13,65} を用意しています。
これらは必要に応じて、`aclocal-1.4 -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。

libpng

/usr/X11/lib/ 以下に在る。 ただし、/usr/X11/lib/pkgconfig/libpng{12}.pc 内で共に Cflags: -I${includedir}/libpng12 を指していながら、実際には/usr/X11/include/libpng しかない！

以下の設定が /usr/osxws/etc/profile-osxws でなされます。

PKG_CONFIG_PATH=”/usr/osxws/lib/pkgconfig:/usr/osxws/share/pkgconfig:/usr/lib/pkgconfig:/usr/X11/lib/p

X11

xmkmf と imake が廃止されています。
Xaw3d 依存の gv などに影響しますが、これを機に X11 apps は Obsoletes 扱いにします。

6　インストーラを作る

MacOS X WorkShop 10.6 のインストーラを SnowLeopard 上の Xcode-3.2.3 で開発した際の備忘録です。

総合的且つ正確な情報は Apple の **PackageMaker User Guide**[39] 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 (Section3.3 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。

6.1　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。

6.2　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。

[39] http://developer.apple.com/library/mac/#documentation/DeveloperTools/Conceptual/PackageMakerUserGuide/Introduction/Introduct

6.3　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[40] 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。

6.4　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

[40] MacOSX-WS-10.6.1.tar.bz2

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。

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。

6.5 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。

● 「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。

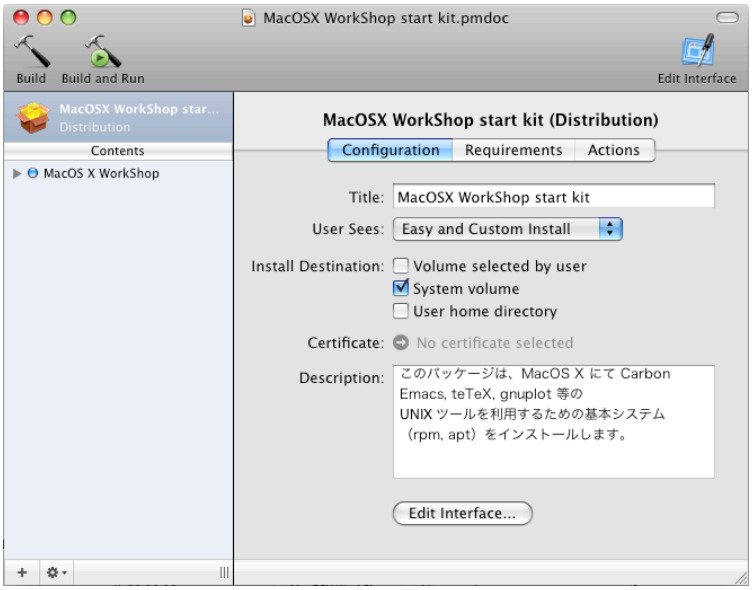

図 1: 「PackageMaker-Distribution-Configuratio」

● 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。

● 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。

6.6 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。
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。

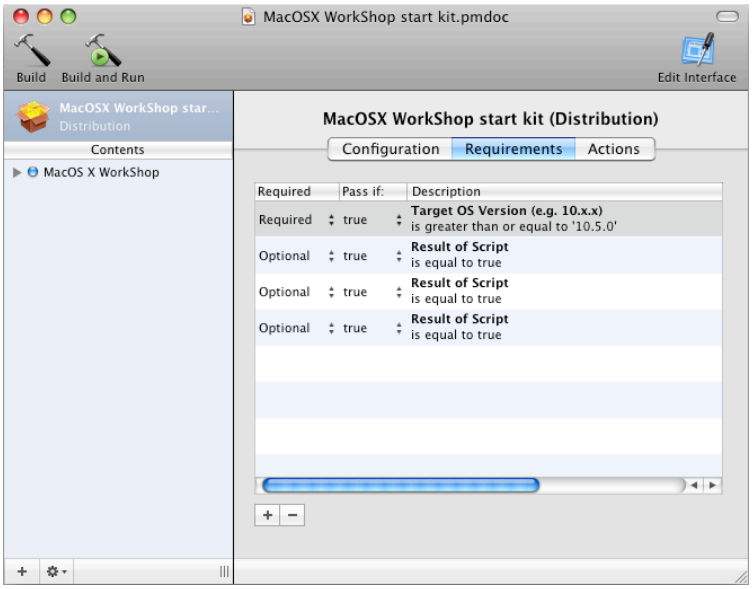

図 2: 「PackageMaker-Distribution-Requirements」

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-10.6.1.tar.bz2**[41] の中にある make-pkg.sh をご覧ください。

