

Configurations for GNU Emacs

Takaaki Ishikawa

13 年 1 月 26 日

Table of Contents

- [1. はじめに](#)
- [2. init.el のヘッダ](#)
 - [2.1. cl を使う](#)
 - [2.2. リストで関数を渡せる autoload-if-found を使う](#)
 - [2.3. \[eval-after-autoload-func.el\] 遅延読み込み](#)
 - [2.3.1. マクロ版](#)
 - [2.3.2. 関数版](#)
 - [2.4. パス設定](#)
 - [2.5. 警告の抑制](#)
- [3. コア設定](#)
 - [3.1. 言語／文字コード](#)
 - [3.2. 日本語入力](#)
 - [3.3. 基本キーバインド](#)
 - [3.4. ナローイングするか](#)
 - [3.5. バッファの終わりでの newline を禁止する](#)
 - [3.6. 常に最終行に一行追加する](#)
 - [3.7. 長い文章を右端で常に折り返す](#)
 - [3.8. バッファが外部から編集された場合に自動で再読み込みする](#)
 - [3.9. 同じバッファ名が開かれた場合に区別する](#)
- [4. カーソル移動](#)
 - [4.1. バッファ内の移動](#)
 - [4.2. バッファ間の移動](#)
 - [4.3. スクロールを制御](#)
 - [4.4. スクロールで表示を重複させる行数](#)
 - [4.5. \[SmoothScroll.el\] カーソル固定でスクロールする](#)
 - [4.5.1. キーバインド](#)
 - [4.6. \[point-undo.el\] カーソル位置を簡単にたどる](#)
 - [4.6.1. キーバインド](#)
 - [4.7. \[cycle-buffer.el\] カレントバッファの表示切り替え](#)
 - [4.7.1. キーバインド](#)
- [5. 編集サポート](#)
 - [5.1. 矩形編集](#)
 - [5.2. Yank 時に装飾を取る](#)
 - [5.3. ファイル保存時に時間を記録する](#)
 - [5.4. 選択リージョンを使って検索](#)

- [5.5. ChangeLog モード](#)
- [5.6. テキストモード](#)
- [5.7. C/C++モード](#)
- [5.8. HTML モード](#)
- [5.9. Info モード](#)
- [5.10. スペルチェック](#)
 - [5.10.1. キーバインド](#)
- [5.11. リアルタイムスペルチェック](#)
- [5.12. \[latex-math-preview.el\] TeX 数式をプレビュー](#)
- [5.13. \[po-mode.el\] 翻訳ファイルの編集](#)
- [5.14. \[word-count.el\] リージョン内の文字をカウントする](#)
 - [5.14.1. キーバインド](#)
- [5.15. \[yatex.el\] YaTeX モード](#)
- [5.16. \[wclock.el\] 世界時計](#)
- [5.17. \[yasnipet.el\] Emacs 用のテンプレートシステム](#)
- [5.18. \[sdc.el\] 英辞郎で英単語を調べる](#)
 - [5.18.1. キーバインド](#)
- [5.19. \[lookup.el\] 辞書](#)
 - [5.19.1. キーバインド](#)
- [5.20. \[cacoo\] Cacoo で描く](#)
- [5.21. \[jedit\] バッファ内の同じ文字列を一度に編集する](#)
- [6. 表示サポート](#)
 - [6.1. モードラインのモード名を短くする](#)
 - [6.2. モードラインの Narrow を短くする](#)
 - [6.3. visible-bell のカスタマイズ](#)
 - [6.4. 常に **scratch** を表示して起動する](#)
 - [6.5. バッテリー情報をモードラインに表示する](#)
 - [6.6. スクロールバーを非表示にする](#)
 - [6.7. ツールバーを非表示にする](#)
 - [6.8. 起動時のスプラッシュ画面を表示しない](#)
 - [6.9. カーソル行の行数をモードラインに表示する](#)
 - [6.10. カーソル行の関数名をモードラインに表示する](#)
 - [6.11. 時刻をモードラインに表示する](#)
 - [6.12. 対応するカッコをハイライトする](#)
 - [6.13. \[migemo.el\] ローマ字入力で日本語を検索する](#)
 - [6.14. \[anything.el\] 何でも絞り込みインターフェイス](#)
 - [6.14.1. キーバインド](#)
- [7. メディアサポート](#)
 - [7.1. \[bongo.el\] Emacs のバッファで音楽ライブラリを管理する](#)
 - [7.2. \[GoogleMaps.el\] GoogleMaps を Emacs 内で使う](#)
 - [7.3. \[org-google-weather.el\] org-agenda に天気を表示する](#)
- [8. 履歴／ファイル管理](#)
 - [8.1. Undo バッファを無限に取る](#)
 - [8.2. バッファ保存時にバックアップファイルを生成する](#)
 - [8.3. ミニバッファの履歴を保存しリストアする](#)
 - [8.4. 履歴サイズを大きくする](#)
 - [8.5. Emacs 終了時に開いていたバッファを起動時に復元する](#)