[41] MacOSX-WS-10.6.1.tar.bz2

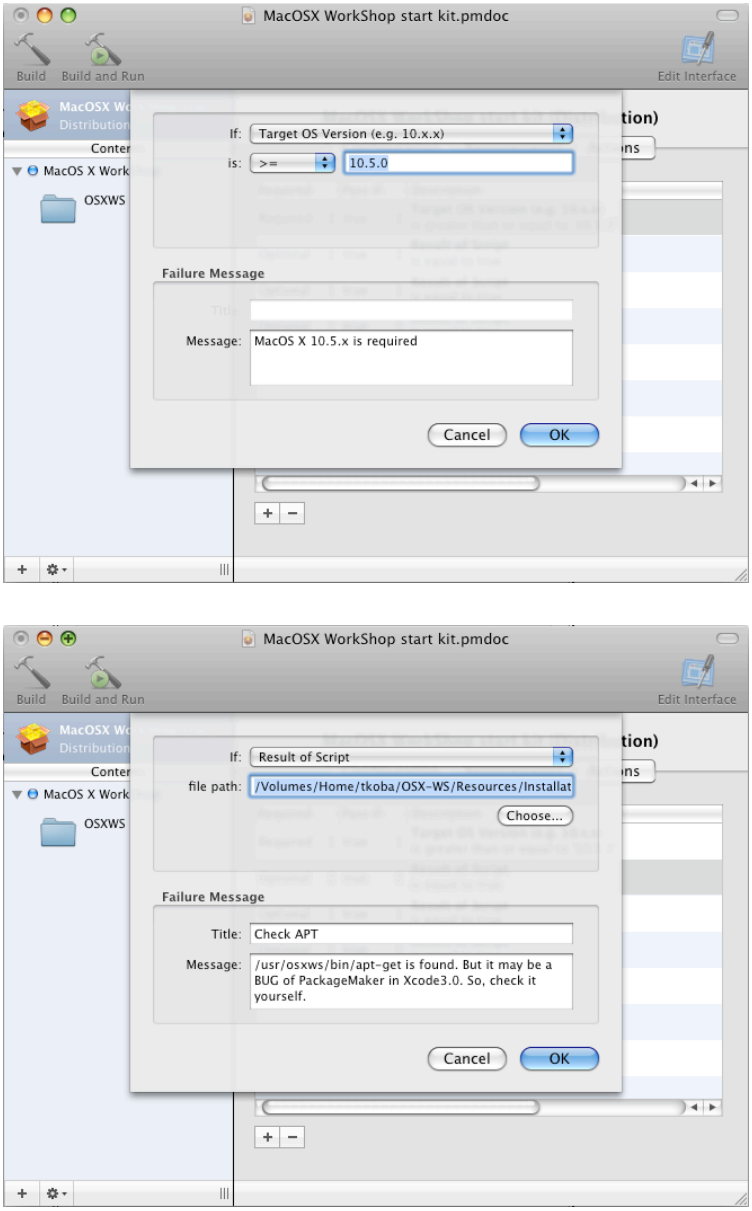

図 3: 「PackageMaker-Distribution-Requirements Describe」

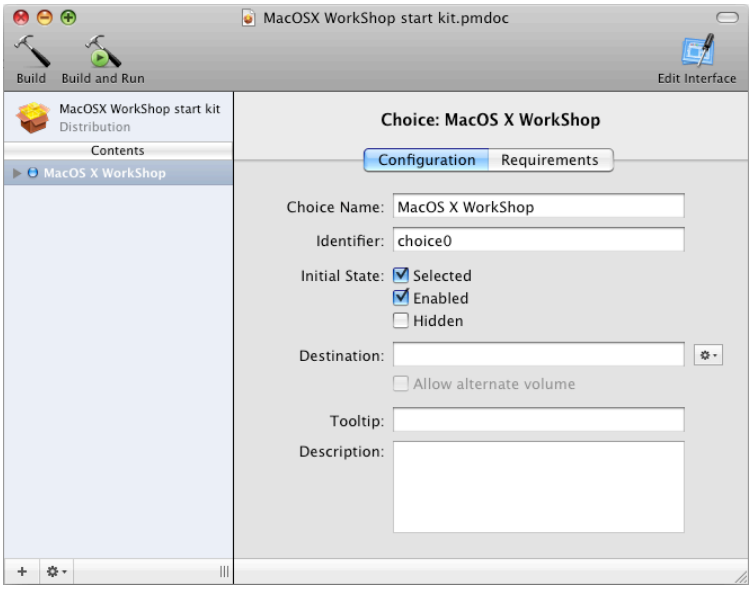

図 4: 「PackageMaker-Contents-Configuration」

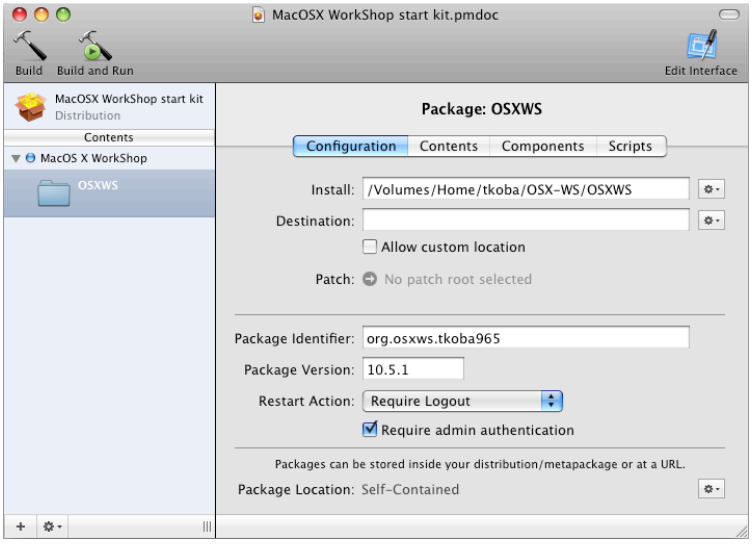

図 5: 「PackageMaker-Package-Configuration」

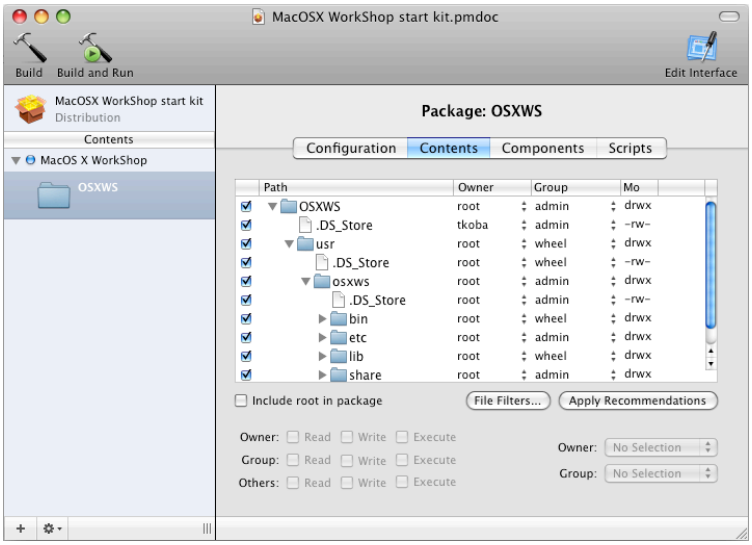

図 6: 「PackageMaker-Package-Contents」

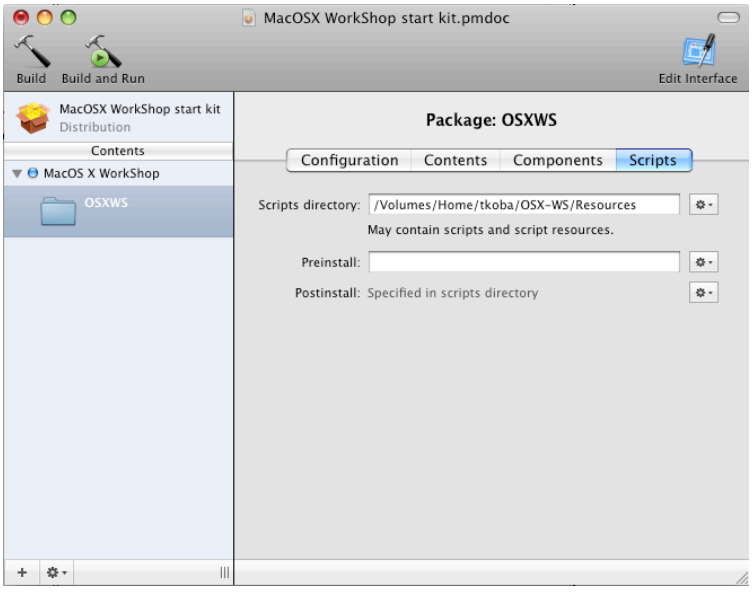

図 7: 「PackageMaker-Package-Scripts」

7　apt-rpm tree を作る

MacOS X WorkShop 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。

7.1　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。

7.2　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。

7.3　パッケージを置く

MacOS X WorkShop は現時点では 10.3 から 10.6 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/{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。

7.4 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

```
[SnowLeopard] genbasedir --progress --bz2only [OSXWS]/SnowLeopard/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。

7.5 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-snow-leopard-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。

8 Tiger 版からの変更点

SnowLeopard 版は Leopard 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。

- Carbon 非依存へ：Snow Leopard x86_64 ではついに非サポート
- X11 依存からの脱却：X11 上での日本語環境メンテに手が回らないのと、Cocoa native のツールで優秀なものが揃ってきたため。
- Universal Binary は PPC PP64 を除き i386, x86_64 の２つに削減。
  configure に与える –host –build –target を工夫して、cross compile の沼からの脱却に成功。
- rpm-4.9.0
- Cocoa Emacs (23.x)
  - 一応動いてはいるけれど、mew の動作が緩慢。-nw での動作は問題なし。
  - ホームディレクトリにあった設定ファイルの置き場を HOME/.emacs.d/ に変更。
- OpenFOAM を採用
  ソルバーを開発するには/usr/osxws 以下が「大文字と小文字を区別する」ボリュームでなけれななりません。
- TeXLive へ移行
- lisp 環境の提供を廃止

8.1 パッケージについて

- 追加したパッケージ
  - MjoGraph
  - OpenFOAM 1.7.1, 2.0.0
  - qt4
  - fftw3r
  - gawk
  - gcc-4.5.1
  - git
  - python-2.6.7 とその関連物
  - vtk
- 削除したパッケージ
  - X11 関連：Plotmtv, Xaw3d, gv

9　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 (SnowLeopard)**[42] をご利用ください。

9.1　Emacs 関連

MacOS X WorkShop 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**[43] 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

[42] http://www.bach-phys.ritsumei.ac.jp/OSXWS/SnowLeopard/rpm2html/
[43] http://macemacsjp.sourceforge.jp/

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

9.2 TeX 関連

パッケージの内容は、土村さんと山本さん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。

pTeXLive 本体 UTF8 で作成しています。
ptexlive-20100711 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 jvf

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」[44] をインストールします。

その他 LaTeXiT LaTeXiT[45]
Apple の Keynote 等で数式を扱う際に利用出来ます。

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

9.3 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.

openMotif The Open Motif runtime components.

ttfonts-ja Free Japanese TrueType fonts

内容はさざなみフォント [46] です。

[44] http://sourceforge.jp/projects/efont/files/
[45] http://www.chachatelier.fr/latexit/latexit-home.php?lang=en
[46] http://sourceforge.jp/projects/efont/files/

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

9.4 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-4.5.1 A GNU Compiler Collection.
apple-gcc-5666.3.gf
gfortran を提供します。

Apple 提供の gcc-4.0,4.2.1 とは update-alternatives で切り替え可能です。

9.5 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。

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 (.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。

`/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。

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。

10　スクリーンショット

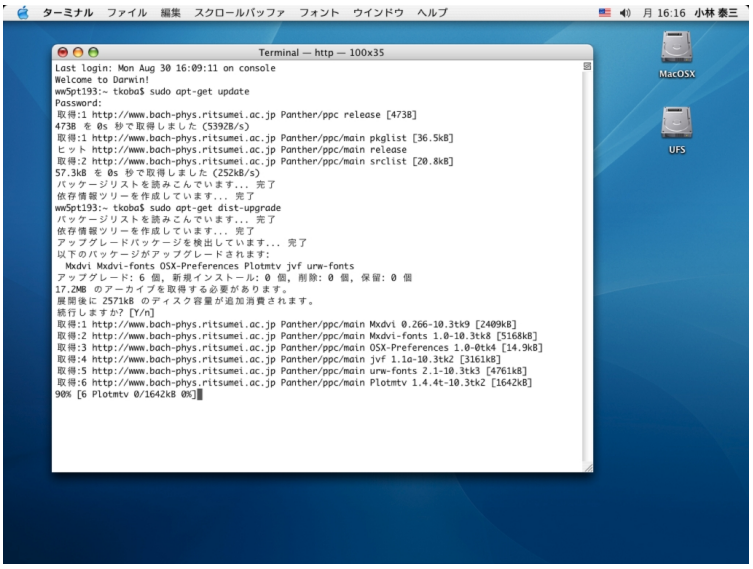

図 8: apt-get でアップデートパッケージをダウンロード中。

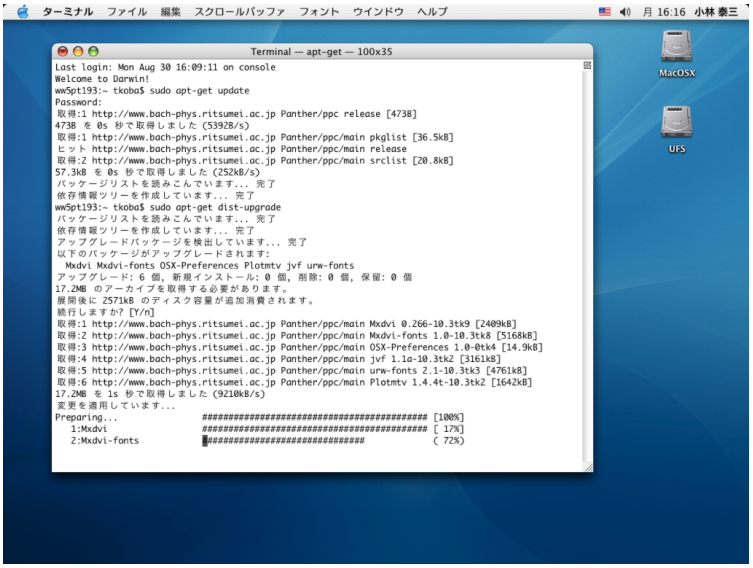

図 9: apt-get でパッケージを更新中。

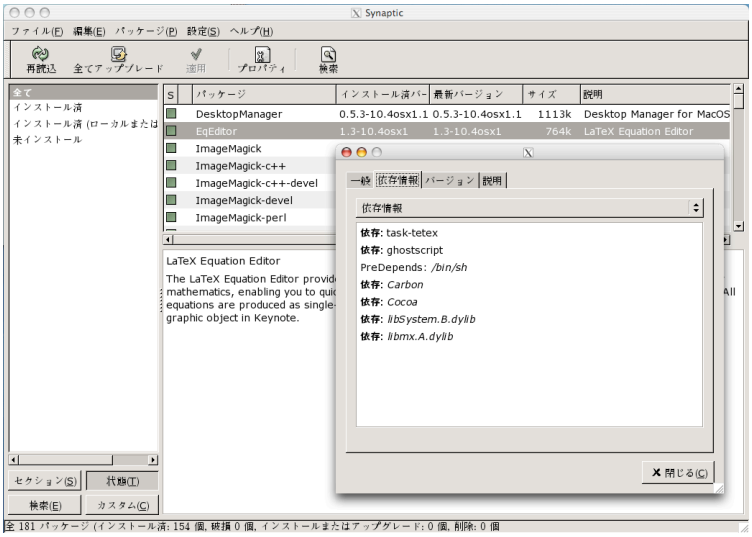

図 10: X11 上の apt-rpm frontend である synaptic

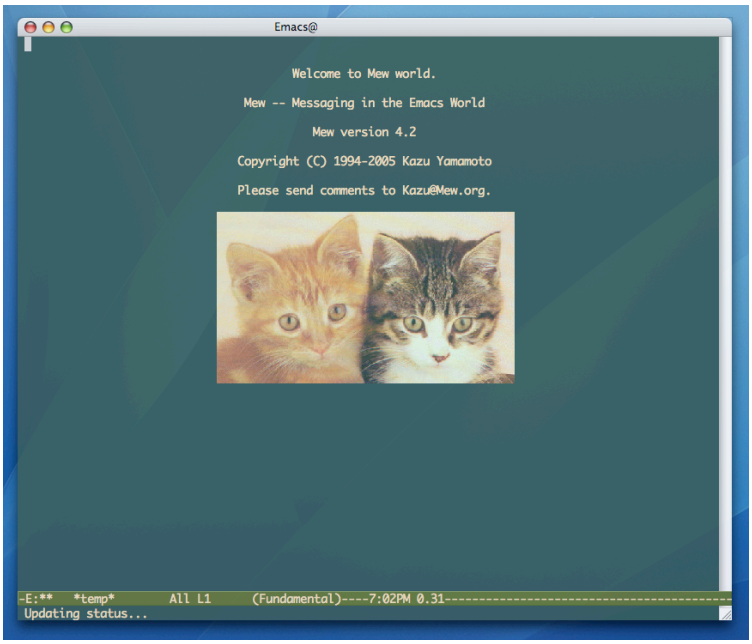

図 11: Emacs で mew を立ち上げているところ。

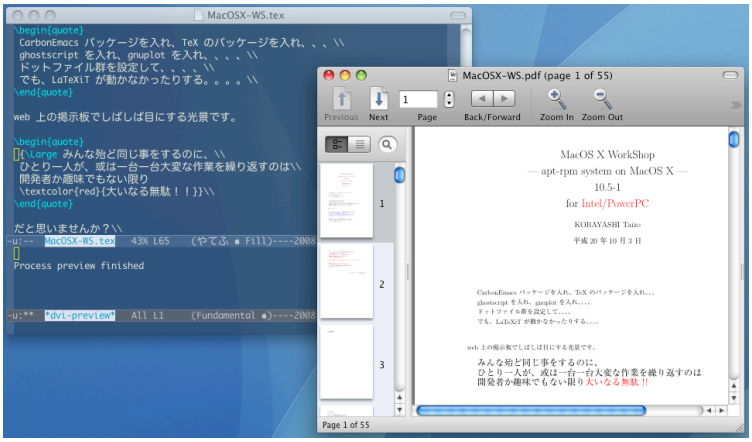

図 12: Emacs で yatex を用いて LaTeX の文章を書き Skim でプレビューしているところ。

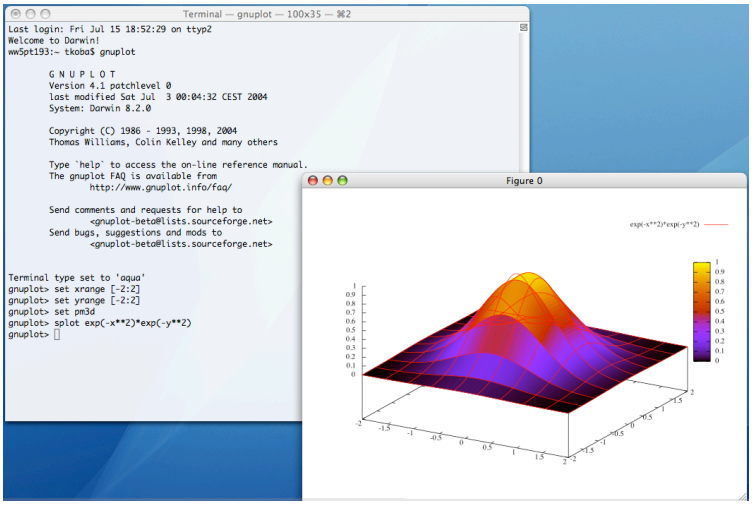

図 13: お馴染み gnuplot です。aquaterm, PDFlib-Lite も同時にインストールされます。

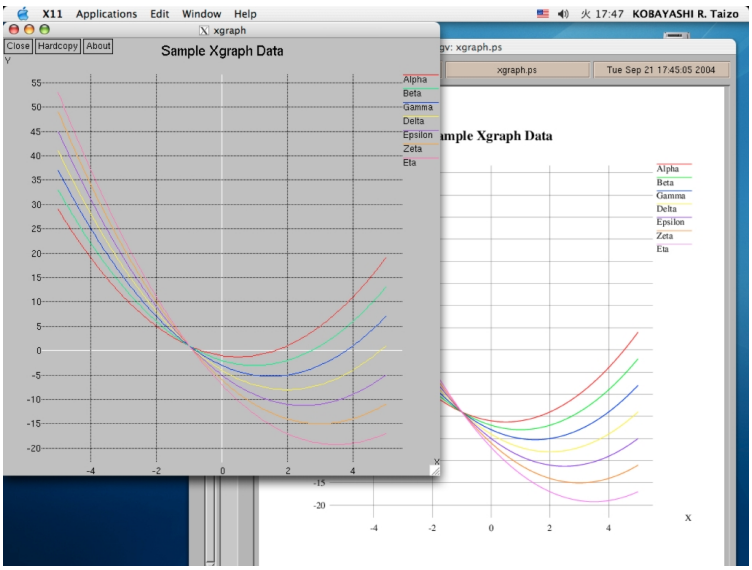

図 14: Vine Linux と同じパッチを適用してあります。Color PS を生成できます。PS ファイルをダブルクリックすれば大抵の場合 PDF に変換でき、プレビューで確認と印刷ができます。

11　過去の議論

ここは **Mac Wiki**[47] にて行われた過去の議論の書庫です。

11.1　10.6-1 公開迄

既知の不具合

- 64bit にて apt と synaptic の間に不具合有り。
  - gtk2 周りを見直して、x86_64 でも synaptic が動くようになったが、pkgPackageManager::GetArchives () が失敗する。
  - rpm-4.9.0 base にして解決。
- OpenFOAM-1.7.1 の libOpenFOAM.dylib が腐っていて、main() に入る前に全部落ちる。
  - gcc-4.4.3, 4.4.4, 4.5.1 と試したが全滅。MacPorts の gcc なら良いらしい。と云う事は、binutils を含めた toolchain を全部揃えないと駄目なのかもしれない。
  - MacPorts 使いの打田さんの協力を得て、取り敢えず動く OpenFOAM rpm はできた。原因は gcc の fat 化に不具合にあった。この gcc fat 化の不具合はまだ若干残っており、これを解決したらリリースに向けた整備に移れる筈。
  - gcc の fat 化修正と DYLD_LIBRARY_PATH 指定の不備を修正し、OpenFOAM-1.7.1 が無事に動くようになった。
  - /usr/osxws 以下が「大文字と小文字を区別する」ボリュームに無いと、rpm で OpneFOAM をインストールした後で手元のソースをコンパイルできません。大文字と小文字を区別しないボリュームでも今の所パッケージに同梱されたバイナリーの実行には問題ないようです。

開発メモ

- guil
  - guile が

```
223: 65* [load-extension "libguile-srfi-srfi-1-v-3" "scm_init_srfi_1"]