- [8.6. 最近開いたファイルリストを保持](#)
- [8.7. 深夜にバッファを自動整理する](#)
- [8.8. \[auto-save-buffers.el\] 一定間隔でバッファを保存する](#)
- [8.9. \[backup-dir.el\] バックアップファイルを一箇所に集める](#)
- [8.10. \[session.el\] 様々な履歴を保存し復元に利用する](#)
- [9. 開発サポート](#)
 - [9.1. 便利キーバインド](#)
 - [9.2. \[gist.el\] Gist インターフェイス](#)
 - [9.3. \[doxymacs.el\] Doxygen のコメントを簡単に入力する](#)
 - [9.4. \[matlab.el\] Matlab 用の設定](#)
 - [9.5. \[auto-complete.el\] 自動補完機能](#)
 - [9.6. \[auto-complete-clang.el\] オムニ補完](#)
 - [9.6.1. 参考サイト](#)
 - [9.7. \[hideshowvis.el\] 関数の表示／非表示](#)
 - [9.7.1. キーバインド](#)
- [10. Org Mode](#)
 - [10.1. 基本設定](#)
 - [10.2. contribution を使う](#)
 - [10.3. iCal との連携](#)
 - [10.4. スピードコマンド](#)
 - [10.5. ソースブロックの入力キーをカスタマイズ](#)
 - [10.6. Pomodoro](#)
 - [10.7. face 関連](#)
 - [10.8. TODO キーワードのカスタマイズ](#)
 - [10.9. \[org-agenda\]](#)
 - [10.10. \[appt.el\] アラーム設定](#)
 - [10.11. \[org-capture\] 高速にメモを取る](#)
 - [10.12. \[org-refile\]](#)
 - [10.13. \[org-babel\]](#)
 - [10.14. \[MobileOrg\] iOS との連携](#)
 - [10.15. \[org-tree-slide.el\] Org でプレゼンテーション](#)
 - [10.16. \[org-fstree\] ディレクトリ構造を読み取る](#)
 - [10.17. \[calfw-org\] calfw に org の予定を表示する](#)
 - [10.18. \[org-export-generic\] エクスポート機能を拡張する](#)
 - [10.19. \[org-odt\] ODT 形式に出力](#)
 - [10.20. README を常に org-mode で開く](#)
 - [10.21. その他](#)
 - [10.21.1. キーバインド](#)
 - [10.22. org-mode の latex エクスポート関数をオーバーライド](#)
- [11. フォント／配色関連](#)
 - [11.1. 正規表現を見やすくする](#)
 - [11.2. カーソル行に色をつける](#)
 - [11.3. カーソルの色](#)
 - [11.4. カーソルを点滅させない](#)
 - [11.5. フォント設定](#)
 - [11.5.1. フォントのインストール方法](#)
 - [11.5.2. フォントチェック用コード](#)