/usr/osxws/share/guile/1.8/srfi/srfi-1.scm:223:1: In procedure dynamic-link in expression (
/usr/osxws/share/guile/1.8/srfi/srfi-1.scm:223:1: file:"libguile-srfi-srfi-1-v-3", message:
```

    といって動かない。 すると gnutls が作れず lftp が作れなくなる。 しょうがないので –disable-guile でしのいだ。
    最近の autotools も動きが微妙で、cross compile させるのが難儀している。

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

- X11
  - xmkmf imake がない！ Xaw3d 依存の gv などに影響するが、、もう、いい加減、レガシーな X11 apps は Obsoletes しようかしら。
    - ps を直接見る機会も少なくなりましたし pdf ファイルに変換できますので、 gv を無くしても大丈夫だと思います。-Seto
    - 瀬戸さん、コメントをありがとうございます。Obsoletes しようと思います。–Tkoba
  - 基本的に X11 アプリは縮小させる方向にします。少なくとも X11 上での日本語ハンドリングはサポートしないことにします。
  - xgraph は設計が古いため特にポインタ周辺の 64bit への対応が非常に面倒になっています。32bit 版のみの提供にします。
    - 新山さん打田さんのご尽力で、fat での提供を続けます。
  - Terminal.app と X11 の親和性が良くなったので mlterm, /.xinitrc を廃止。
- AquaTerm
  - 古い 1.0.1 のソース中の NS_HANDLER を`@try{}@catch{}@finally{}` に変更して 10.6 へ移植した。
- TeX
  - Vine Linux と同じく ptetex3-20090610 に更新されたので、漢字コードは自動判別されるようになりました。platex-{sjis,euc} などは廃止します。
  - Vine Linux の ptexlive を基にしたものに移行します。
- OSX アプリの扱いについて
  - これから増えていくであろう/Applications/OSXWS/以下のアプリケーションについて考えています。apt-get でのバージョン管理を想定していない個々のアプリは、メニューバーの「アップデートを確認...」以外にも、起動時に確認したり、環境設定内に確認ボタンがあったりします。小林さんは、OSXWS でインストールするアプリは、こういった機能を無効化して配布するとのことでしたが、マイナーアップデートの度に書き換えて配布するのも大変かと思います。また、ユーザが apt-get dist-upgrade を実行する頻度よりも、アプリケーションを立ち上げる頻度のほうが多い場合も想定されます。ということでせっかくアプリケーション内に用意されたアップデート機能は利用したほうがいいと思っています。

    OSX は、管理者＝ユーザというシチュエーションがもっとも一般的ですが、OSXWS は大学などの端末で管理者が一括管理できるということも想定しています。OSX のユーザは、通常ユーザと管理者の 2 種類が用意されていますが、OSXWS は/usr 以下にファイルを置くため su 権限が必要という状況です。 まず、大きな違いではないですが、/Applications/OSXWS/以下のアプリのアクセス権を su 権限ではなく、OSX の管理者ユーザ権限でインストールしておいてもいいのではないでしょうか？ 一般的なユーザはこのような動作を期待していると思います。

    また、OSX アプリは、もともと依存関係などの不整合は考えられないので、バージョン管理もユーザがアップデートしている可能性もあるけど、apt-get でも一括で update できるというポリシーでうまくいくような気がします。マルチユーザ環境での、通常ユーザはいずれにせよアップデートする権限はないですし。–Seto

– 瀬戸さん、ご意見をありがとう御座います。折角のご意見ですが、これは受け入れる訳にはいきません。なぜならば、OSXWS は「利用環境」としての一貫性を持たせる必要があるからです。apt-rpm で管理されているものは、一貫して apt-rpm の DB との一貫性を保たなければ「環境」全体の整合性が保証できなくなってしまいます。これは MacOSX ネイティブの Cocoa アプリでも事情は同じです。そもそもなぜ /Applications から/Applications/OSXWS に Cocoa/Carbon アプリを移動したのかを振り返ってみてください。ユーザーが自分の手でインストールするものとかぶらないようにする為です。同様に/Applications/OSXWS 以下を OSX の管理者ユーザ権限でインストールするのも意味がありません。また、OSX アプリも依存関係はあります。eb や oniguruma 等の lib, framework は独立した別パッケージで、複数のツールから利用されています。

  重要な事は、OSXWS の利用者が戸惑わないようにする事（フェイルセイフの思想）と「環境」の一貫性を保証する事、です。

  それを考えると、アプリケーション独自の自動アップデート機能を省くのが最もリーズナブルです。手間はそれほどではありません。–Tkoba

– 解説有り難うございます。OSXWS の方針について理解しました。ユーザからみると、OSX 標準アプリのように見えるのが一番ということですね（iPhone アプリのノウハウで、Apple がソフトウェア・アップデートを一般開発者に解放するのがベストですかね？）。私の上の考えは、これまでの経験でいくつかのアプリケーションの最新版に差が生じたため、別に自分でインストールしたものを使う状況だったため、その一番スマートで手間がかからない解決方法は何かという問題からでした。ちなみに MacPort の Cocoa アプリも、アクセス権や/Application/MacPorts に置くという方針は OSXWS と同じですが、アプリ毎のアップデート機能はそのままにしてあります。–Seto

– せっかくですから少しだけ議論を続けさせてもらいます。混乱を避けるためには、OSXWS の一貫性は大切ですが、実用上は個々のアプリの「最新」にばらつきを出さないことも重要だと思います（タイムラグが短ければいいですが数週間の遅れは起こりえます)。ドラッグ＆ドロップでインストールできるタイプのアプリケーションは、最新バージョンにする手段が複数あっても（削除などを許さなければ)、具体的な問題が生じないように思うのですがどうでしょうか？アップデートの重複をさけるため、またより壊れ難いシステムを作るには、ユーザによる更新も Info.plist をみて DB に反映できるようにするのが一番かもしれませんが。–Seto

– もう一度、なぜ/Applications から /Applications/OSXWS に Cocoa/Carbon アプリを移動したのかをよく考えてみてください。自分自身で手で管理をする人ならば、普通に /Applications にドラッグ＆ドロップで入れて貰えればよいだけのことです。わざわざ OSXWS 管理下のものを手で弄る事を可能にするのは労多くして益殆ど無しです。

  実際の OSXWS の利用者（計算機に詳しくない私の近くの物理屋さん）の利用状況を見ると、一旦仕事の環境ができてしまうと不具合が生じない限り、手元のソフトのバージョン等は気にしないものです。そのような大多数の利用者が戸惑わないようにするには、利用者の不意の操作が OSXWS 管理下にあるものの一貫性と整合性を崩さないようにしなければなりません。少なくとも、OSXWS で入れたものを普通に使っているだけなのに OSXWS で構築した環境の整合性が損なわれる事態は避けねばなりません。

  瀬戸さんの思い描いている状況の良さはわかりますが、OSXWS の最も重要視する視点は「利用者が計算機のお守りから解放されて自分自身の仕事に集中できる環境を作る事」にあります。ここが他のディストリビューションと一線を画するところだと自負しています。このことと、私

自身のリソースで実現できる範囲内に物事を収めるバランスが大事です。–Tkoba

– 今回の提案について不採用了解しました。私信のほうでも、背後にある設計思想から丁寧に説明して頂き有り難うございました。–Seto

- ＜私信の抜粋＞
.... 私にしても、.... 自分で研究ツールを開発して配布するとしたら、Linux, *BSD, 等の Unix/GNU 系で広く使われるように、aotutools に対応させたり、MacOSX 向けに Cocoa で配布するなら自動アップデート機能は間違いなく付けます。

これは例えて云えば、一般のソフトウェア開発者はある意味で大工さんであり、あらゆる地域で住まう事が可能な家を設計するようなものだからです。寒冷地の北海道から温暖な沖縄でまで「対応可能」な家を理想として設計するでしょう。

その一方で、OSXWS のようなディストリビューションは、云ってみれば「京都市」を都市設計し行政サービスを行うようなものです。ゴミの出し方などの生活に密着したものから、上下水道・ガス・電気等のインフラの仕様策定と実施・監督までおこない、「京都市」内にある建物が機能して市民の生活の快適性を最大化するのが仕事です。その為にはどのような建物でもある程度「京都市用」にデフォルトから仕様変更が必要になります。

つまり、個々のソフトウェア開発者は開発物の汎用性を重視しますし、ディストリビューターは全体の整合性と環境を利用する立場での一貫性を重視します。

大事なのはそれぞれに限界がある事です。

昨今のソフトウェア開発は、従来のソフトウェア開発の手法をそのまま環境の整備にまで押し広げようとしています。TeX live がその好例でしょう。私に云わせれば「大工さんが外交問題までとりしきっている」ようで.... 滑稽に見えます。

かつて mulitcs が破綻して UNIX が誕生したように、ソフトウェア開発の現場でも「大工さん」と「行政サービス」の分離が近い将来必要になるでしょう。

一方で、ディストリビューションとしての限界もあります。rpm とか deb と云った管理ツールの問題ではありません。それは「京都市」として作ってしまったら、それはあくまで「京都市」であって、同時に「那覇市」にはなり得ない、と云う事です。つまり、その都市の人による好き嫌いは、どうしようもなく出る、と云う事です。

特に、計算機の世界では、スキルのある人は自分の住み心地の良い環境を自分で作る事が用意にできるので、他人のお仕着せを好まないのは当たり前の事なのです。

だからなのでしょう。ディストリビューションと云うある意味でメタなソフトウェア開発そのものが情報処理の学問分野で注目される事はこれまで.... ありませんでした。(Linux box を rpm や deb を使わずにすべてソースから構築する事をほんの少し想像してみれば、rpm や deb などがどれ程画期的なものか解る筈でなのですが、.....)

私が OSXWS の web 上で姉妹ツリーを呼びかけているのは、さまざまな「都市」があってしかるべきだと思うからです。

OSXWS はいわば MacOS X と云う大きな国の中の一つの自治州のようなものです。

開発元が直接公開している hoge.app は MacOS X 国向けの建て売り前の住宅のようなものであるとすると、OSXWS として再配布される hoge.app は、OSXWS 州の認定と保証と必要な改修が済んだ即入居可能な住宅に相当するでしょう。大事なのは MacOS X/hoge.app と MacOS X/OSXWS/hoge.app

とは別物であると認識する必要が「行政」にはある事です。/Applications/OSXWS を作ったのは、それを利用者側にも明示的に示す意図があります。

私は OSXWS のディストリビューターですから、この「行政」の視点を守るのが務めなのです。それをしなければ、早晩「都市」はスラムになります。

.....

¿ そもそも OSX の場合、OS とアプリケーションという 2 つの関係しかないわけですが、
¿ OSXWS は、ディストリビューターという 3 つ目の存在ですから、
¿ そのあり方を考えるためには最初から考え直す必要があるということですね。

.... 大多数の目からすればそうなのでしょうね。しかし、一歩引いて考えてみてください。

全てのソフトは「配布されて」初めて使えるようになる訳です。そしてその配布先には必ず前提とされる「環境」がついて回ります。

私からすると Apple は立派なディストリビューターです。しかも、ディストリビューターとは何ぞや、と云う本質的な問題に答えを持っている数少ないディストリビューターだと思います。

ただ、これまで見えにくかったのは、まだまだ計算機環境の本質がある意味で原始的な状況を引きずっているからだと私は思います。

たとえ話ですが、村が世界だった太古の社会を想像してください。村のインフラ、例えば水をどうするか等は、村人が集まって議論して、長老が結論を出してそれに従う方法でうまく行っていたでしょう。このような状況では「行政」の概念が発生する訳がありません。

しかし、生活の要求水準が上がり、活動範囲が広くなると、調整役としての行政が必要になってきます。

昨今の計算機を取り巻く状況は、複数の計算機を有機的につないで利用しようとしています。いま正に計算機の世界はこの転換期を迎えていると思っています。

- osxws ディレクトリの場所について
  - OSXWS の主要なファイルは /usr/osxws 以下に置かれていますが、この場所を/osxws に引っ越すという提案はどうでしょうか？ これに、OSXWS だけで完結する理由ではなく、単に他に合わせてみてはという提案です。 MacPort が /opt/以下に、Fink が /sw/ 以下に、そして、違うものではありますが Apple の Developer Tool が /Developer/ 以下にファイルを置いています。一方、osxws の場所は Finder を使って見る事のできない /usr/ 以下に置かれています。複数のディストリビューションを同時に利用することが推奨されないとしても、ユーザにとっては他と似ているというのは状況を把握するのが楽になりますし、Finder から見るだけで OSXWS がインストールされたシステムであることが一目瞭然になります。OSXWS の初期のころは、UNIX として行儀の良い/usr/local/ を使っていました。しかし複数の他のディストリビューションとごっちゃになる事を避けるために、/usr/osxws という独立したディレクトリを作りました。今、OSX として行儀がよい場所はどこかなと考えた時、/osxws/ が一番良いのではないかと思いました。場所の変更は、少し混乱を招く可能性はありますが、大きな変更の機会は今くらいしかないので検討いただければと思います。- Seto
    - MacPorts は /opt/local 以下に入ります。/opt は慣習的によく使われていて、例えば Intel コンパイラが/opt/Intel を利用します。FYI - ぜ

* 情報有り難うございます。opt ディレクトリは MacPorts 専用ではなく、Linux などでも使われる一般的な名称のディレクトリだったのですね。 Intel コンパイラなどが入るという事は、OS のパッケージマネージャーを使わずにインストールするソフトウェアの置き場ということですね。もし MacPorts が、/opt/macports/ を使っていれば、OSXWS は /opt/osxws/ という可能性もありましたね。- Seto
* Fink と MacPorts の説明がありましたので参考までに。- Seto
  `http://www.finkproject.org/doc/packaging/fslayout.php?phpLang=ja`
  `http://trac.macports.org/wiki/FAQ#defaultprefix`

議論と要望

- 正式な対応についてではありませんがとりあえず動作確認の報告だけ。まだ十分に検証出来ていませんが Snow Leopard 上で導入した OSXWS 10.6 の Cocoa Emacs, pdflatex, apt-get, gnuplot(X11/aqua term) などは Lion で特に問題なく動いているようです。インストーラー等は確認できていません。–Seto 2011 年 7 月 21 日 (木) 23:42 (JST)
- Lion への対応の予定はいかがでしょうか？ –132.229.223.2 2011 年 7 月 21 日 (木) 17:29 (JST)
- 最新の OSX-Preferences でインストールされる yatex.el で YaTeX-inhibit-prefix-letter t と設定されていて，C-c から始まるコマンドが使えなくなっていますが，これは何か理由があって設定してあるのでしょうか？ –133.5.165.4 2011 年 1 月 25 日 (火) 12:29 (UTC)
  - YaTeX のメーリングリスト [yatex:04567〜04581] でこの件が話題になっていました。C-c アルファベットのキーバインドはユーザー定義用に残すという Emacs の標準の作法を考慮して、Ubuntu/Debian/Vine のデフォルトで YaTeX-inhibit-prefix-letter t となり、作者の広瀬さんも t のほうが良いという認識でした。そして、YaTeX のデフォルトを突然変更できないので、ディストリビューションが配布する初期設定ファイルの中で t としてほしいという意見を書かれていました。今回の変更はそれに従いました。アナウンスが遅れて申し訳ありません。–Seto
- ATOK を使用していると Cocoa emacs のカーソルの色が入力モードを変えても変化しないという問題がありましたが，http://d.hatena.ne.jp/stakizawa/20101225 で対応策が示されています．もしよければ ATOK2008,2010 などを含めて対応していただけるとありがたく思います．来年もよろしくお願い致します． –133.5.165.65 2010 年 12 月 31 日 (金) 12:23 (UTC)
  - ご提案を有り難うございます。（東工大の滝澤さんのページですね。仕事仲間です。。）本件は side effect も無さそうなので、対応しようと思います。–Tkoba
  - emacs-23.2-3 にて対応しました。–Tkoba
  - 動作を確認しました．一段と快適になりました．素早い対応，ありがとうございます．
  - emacs-23.2-4 にてはしもとさんのピコピコパッチ [Macemacsjp-users 1679] を採用しました。–Tkoba
- Vine では山本宗宏さんがご尽力くださり ptexlive に更新されています。http://ml.vinelinux.org/vineseed/msg04498.ht
  OSXWS も ptexlive base にした方が良いかしら。 –Tkoba 2010 年 12 月 27 日 (月) 07:02 (UTC)
- 分かりました．ありがとうございます． –114.185.12.162 2010 年 12 月 26 日 (日) 06:57 (UTC)

- TeX Live の pdflatex(1.40) だとエラーが出ませんね。Skim のサイトにも新しいディストリビューションを使うようにと指示がありますし、OSXWS(teTeX) の pdfLaTeX が 1.21a であることが原因かもしれませんね（リリースノートの要約をみるだけではわかりませんでした)。とりあえず、プレビュア→エディタはオプション無しでも使えますので確認してください。 –Seto 2010 年 12 月 25 日 (土) 12:42 (UTC)
- pdfsync に関する Skim の設定についてのメモです。ちゃんと理解して書き直す必要がありそうえすが。http://web.me.com/setoryohei/osx/skim_as_TeX_previewer.html –Seto 2010 年 12 月 25 日 (土) 08:25 (UTC)
- オプション-synctex=1 がなくてもプレビュアからエディタ（Skim から Emacs へ）のジャンプはできています。私も状況をよく理解できていないようで、調べてみます。 –Seto 2010 年 12 月 25 日 (土) 08:04 (UTC)
- すいません、私もちゃんと確認していなかったみたいですが、同じエラーがでていました。エラーが出てもタイプセットは完了したのですね。YaTeX を使っていて、タイプセットに失敗した時しかログを見ないので気がついていませんでした。 –Seto 2010 年 12 月 25 日 (土) 08:01 (UTC)
- 返信が遅くなり申し訳ありません．ファイル名を入れても unrecognized option '-synctex=1' というエラーがでてきます．`\usepackage{pdfsync}`も tex ファイルに記述しています． –Hashi 2010 年 12 月 25 日 (土) 01:14 (UTC)
- ↓ログインを忘れていました。 –Seto 2010 年 12 月 23 日 (木) 11:43 (UTC)
- タイプセットするファイル名を忘れていませんか？`$ pdflatex -synctex=1 hoge.tex` –141.40.107.12 2010 年 12 月 23 日 (木) 11:42 (UTC)
- pdfsync がうまく動かないのですが, どうすればいいのでしょうか．`$pdflatex -synctex=1 pdflatex: unrecognized` `This is pdfeTeX, Version 3.141592-1.21a-2.2 (Web2C 7.5.4)` –Hashi 2010 年 12 月 23 日 (木) 11:38 (UTC)
- TeX で dvipdfmx の挙動でおかしなところがあるようです．cmex7 というフォントがあると, mktexpk が起動して fmex7 を作ろうとしてフォントが作れないと言って停止します．10.5 の時は同じファイルを問題なく処理できます．インストール時から何もいじっていませんが，どの辺を疑えばよろしいでしょうか． –59.146.84.153 2010 年 5 月 1 日 (土) 11:05 (UTC)
  - 自己フォローです．tetex-3.0-12osx10.6.fat パッケージに　/usr/osxws/share/texmf/fonts/type1/public/tt2001/ という大きさ 0 のファイルが含まれていて, kpsewhich でこちらが上位に来ていました．このファイルを削除すると本来使用されるべき/usr/osxws/share/texmf-dist/fonts/type1/public/tt2001/fmex7.pfb が読み込まれ，正常に動作するようになりました．ご報告まで.
  - ご指摘と解決のご連絡をありがとう御座います。また、対応が遅くなり失礼致しました。tetex の build に必要な t1utils が抜けていたのが原因でした。tetex-3.0-13 にて対処致しました。–Tkoba
- nw モード使えるようになりました。更新、有り難うございます。–Seto 2010 年 4 月 20 日 (火) 08:53 (UTC)
- MacBook Pro OSX 10.6.3 に OSXWS 10.6 を入れてますが、同様に Emacs -nw で Fatal error になりました。システム固有の emacs は動作しました。