- [11.6. パッチをカラフルに表示する](#)
- [11.7. 背景を黒系色にする](#)
- [11.8. \[rainbow-mode.el\] 配色のリアルタイム確認](#)
 - [11.8.1. 色一覧](#)
- [12. Macintosh 専用](#)
 - [12.1. \[igrep\]](#)
- [13. フレーム/ウィンドウ制御](#)
 - [13.1. 起動時の設定](#)
 - [13.2. \[elscreen.el\] Emacs バッファをタブ化](#)
 - [13.3. \[e2wm.el\] 二画面表示](#)
 - [13.4. \[frame-ctr.el\] キーボードでフレームの場所を移す](#)
 - [13.4.1. キーバインド](#)
 - [13.5. \[popwin.el\] ポップアップウィンドウの制御](#)
- [14. ユーティリティ関数](#)
 - [14.1. \[pomodoro.el\] ポモドーロの実践](#)
 - [14.2. \[utility.el\] 自作してテスト中の便利関数群](#)
- [15. provide](#)

1. はじめに

- [init.el](#) 本体は, [github](#) に公開しています.
- [init.org](#) から [init.el](#), [init.pdf](#), [wiki](#) を生成しています.
- 一部の関数定義を [utility.el](#) に分離しています.

2. init.el のヘッダ

```
;;; Configurations for Emacs
;;;                                     Takaaki ISHIKAWA <takaxp@ieee.org>
;;; Cite: http://www.mygooglest.com/fni/dot-emacs.html (GREAT!)
```

2.1. cl を使う

http://lisperblog.blogspot.com/2008/12/blog-post_18.html

```
(eval-when-compile (require 'cl))
```

2.2. リストで関数を渡せる *autoload-if-found* を使う

<http://d.hatena.ne.jp/jimo1001/20090921/1253525484>

eval-after-load とのペアでマクロ化したバージョンもある (次章参照) .

<http://e-arrows.sakura.ne.jp/2010/03/macros-in-emacs-el.html>

```
(defun autoload-if-found (functions file &optional docstring interactive type)
  "set autoload iff. FILE has found."
  (if (not (listp functions))
      (setq functions (list functions)))
  (and (locate-library file)
       (progn
         (dolist (function functions)
```

```
(autoload function file docstring interactive type))
t )))
```

2.3. [eval-after-autoload-func.el] 遅延読み込み

Twitter でぼやいていたら@[cymat](#) さんが降臨して次のマクロを作ってくださいました。感謝感謝。

<https://gist.github.com/3513287>

autoload-if-found で遅延読み込みすると、eval-after-load と組み合わせるので、どうしてもインデントが増えてしまう。

例えば、cycle-buffer を遅延読み込みしたい場合、setq で変数を書き換えするために随分とインデントが進んでいます。

```
(when (autoload-if-found
      '(cycle-buffer cycle-buffer-backward) "cycle-buffer" nil t)
  (eval-after-load "cycle-buffer"
    '(progn
      (setq cycle-buffer-allow-visible t)
      (setq cycle-buffer-show-length 12)
      (setq cycle-buffer-show-format '(" <(%s)>" . " %s")))))
```

これを、次の eval-after-autoload-func.el を使うと、非常にシンプルになる。行数も桁数もスッキリです。

```
(eval-after-autoload-if-found
  '(cycle-buffer cycle-buffer-backward) "cycle-buffer" nil t nil
  (setq cycle-buffer-allow-visible t)
  (setq cycle-buffer-show-length 12)
  (setq cycle-buffer-show-format '(" <(%s)>" . " %s")))
```

2.3.1. マクロ版

<https://gist.github.com/3499459>

```
(defmacro eval-after-autoload-if-found
  (functions file &optional docstring interactive type &rest after-body)
  "Set up autoload and eval-after-load for FUNCTIONS iff. FILE has found."
  `(let* ((functions ,functions)
          (docstring ,docstring)
          (interactive ,interactive)
          (type ,type)
          (file ,file))
    (when (locate-library file)
      (mapc (lambda (func)
              (autoload func file docstring interactive type))
            (if (listp functions)
                functions
                (list functions)))
      ,@(when after-body
          `((eval-after-load file '(progn ,@after-body))))
      t)))
```