```
% emacs -nw Fatal error (11) /usr/osxws/bin/openemacs: line 9: 155 Abort trap /Applications/OSXW
```

–133.19.129.23 2010 年 4 月 16 日 (金) 13:19 (UTC)

  - ご連絡を有り難うございます。23.1.95 ではそうなってしまっていますね。本日 23.1.96 が出て修正されていました。更新致しましたのでご確認ください。–Tkoba

- 私の環境では Emacs の nw モードが使えず Fatal error となってしまうのですが、他の方はどうでしょうか？ –Seto 2010 年 4 月 15 日 (木) 05:28 (UTC)

- Emacs でデフォルト設定では半角バックスラッシュが入らないようです。デフォルト設定では Option キーで入力する方法も使えません。ことえりの環境設定をすれば問題ありませんが，osxws-sec3.el の設定が効いてないようです。 –114.51.214.116 2010 年 4 月 14 日 (水) 14:11 (UTC)

  - バックスラッシュについて US キーボードを使っていることもあり把握していませんでした。Option キーの動作については Emacs23 のデフォルト設定で Meta キーになっているみたいです。 init_osxws.el に (setq ns-option-modifier 'none) と書いてください。-seto

- Emacs で，カーソルのある文字の色が反転しないため，ちょっと見づらいです．対処の方法はあるでしょうか？ –133.5.165.4 2010 年 3 月 30 日 (火) 11:11 (UTC)

  - init_osxws.el でコメントアウトされている default-frame-alist を設定するところで (cursor-type . bar) か (cursor-type . caret)) を追加して試してみてください。デフォルトの設定にしてもよいでしょうね。-Seto
  - (add-to-list 'default-frame-alist '(cursor-type . bar)) -Seto
  - 回答ありがとうございます．10.5 の emacs ではデフォルトのカーソル形状でも文字色が反転していたため見やすかったのですが，これは設定の問題ではなく現在の emacs の仕様（バグ？）と理解してよろしいでしょうか．
  - OSXWS の設定は変わっていないので、Emacs のデフォルトの設定が変わったかバグかもしれないですね。全く同じ default-frame-alist でも Carbo Emacs だと反転しました。他に変数があるか探しましたが分かりませんでした。-Seto

- emacs である種の全角文字が表示されないようです．例えばギリシャ文字の小文字（大文字は大丈夫），いくつかの記号（囲み文字など）などです． –59.146.84.153 2010 年 3 月 19 日 (金) 14:34 (UTC)

  - 報告ありがとうございます。フォントの設定には詳しくないのですが、osxws-sec6.el で mule-unicode-0100-24ff の設定をしているところを外すと表示されますね。何が悪いのかよく理解していませんが報告まで。-Seto
  - そのように設定したら確かに正しく表示されるようになりました．回答ありがとうございました．
  - この対処法では Unicode 文字にヒラギノ丸ゴが使われます。 などのユニコード文字と ascii が違うフォントになってしまうので、他の対処法を考える必要があります。-Seto

- Emacs の起動が遅い件ですが (mac-key-mode 1) の評価に数秒かかっているみたいです。 –Seto 2010 年 3 月 12 日 (金) 05:11 (UTC)

- TeX で euc で書かれたソースを platex2pdf-euc でコンパイルしようとすると，jsarticle.cls を処理できないようです．ソースが utf なら問題ありません．10.5 では両方処理できていたように思うのですが，可能なら対処していただけると嬉しく思います． –121.2.219.143 2010 年 3 月 7 日 (日) 07:33 (UTC)
  - jsarticle.cls の文字コードが 10.5 のときは JIS でしたが、10.6 では utf8 になっているみたいですね。もし今すぐに必要ということでしたら、/usr/osxws/share/texmf/ptex/platex/js/jsarticle.cls を TeX のコンパイル作業をするディレクトリにコピーして、その文字コードを JIS に変更されると良いと思います。–Seto
- emacs を terminal.app から立ち上げるとフォーカスが emacs に移動しないようです．Finder から立ち上げると問題ないようですが．–121.2.219.143 2010 年 2 月 27 日 (土) 02:20 (UTC)
  - ご連絡をありがとうございます。この問題は当方も認識しており、これは現在の 23.1.93 のバグだと思います。正式リリースの 23.2 では修正されるのではないかと思います。–Tkoba
  - 新しい emacs-23.1.93-2osx10.6.fat ではこの問題が起こりませんでした．対処していただき，ありがとうございました．
- いつもお世話になっております．ngraph を使い始めて気が付いたのですがレジェンドの日本語が表示されないようです．英文の入力は支障がないようです． –121.118.82.87 2010 年 2 月 22 日 (月) 06:27 (UTC)
  - ご連絡をありがとう御座います。MacOS X も 10.6 になり、そろそろ X11 からの脱却を考える頃かと思います。一応 ngraph も準備する予定ですが、mjograph 等への移行を始める予定です。–Tkoba
- Cocoa Emacs では utf-8m.el が動作しません。Emacs 本体に同等機能があるはずなので、不要だと思います。 –ぜ 2010 年 1 月 19 日 (火) 05:16 (UTC)
  - いつも有り難うございます。銭谷さんのリリースと同様、最初は CarbonEmacs のみでスタートしようと思います。 Cocoa Emacs 開発時の ToDo に入れておきます。–Tkoba
  - と書いた舌の根も乾かぬうちに何ですが、やはり Snow Leopard では Carbon はなかなか面倒くさそうです。いま Cocoa Emacs の状態を確認しています。–Tkoba
- いつもお世話になっています。OSXWS 10.5 から bash-completion パッケージがなくなったようですが、再録して頂けないでしょうか？ –133.19.126.5 2009 年 12 月 15 日 (火) 07:00 (UTC)
  - 連絡をありがとう御座います。復活させる方向で作業致します。–Tkoba
- お待ちしておりました！ テストなどできる限り協力しますので，アナウンスよろしくお願いします．–133.5.165.65 2009 年 12 月 1 日 (火) 09:07 (UTC)

11.2 dot.emacs 10.6-1 公開迄

OSXWS-10.6-x の cocoa emacs 設定ファイル

- Vine Linux の仕組みを参考に、Emacs のデフォルトの設定を提供する仕組みを大幅に変更する予定です。詳しくは [1] に。とりあえず、最小限の設定のみを提供する方針です。

  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>
;; 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 - 10.6 "))

(setq report-emacs-bug-address "osxws@xxxx.xx.xxx-u.ac.jp")

(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_japanese t
  "A boolean for loading osxws-setting section 1 (language for Japanese)"
  :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: utf-8-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_japanese nil)    <--- en では有効に設定
;;(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>))
  )

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

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; 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: <2011-03-08 08:20:57 seto>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; 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)
  ;; deleting files goes to OS's trash folder
  ;;(setq delete-by-moving-to-trash t)
  ;; start emacsclient server
  ;(server-start)
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 1 language configurations
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(when osxws-default-language_japanese
  ;; japanese settings for Cocoa Emacs Package
  (set-language-environment 'Japanese)
  (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)))

  ;; 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
    ;; if display-height is less than 900, set font size 12pt.
    (let* ((size (if (< (display-pixel-height) 900) 12 14))
           (asciifont "Menlo")
           (jpfont "Hiragino Maru Gothic ProN")
           (h (* size 10))
           (fontspec (font-spec :family asciifont))
           (jp-fontspec (font-spec :family jpfont)))
      (set-face-attribute 'default nil :family asciifont :height h)
      (set-fontset-font nil 'japanese-jisx0213.2004-1 jp-fontspec)
      (set-fontset-font nil 'japanese-jisx0213-2 jp-fontspec)
      (set-fontset-font nil 'katakana-jisx0201 jp-fontspec) ; 半角カナ
      (set-fontset-font nil '(#x0370 . #x03FF) fontspec) ; ギリシャ文字
    )
    (setq face-font-rescale-alist
     '(("^-apple-hiragino.*" . 1.2)
      (".*osaka-bold.*" . 1.2)
  (".*osaka-medium.*" . 1.2)
  (".*courier-bold-.*-mac-roman" . 1.0)
  (".*monaco cy-bold-.*-mac-cyrillic" . 0.9)
  (".*monaco-bold-.*-mac-roman" . 0.9)
  ("-cdac$" . 1.3)))

    ;;-------------------------------------------------------------
    ;; 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)))
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; 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: <2011-03-10 17:13:00 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq auto-mode-alist
      (cons (cons "\\.tex$" 'yatex-mode) auto-mode-alist))
(autoload 'yatex-mode "yatex" "Yet Another LaTeX mode" t)

;; platex with Skim
(setq tex-command "platex2pdf"
      dvi2-command "open -a Skim")
;; pdflatex with Skim
;(setq tex-command "pdflatex -synctex=1"
;      dvi2-command "open -a TeXShop")
;; platex with TeXShop
;(setq tex-command "~/Library/TeXShop/bin/platex2pdf-euc"
;      dvi2-command "open -a TeXShop")
;; pdflatex with TeXShop
;(setq tex-command "pdflatex"
;      dvi2-command "open -a TeXShop")

(setq dviprint-command-format "dvips %s | lpr"
      YaTeX-inhibit-prefix-letter t
      YaTeX-kanji-code 4 ; (1 JIS, 2 SJIS, 3 EUC, 4 UTF-8)
      YaTeX-use-AMS-LaTeX t ; AMS-LaTeX
      section-name "documentclass"
```

```
      makeindex-command "mendex -U"
      bibtex-command "jbibtex -kanji=utf8"
      YaTeX-skip-default-reader  t
      YaTeX-latex-message-code 'utf-8
      YaTeX-use-font-lock t
)

(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)
                ))
))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; 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:
```