2.3.2. 関数版

<https://gist.github.com/3513287>

```
(defun eval-after-autoload-if-found (functions file &optional docstring
interactive type after-body)
  "Set up autoload and eval-after-load for FUNCTIONS iff. FILE has found."
  (when (locate-library file)
    (mapc (lambda (func)
            (autoload func file docstring interactive type))
          (if (listp functions)
              functions
              (list functions))))
  (when after-body
    (eval-after-load file `(progn ,@after-body)))
  t))
```

2.4. パス設定

以下の2つ変数は、.emacs で=defconst=している。

```
(setq default-path "~/ .emacs.d/")
(setq default-private-path "~/ .emacs.d/")
```

2.5. 警告の抑制

起動時に警告が出てうっとうしい場合に使います。起動直後に呼ばれるように、.emacs の上の方を書いておくとよいと思います。

<http://d.hatena.ne.jp/kitokitoki/20100425/p1>

```
(setq byte-compile-warnings
      '(free-vars unresolved callargs redefine obsolete noruntime
        cl-functions interactive-only make-local))
```

3. コア設定

3.1. 言語／文字コード

徹底的に UTF-8 に合わせます。

save-buffer-coding-system を設定すると、buffer-file-coding-system の値を無視して、指定した save-buffer-coding-system の値でバッファを保存する。つまり、buffer-file-coding-system に統一するなら設定不要。

set-default-coding-systems か prefer-coding-system を設定すると、同時に file-name-coding-system=, =set-terminal-coding-system=, =set-keyboard-coding-system も同時に設定される。=prefer-coding-system= は、文字コード自動判定の最上位判定項目を設定する。

set-buffer-file-coding-system は、X とのデータやりとりを設定する。

```
(prefer-coding-system 'utf-8-unix)
(set-language-environment "Japanese")
(set-locale-environment "en_US.UTF-8") ; "ja_JP.UTF-8"
(set-default-coding-systems 'utf-8-unix)
(set-selection-coding-system 'utf-8-unix)
(set-buffer-file-coding-system 'utf-8-unix)

; (set-clipboard-coding-system 'utf-8) ; included by set-selection-coding-
```

```
system
; (set-keyboard-coding-system 'utf-8) ; configured by prefer-coding-system
; (set-terminal-coding-system 'utf-8) ; configured by prefer-coding-system
; (setq buffer-file-coding-system 'utf-8) ; utf-8-unix
; (setq save-buffer-coding-system 'utf-8-unix) ; nil
; (set-buffer-process-coding-system 'utf-8 'utf-8)
; (setq process-coding-system-alist
;       (cons '("grep" utf-8 . utf-8) process-coding-system-alist))
```

3.2. 日本語入力

Emacs23 用にインラインパッチを適用している場合に使います。Lion でも使える自分用にカスタマイズした [inline-patch](#) を使っています。

```
(when (and (eq window-system 'ns)
           (or (= emacs-major-version 23) (= emacs-major-version 24)))
  (setq default-input-method "MacOSX")
  (mac-add-key-passed-to-system 'shift))
```

3.3. 基本キーバインド

次の機能にキーバインドを設定する。

- Cmd+V でペースト (Mac 用)
- Cmd と Option を逆にする (Mac 用)
- 削除

```
(when (eq window-system 'ns)
  (global-set-key (kbd "M-v") 'yank)
  (setq ns-command-modifier 'meta)
  (setq ns-alternate-modifier 'super)
  (global-set-key [ns-drag-file] 'ns-find-file)) ; D&D for Emacs23
(global-set-key [delete] 'delete-char)
(global-set-key [kp-delete] 'delete-char)
```

3.4. ナローイングするか

ナローイングを有効にする。デフォルトは、ナローイングを知らないユーザが「データが消えた！」と勘違いしないように、無効になっている。

Org でナローイングを使う場合は、特に設定しなくてもよい。

```
(put 'narrow-to-region 'disabled nil)
```

3.5. バッファの終わりでのnewlineを禁止する

```
; Avoid adding a new line at the end of buffer
(setq next-line-add-newlines nil)
```

3.6. 常に最終行に一行追加する

```
; Limit the final word to a line break code (automatically correct)
(setq require-final-newline t)
```

3.7. 長い文章を右端で常に折り返す

```
(setq truncate-lines nil)
(setq truncate-partial-width-windows nil)
```

3.8. バッファが外部から編集された場合に自動で再読み込みする

```
(global-auto-revert-mode 1)
```

3.9. 同じバッファ名が開かれた場合に区別する

```
(require 'uniquify)
(setq uniquify-buffer-name-style 'post-forward-angle-brackets)
```

4. カーソル移動

カーソルの移動は、次のポリシーに従っています。デフォルトでは C-v/M-v で上下移動になっているが、M-v は windows のペーストに対応するので混乱を招く。

行移動	C-n/C-p
ページ移動（スクロール）	M-n/M-p
ウィンドウ移動	C-M-n/C-M-p

4.1. バッファ内の移動

先頭に移動，最終行に移動，ページ単位の進む，ページ単位の戻る，行数を指定して移動

```
(global-set-key (kbd "C-M-t") 'beginning-of-buffer)
(global-set-key (kbd "C-M-b") 'end-of-buffer)
;; Backward page scrolling instead of M-v
(global-set-key (kbd "M-p") 'scroll-down)
;; Frontward page scrolling instead of C-v
(global-set-key (kbd "M-n") 'scroll-up)
;; Move cursor to a specific line
(global-set-key (kbd "C-c g") 'goto-line)
```

4.2. バッファ間の移動

C-c o の代わりに、ウィンドウの移動をワンアクションで行う。

```
(global-set-key (kbd "C-M-p") '(lambda () (interactive) (other-window -1)))
(global-set-key (kbd "C-M-n") '(lambda () (interactive) (other-window 1)))
```

4.3. スクロールを制御

一行ずつスクロールする。デフォルトではバッファの端でスクロールすると、半画面移動する。また、上下の端にカーソルがどのくらい近づいたらスクロールとみなすかも指定できる。

<http://marigold.sakura.ne.jp/devel/emacs/scroll/index.html>

非 ASCII 文字を扱っているときに一行ずつスクロールしない場合は、scroll-conservatively の値を 1 では大きい数字にすると直るかもしれない。

<http://www.emacswiki.org/emacs/SmoothScrolling>

scroll-margin を指定すると、カーソルがウィンドウの端から離れた状態でスクロールされる。

```
;; Scroll window on a line-by-line basis
(setq scroll-conservatively 1000)
(setq scroll-step 1)
(setq scroll-margin 0) ; default=0
```

スクロール時のジャンプが気になる場合は次のパッケージを使うとよい。

<http://adamspiers.org/computing/elisp/smooth-scrolling.el>

```
(when (autoload-if-found
      '(smooth-scrolling) "smooth-scrolling" nil t)
  (eval-after-load "smooth-scrolling"
    '(progn
      (setq smooth-scroll-margin 1))))
(eval-after-autoload-if-found
  '(smooth-scrolling) "smooth-scrolling" nil t nil
  '((setq smooth-scroll-margin 1)))
```

4.4. スクロールで表示を重複させる行数

```
;; Scroll window on a page-by-page basis with N line overlapping
(setq next-screen-context-lines 1)
```

4.5. [SmoothScroll.el] カーソル固定でスクロールする

<https://raw.githubusercontent.com/takaxp/EmacsScripts/master/SmoothScroll.el>

<https://github.com/pglotov/EmacsScripts/blob/master/SmoothScroll.el>

カーソル位置と行を固定してバッファを背景スクロールできる。

オリジナルのままだとコンパイル時に警告がでるので、=line-move-visual= で書き換えて使っている。

```
(eval-after-autoload-if-found
  '(scroll-one-up scroll-one-down) "smoothscroll" nil t)
(autoload-if-found
  '(scroll-one-up scroll-one-down) "smoothscroll" nil t)
```

4.5.1. キーバインド

```
(global-set-key (kbd "s-<up>") 'scroll-one-down)
(global-set-key (kbd "s-<down>") 'scroll-one-up)
```