変更前

現在の OSXWS-10.6（α 版）に以下の設定ファイルが反映されました。ホームディレクトリにあった Emacs 関連の設定ファイルは全て /.emacs.d/に移動し、ファイル名も以下のように変更されました。

- /.emacs から /.emacs.d/init.el
- /.emacs_osxws.el から /.emacs.d/init_osxws.el
- /.custom_osxws.el から /.emacs.d/custom_osxws.el
- /.yatex.el から /.emacs.d/yatex.el
- /.mew.el から /.emacs.d/mew.el

まだ OSXWS は α 版ということで、これからしばらくこれらの設定ファイルと osxws-sec[0-7].el を変更する可能性があります。ローカルファイルを編集される場合は注意してください。ちなみにデフォルトの設定ファイルは、/usr/osxws/share/OSXWS/jp/にあります。
基本的に採用する方向です。詳細やバグなどのご指摘をいただければ幸いです。–Tkoba

- グローバル共通（init.el 前）

  OSXWS の Emacs は起動時に/usr/osxws/etc/emacs-[EMACSVER]/site-start.d/を読みに行きます。

```
 $ cat /usr/osxws/etc/emacs-[EMACSVER]/site-start.d/20apel-init.el
 (require 'path-util)
$ cat /usr/osxws/etc/emacs-[EMACSVER]/site-start.d/51semi-init.el
 (require 'mime-setup)
$ cat /usr/osxws/etc/emacs-[EMACSVER]/site-start.d/53aspell-init.el

;; 日本語交じりの文書を扱う
;; http://web.archive.org/web/20011214075400/http://queen.heart.ne.jp/cgi-bin/browse?msgid=%3cy
;; http://www.matsusaka-u.ac.jp/~okumura/texfaq/qa/3898.html
;;(eval-after-load "ispell"
;;  '(add-to-list 'ispell-skip-region-alist '("[^\000-\377]+")))
;; ↑この設定がなくても、日本語交じりの文書をスペルチェックできるようです。-Seto
;; ispell の代わりに aspell を使う
(setq-default ispell-program-name "aspell")

$ cat /usr/osxws/etc/emacs-[EMACSVER]/site-start.d/90emacs-lisps-init.el

;; crontab-mode
(autoload 'crontab-mode "crontab-mode" "Major mode for editing crontabs." t)
(add-to-list 'auto-mode-alist '("\\.cron\\(tab\\)?\\'" . crontab-mode))

;; css-mode
(autoload 'css-mode "css-mode" "Major mode for editing CSS style sheets." t)
```

```
(add-to-list 'auto-mode-alist '("\\.css\\'" . css-mode))

;; htmlize
(autoload 'htmlize-buffer "htmlize" "Convert buffer to html." t)
(autoload 'htmlize-region "htmlize" "Convert region to html." t)

;; php-mode
(autoload 'php-mode "php-mode" "Major mode for editing PHP code." t)
(add-to-list 'auto-mode-alist '("\\.php[s34]?\\'" . php-mode))
(add-to-list 'auto-mode-alist '("\\.phtml\\'" . php-mode))
(add-to-list 'auto-mode-alist '("\\.inc\\'" . php-mode))

;; mac-key-mode
;; (autoload 'mac-key-mode "mac-key-mode" "Mac-style key-binding mode." t)
;; mac-key-mode は Cocoa に対応するまで外しています。

;; mac-print-mode
(autoload 'mac-print-mode "mac-print-mode" nil t)
(autoload 'mac-print-buffer "mac-print-mode" nil t)

;; php-mode
(autoload 'php-mode "php-mode" "Major mode for editing PHP code." t)
(add-to-list 'auto-mode-alist '("\\.php[s34]?\\'" . php-mode))
(add-to-list 'auto-mode-alist '("\\.phtml\\'" . php-mode))
(add-to-list 'auto-mode-alist '("\\.inc\\'" . php-mode))

;; smart-dnd
(autoload 'smart-dnd-setup "smart-dnd" "Configurable drag-n-drop feature." t)

;; yaml-mode
(autoload 'yaml-mode "yaml-mode" nil t)
(add-to-list 'auto-mode-alist '("\\.yml\\'" . yaml-mode))

; end

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

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  osxws.el for MacOS X WorkShop
;;            KOBAYASHI Taizo <xxxxxxx@xxxxxxx>
;; 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 - 10.6 "))

(setq osxws-emacs-version
      (progn
        (string-match "\\.[0-9]+$" emacs-version)
        (substring emacs-version 0 (match-beginning 0))))
(setq osxws-emacslisps-elc-path
      (concat "/usr/osxws/share/emacs-"
              osxws-emacs-version "/site-lisp/emacs-lisps/"))
(setq osxws-emacslisps-path "/usr/osxws/share/emacs/site-lisp/emacs-lisps/")

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

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

http://emacs.g.hatena.ne.jp/sakito/20100216 の方針に従って以下を追加しました。–Seto

- (emacs-version) で MacOS X WorkShop の文字列を含ませる
- report-emacs-bug の宛先を OSXWS のメーリングリストのアドレスに

- ローカル共通

Emacs の初期設定は、/.emacs、/.emacs.el、/emacs.d/init.el の順で、最初に見つかったファイルを読み込みます。ここではホームディレクトリに .emacs と .emacs.el がないものと仮定て、 init.el では OSXWS の Emacs かそれ以外かを判定して初期設定ファイルを読み込むようにします。Carbon Emacs パッケージなど OSXWS 以外の Emacs 利用する場合は、 /emacs.d/init_for_other_emacs.el（このファイル名は 1 つの例）に設定を書く事が出来ます。M-x customize を使用して設定値を保存するときに、.emacs が自動で作られないように custom-file を指定しました。custom_for_other_emacs.el は最初に M-x customize を使用したときにファイルが作成されます。

```
 $ cat ~/.emacs.d/init.el

(cond
 ((boundp 'osxws-emacs-flag)
```

```
  ;; for Mac OS X WorkShop
  (setq user-init-file "~/.emacs.d/init_osxws.el")
  (setq custom-file "~/.emacs.d/custom_osxws.el"))
 (t
  ;; for others
  (setq user-init-file "~/.emacs.d/init_for_other_emacs.el")
  (setq custom-file "~/.emacs.d/custom_for_other_emacs.el"))
 )

(if (file-exists-p (expand-file-name user-init-file))
    (load-file (expand-file-name user-init-file)))
(if (file-exists-p (expand-file-name user-init-file))
    (load-file (expand-file-name custom-file)))
```

OSXWS 固有の設定をここで読み込みます。また、ユーザーのカスタマイズはこのファイルの後半に書き込まれる事を想定し、その設定例（一部はコメントアウト）が置かれています。

```
 $ cat ~/.emacs.d/init_osxws.el

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; ~/.emacs.d/init_osxws.el
;;  .emacs for MacOS X WorkShop
;; Time-stamp: <2008-07-03 18:24:55 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq user-init-file "~/.emacs.d/init_osxws.el")

;; Section 0 fundamental configurations
(load-file (concat osxws-emacslisps-path "osxws-sec0.el"))

;; Section 1 language setting
(load-file (concat osxws-emacslisps-path "osxws-sec1.el"))

;; Section 2 appearance setting
(load-file (concat osxws-emacslisps-path "osxws-sec2.el"))

;; Section 3 keyboard/keybindig setting
(load-file (concat osxws-emacslisps-path "osxws-sec3.el"))

;; Section 4 shell-command setting
(load-file (concat osxws-emacslisps-path "osxws-sec4.el"))

;; Section 5 inline-patch setting
(load-file (concat osxws-emacslisps-path "osxws-sec5.el"))
```

```
;; Section 6 CarbonEmacs window mode setting
(load-file (concat osxws-emacslisps-path "osxws-sec6.el"))

;; Section 7 anything else setting
(load-file (concat osxws-emacslisps-path "osxws-sec7.el"))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Edit
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Startup-message
;; Once you are familiar with the contents of the startup screen,
;; you may inhibit the startup screen.

;; YaTeX
(if (file-exists-p "~/.emacs.d/yatex.el")
      (load-file "~/.emacs.d/yatex.el"))

;; Mew 6.2  -  Messaging in the Emacs World
(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"))

;; 起動時にメッセージ画面を表示しない
(setq inhibit-startup-message t)

;; CarbonEmacs window mode setting
(when (eq window-system 'ns)
  ;; window size
  ;;(setq default-frame-alist
  ;;   (append
  ;;     '((width . 100) (height . 40) (top . 0) (left . 520))
  ;;       (vertical-scroll-bars . nil)
  ;;      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>))
```

```
)

;; Command + カーソルキーを OSX 標準の動作にする
;; default : ns-next-frame in ns-win.el
(define-key global-map [s-left] 'move-beginning-of-line)
(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)

;;; show the present time on status bar
(when (equal (substring (shell-command-to-string "defaults read -g AppleLocale") 0 2) "ja")
       (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)

;; End Edit:


$ cat ~/.emacs.d/custom_osxws.el

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; customize saves customizations here in Emacs of MacOS X WorkShop
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(setq custom-file "~/.emacs.d/custom_osxws.el")
```

- グローバル共通（init.el 後

```
 $ cat /usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-sec0.el

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  osxws.el for MacOS X WorkShop
;; Time-stamp: <2010-01-28 12:33:04 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

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

```
;; 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)

;; deleting files goes to OS's trash folder
(setq delete-by-moving-to-trash t)

;;; start emacsclient server
(server-start)

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

 $ cat /usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-sec1.el

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  osxws.el for MacOS X WorkShop
;; Time-stamp: <2010-01-28 12:34:19 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 1 language configurations
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; japanese settings for Carbon Emacs Package
(when
    (equal (substring (shell-command-to-string "defaults read -g AppleLocale") 0 2) "ja")
    (set-language-environment 'Japanese)
```

```
    (prefer-coding-system  'utf-8-unix)
     )

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

```
 $ cat /usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-sec2.el
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  osxws.el for MacOS X WorkShop
;; Time-stamp: <2008-10-01 12:09:09 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

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

;;; hide tool-bar
(tool-bar-mode 0)

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

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

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

```
 $ cat /usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-sec3.el
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  osxws.el for MacOS X WorkShop
;; Time-stamp: <2010-01-28 12:41:44 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

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

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

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

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

;; fn + delete でカーソル位置の文字を削除
(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-\\])

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

```
 $ cat /usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-sec4.el
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  osxws.el for MacOS X WorkShop
;; Time-stamp: <2008-07-03 18:11:33 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

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

;;; 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)

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

```
 $ cat /usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-sec5.el
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  osxws.el for MacOS X WorkShop
;; Time-stamp: <2008-10-01 12:08:33 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 5 inline-patch by Hashimoto-san
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(when (eq window-system 'ns)
  (setq default-input-method "MacOSX")
  (add-hook 'minibuffer-setup-hook 'mac-change-language-to-us)
 ;; (mac-add-ignore-shortcut '(control))
  (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 [(alt \ )] 'toggle-input-method)
;; start up by Shift-Space
(global-set-key [?\S-\ ] 'toggle-input-method)

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

```
 $ cat /usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-sec6.el
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  osxws.el for MacOS X WorkShop
;; Time-stamp: <2008-07-03 18:17:15 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

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

(when (eq window-system 'ns)
  ;; if display-height is less than 900, set font size 12pt.
  (let* ((fontsize (if (< (display-pixel-height) 900) 12 14))
         (hira-fontsize (+ fontsize 2)))
     (create-fontset-from-ascii-font
     (concat "Menlo-" (number-to-string fontsize) ":weight=normal:slant=normal") nil "menlomaru
    (set-fontset-font "fontset-menlomarugo"
      'unicode
      (font-spec :family "Hiragino Maru Gothic ProN" :size hira-fontsize)
      nil
      'append)
    (set-fontset-font "fontset-menlomarugo"
      '(#x0080 .  #x024F)
      (font-spec :family "Menlo")
      nil
      'prepend)
    (add-to-list 'default-frame-alist '(font . "fontset-menlomarugo")))

  ;;-------------------------------------------------------------
  ;; 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>")))))
 ;;-------------------------------------------------------------
 )

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

http://emacs.g.hatena.ne.jp/sakito/20100127 に従った設定方法に変更しました。

font の設定は Snow Leopard と Cocoa Emacs の組み合わせを想定しました。

OSXWS10.6 が Leopard を対象とする場合は、リンク先のように対応する必要があるかもしれません。http://d.hatena.ne.jp/setoryohei/20100915

```
$ cat /usr/osxws/share/emacs/site-lisp/emacs-lisps/osxws-sec7.el

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  osxws.el for MacOS X WorkShop
;; Time-stamp: <2008-07-03 18:18:18 tkoba>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 7 anything 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)))

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

- YaTeX の設定ファイル

  init_osxws.el から読み込まれます。

```
 $ cat ~/.emcas.d/yatex.el

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  ~/.emcas.d/yatex.el for MacOS X WorkShop
;;            KOBAYASHI Taizo <xxxxxxxxx@xxxxxx.com>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; yatex の設定
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(setq auto-mode-alist
      (cons (cons "\\.tex$" 'yatex-mode) auto-mode-alist))
(autoload 'yatex-mode "yatex" "Yet Another LaTeX mode" t)

;;; platex と Skim
(setq tex-command "platex2pdf"
      dvi2-command "open -a Skim")
```

```
;;; pdflatex と Skim
;;(setq tex-command "pdflatex -synctex=1"
;;     dvi2-command "open -a Skim")
;;; platex と TeXShop
;;(setq tex-command "~/Library/TeXShop/bin/platex2pdf-euc"
;;      dvi2-command "open -a TeXShop")
;;; pdflatex と TeXShop
;;(setq tex-command "pdflatex"
;;      dvi2-command "open -a TeXShop")

(setq dviprint-command-format "dvips %s | lpr"
      YaTeX-inhibit-prefix-letter t
      YaTeX-kanji-code 4 ; (1 JIS, 2 SJIS, 3 EUC, 4 UTF-8)
      YaTeX-use-AMS-LaTeX t ; AMS-LaTeX
      section-name "documentclass"
      makeindex-command "mendex -U"
      bibtex-command "jbibtex -kanji=utf8"
      YaTeX-skip-default-reader  t
      YaTeX-latex-message-code 'utf-8
      YaTeX-use-font-lock t
)

(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)
  ))
))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; yahtml の設定
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(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)
       ))

;;;; 常に<p>をいれたら</p>
(setq yahtml-always-/p t)
;;;; 常に<li>をいれたら</li>
(setq yahtml-always-/li t)

;; End:
```

YaTeX の ML で議論されていた YaTeX-inhibit-prefix-letter ですが、広瀬さんが推奨されている t に変更してみました。(慣れるのは大変ですが)

Skim と Emacs の連携。TeX のソースに `\usepackage{pdfsync}` を書いておき、pdflatex -synctex=1 でタイプセットすると、Skim からシフト + コマンド + クリックで Emacs の対応箇所にジャンプします。OSXWS だと、emacsclient が/usr/osxws/bin/以下にあるため、環境設定の Sync のところを以下のように設定する必要があります。

```
Preset: Custom
Command: /usr/osxws/bin/emacsclient
Arguments: --no-wait +%line "%file"
```

また、Skim.app/Contents/SharedSupport/displayline を使って、xdvi-search.el のようなものを書けば、エディタのカーソル箇所から対応する場所のプレビューを表示させることもできるはずですね。

- http://sourceforge.net/apps/mediawiki/skim-app/index.php?title=TeX_and_PDF_Synchronization
- http://mactex-wiki.tug.org/wiki/index.php?title=SyncTeX#Carbon_Emacs.2FSkim

OSXWS-10.6-x の変更点

- 追加した項目
  追加された dired で削除したファイルを OSX のゴミ箱に入れる機能を有効にする設定を Section0 に追加しました。 -Seto

```
;; deleting files goes to OS's trash folder
(setq delete-by-moving-to-trash t)
```

Cocoa Emacs では mac-key-mode なしでも、OSX の標準的なショートカットがデフォルトで設定されているみたいです。fn+delete キーや Command + カーソルキーの振る舞いなど、標準の動作から抜けている部分を補ってみました。Command- 右、左については、「ウインドウのフォーカスの移す」がデフォルトで設定されていましたので、そのカスタマイズということでローカルの osxws.el で設定しました。-Seto

Section 3 への追加

```
;; fn + delete でカーソル位置の文字を削除
(define-key global-map [kp-delete] 'delete-char)
```

osxws.el への追加

```
;; 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)
```

osxws-emacs-version と osxws-emacslisps-path を 99osxws.el で設定しました。 init_osxws.el で設定を読み込む時に使います。

- 削除した項目

  既にコメントアウトされている設定を試しに取り除いてみました。-Seto

  Sec0

```
;(set-clipboard-coding-system 'utf-8)
;(set-terminal-coding-system 'utf-8) ; 'euc-jp-unix)
;(set-language-environment 'Japanese)
;(set-default-coding-systems 'utf-8)
;(set-keyboard-coding-system 'utf-8) ; (if window-system 'sjis-mac 'utf-8))
;(load "menu-tree")
```

  Sec2

```
 ;(require 'paren)

;;; colorizing mark region
;; (setq transient-mark-mode t)

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

  Sec3

```
 ;; (setq ns-control-modifier 'control)
;; (setq ns-option-modifier 'meta)
;; 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)                      ; Greek
;   (#x400 . #x4ff)                      ; Cyrillic
;   (#x2000 . #x206f)                    ; punctuation mark
;   (#x2103 . #x2103)                    ; centigrade degree
;   (#x212b . #x212b)                    ; angstrom
;   (#x2190 . #x21ff)                    ; Arrow
;   (#x2200 . #x22ff)                    ; mathematical symbol
;   (#x2300 . #x23ff)                    ; engineering symbol
;   (#x2460 . #x2473)                    ; number in circle
;   (#x2500 . #x257f)                    ; ruled line
;   (#x25a0 . #x25ff)                    ; mosaic fragment
;   (#x2600 . #x26ff)                    ; other symbol
;   ))
;; C. end
```

Sec6

```
;; Transparency3
;; (setq frame-alpha-lower-limit 20)
;; anti alias (drag and drop)
;; (setq ns-antialias-text t)
```

Sec7

```
 ;; No adding newline at end of buffer
 ;; (setq next-line-add-newlines nil)
```

YaTeX と Mew の設定は /.emacs.d/osxws.el に移動しました。

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

;; Mew 6.2  -  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"))
```

display-time のフォーマットの設定を /.emacs.d/osxws.el に移動しました。設定されたフォーマットを、後から読まれる.emacs.d/osxws.el などで再設定することは簡単ですが、Emacs のデフォルトのフォーマットに戻す方法が簡単でないため、このようなカスタマイズはユーザーが編集可能なところに置くべきと判断しました。-Seto

```
;;; show the present time on status bar
(when (equal (substring (shell-command-to-string "defaults read -g AppleLocale") 0 2) "ja")
       (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)
```

sec3 から以下を取り除きました。ns-command-modifier はデフォルトの'super のままにしておくと、Cocoa Emacs のデフォルトで OSX の基本的なショートカットが使えるみたいです。また、銭谷さんに確認したところ、mac-key-mode は現在のところ Carbon Emacs 専用とのことでした。

```
;;; use "command" keys as alt
(setq ns-command-modifier 'alt)

;;; mac-key-mode
;; (short cut of OSX applications)
(when (or (eq window-system 'mac) (eq window-system 'ns))
  (require 'mac-key-mode)
  (mac-key-mode 1))
```

コメント

- スペスチェック M-X ispell をしたいのですが、ispell-init-process: Error: No word lists can be found for the language "en_US". というメッセージが出てきます。どうすれば良いでしょう？ –194.3.110.235 2010 年 1 月 19 日 (火) 04:53 (UTC)
  - `$ sudo apt-get remove aspell` した後に、`$ sudo apt-get install` [抜けたパッケージ] してください。–Tkoba
- 忙しい中お疲れさまです。OSXWS 全体を 10.6 用 (64/32 ビットユニバーサル) にビルドしなおす作業は大変そうですね。Cocoa Emacs への対応は、時期をずらした方が現実的かもしれないですね。SnowLeopard 上書きアップデートで、OSXWS の Emacs/LaTeX 共に特に問題は無さそうです。–Seto 2009 年 8 月 31 日 (月) 15:23 (UTC)
- 瀬戸さん、コメントを有り難うございます。9 月も予定で一杯になっていますが 10.6 も出てしまった事ですし、時間を作って作業を進めて行きますのでよろしくお願い致します。 –Tkoba 2009 年 8 月 31 日 (月) 12:31 (UTC)
- exec-path と環境変数 PATH の追加が/usr/osxws/share/emacs-22.3/site-lisp/site-start.el で行われますが、osxws-sec0.el にあるほうが良いと思います。理由ですが、Cocoa Emacs を試していたときに、それに osxws-sec[1-7].el を読み込ませてなるべく CarbonEmacs と同じ動作になるほうが望ましいと思いました。–Seto 2009 年 8 月 22 日 (土) 09:43 (UTC)

12 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 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 [48] 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

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

- 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

索 引