4.6. [point-undo.el] カーソル位置を簡単にたどる

autoload や autoload-if-found で定義すると、使いたい時に履歴が取れていないのでよろしくない。

```
(require 'point-undo nil t)
```

4.6.1. キーバインド

シングルキーを割り当てておくと使いやすい。

```
;; [point-undo.el] Move the cursor to the previous position
(global-set-key (kbd "<f7>") 'point-undo)
;; [point-undo.el] Redo of point-undo
(global-set-key (kbd "S-<f7>") 'point-redo)
```

4.7. [cycle-buffer.el] カレントバッファの表示切り替え

<http://www.emacswiki.org/emacs/download/cycle-buffer.el>

Cycle-buffer を使うと、バッファの履歴をスライドショーのようにたどれます。ミニバッファに前後の履歴が表示されるので、何回キーを押せばいいかの目安になります。それを超える場合には、おとなしくバッファリストを使います。直近数件のバッファをたどるのに便利です。

```
(eval-after-autoload-if-found
 '(cycle-buffer cycle-buffer-backward) "cycle-buffer" nil t nil
 '(setq cycle-buffer-allow-visible t)
 (setq cycle-buffer-show-length 12)
 (setq cycle-buffer-show-format '(" <(%s)>" . " %s"))))
```

4.7.1. キーバインド

```
(global-set-key (kbd "M-]") 'cycle-buffer)
(global-set-key (kbd "M-[") 'cycle-buffer-backward)
```

5. 編集サポート

5.1. 矩形編集

Built-in の cua-base.el (CUA-mode) を使う。

```
(cua-mode t)
(setq cua-enable-cua-keys nil)
```

矩形選択した後に、M-n を押すと、連番をふれる。開始値、増加値を入力してから、hoge%03d.pgm などとすれば、hoge001, hoge002, , , と入力される。これと、org-mode n の表機能 (C-c | で選択部分を簡単に表にできる) を組み合わせれば、連番で数値をふったテーブルを容易に作れる。

なお、標準の rect.el に以下の機能が実装されている。

矩形切り取り	C-x r k
矩形貼り付け	C-x r y
矩形先頭に文字を挿入	C-x r t
矩形を空白に変換する	C-x r c

5.2. Yank 時に装飾を取る

```
(setq yank-excluded-properties t)
```

5.3. ファイル保存時に時間を記録する

Built-in の time-stamp.el を使う.

バッファの保存時にタイムスタンプを記録する. 以下の設定では, バッファの先頭から 10 行以内に, "Last Update: " があると, "Last Update: 2011-12-31@12:00" のようにタイムスタンプが記録される.

```
(add-hook 'before-save-hook 'time-stamp)
(eval-after-load "time-stamp"
  '(progn
    (setq time-stamp-start "Last Update: ")
    (setq time-stamp-format "%04y-%02m-%02d@%02H:%02M")
    (setq time-stamp-end "$")
    (setq time-stamp-line-limit 10))) ; def=8
```

5.4. 選択リージョンを使って検索

検索語をミニバッファに入力するのが面倒なので, リージョンをそのまま検索語として利用します.

<http://dev.ariel-networks.com/articles/emacs/part5/>

```
(defadvice isearch-mode
  (around isearch-mode-default-string
    (forward &optional regexp op-fun recursive-edit word-p) activate)
  (if (and transient-mark-mode mark-active (not (eq (mark) (point))))
    (progn
      (isearch-update-ring (buffer-substring-no-properties (mark)
        (point)))
      (deactivate-mark)
      ad-do-it
      (if (not forward)
        (isearch-repeat-backward)
        (goto-char (mark))
        (isearch-repeat-forward)))
      ad-do-it))
```

5.5. ChangeLog モード

```
(setq user-full-name "Your NAME")
(setq user-mail-address "your@address.com")
(add-hook 'change-log-mode-hook
  '(lambda() (setq tab-width 4) (setq left-margin 4)))
```

5.6. テキストモード

<http://d.hatena.ne.jp/NeoCat/20080211>

```
(add-hook 'text-mode-hook
  '(lambda()
    (setq tab-width 4)
```

```
(setq tab-stop-list
      '(4 8 12 16 20 24 28 32 36 40 44 48 52 56 60
        64 68 72 76 80))
(setq indent-line-function 'tab-to-tab-stop)))
```

5.7. C/C++ モード

```
(setq auto-mode-alist
      (append '(("\\.h\\\\" . c++-mode)) auto-mode-alist))
```

5.8. HTML モード

```
(setq auto-mode-alist
      (append '(("\\.html\\\\" . html-helper-mode))
              auto-mode-alist))
```

5.9. Info モード

Org-mode の日本語翻訳済み info を読むための設定. [翻訳プロジェクト](#)で頒布しています.

```
(when (eval-after-autoload-if-found
      '(info) "info" nil t nil
      '((add-to-list 'Info-additional-directory-list
                    (expand-file-name "~/devel/mygit/org-
ja/work/")))))

(defun org-info-ja (&optional node)
  "(Japanese) Read documentation for Org-mode in the info system.
With optional NODE, go directly to that node."
  (interactive)
  (info (format "(org-ja)%s" (or node ""))))
```

5.10. スペルチェック

Built-in の ispell を使う. チェックエンジンは, aspell を利用する.

'ns	sudo port install aspell aspell-dict-en
-----	---

'x32	installer.exe and aspell-en from http://aspell.net/win32/
------	---

```
;;; Use aspell for spell checking instead of ispell.
(when (executable-find "aspell")
  (eval-after-autoload-if-found
    '(ispell-region) "ispell" nil t nil
    '((setq-default ispell-program-name "aspell")
      (when (eq window-system 'w32)
        (setq-default ispell-program-name
                      "C:/Program Files/Aspell/bin/aspell.exe"))
      ;;(setq ispell-grep-command "grep")
      ;; for English and Japanese mixed

      (add-to-list 'ispell-skip-region-alist '("[^\\000-\\377]"))
      (setq ispell-dictionary "english"))
```

```
(setq ispell-personal-dictionary
      (concat default-private-path ".aspell.en.pws"))

;; This will also avoid an IM-OFF issue for flyspell-mode.
;; (setq ispell-aspell-supports-utf8 t)
;; (setq ispell-encoding8-command t)
(setq ispell-local-dictionary-alist
      '((nil "[a-zA-Z]" "[^a-zA-Z]" "" t
             ("-d" "en" "--encoding=utf-8") nil utf-8)))))
```

5.10.1. キーバインド

```
;; Spell checking within a specified region
(global-set-key (kbd "C-c 0") 'ispell-region)
```

5.11. リアルタイムスペルチェック

Built-in の `flyspell.el` を使います.

重いので現在は使っていません。

<http://www.morishima.net/~naoto/fragments/archives/2005/12/20/flyspell/>

```
(dolist
  (hook
    '(text-mode-hook change-log-mode-hook c++-mode-hook
      latex-mode-hook org-mode-hook))
  (add-hook hook (lambda () (flyspell-mode 1)))))

(add-hook 'c++-mode-hook
  (lambda () (flyspell-prog-mode)))

;; Auto complete との衝突を回避
(ac-flyspell-workaround)
```

5.12. [latex-math-preview.el] TeX 数式をプレビュー

<http://www.emacswiki.org/emacs/latex-math-preview.el>

<http://transitive.info/software/latex-math-preview/>

```
(autoload 'latex-math-preview "latex-math-preview" nil t)
```

5.13. [po-mode.el] 翻訳ファイルの編集

<http://www.emacswiki.org/emacs/PoMode>

<http://www.emacswiki.org/emacs/po-mode+.el>

```
(; (autoload 'po-mode "po-mode+" nil nil)
(autoload 'po-mode "po-mode" nil t)
(setq auto-mode-alist
      (cons '("\\\\.po[tx]?\\'|\\\\|\\\\\\.po\\.\\. " . po-mode)
            auto-mode-alist))
```

5.14. [word-count.el] リージョン内の文字をカウントする

有効な頒布元に変更があった。 word-count.el から新しい頒布元にたどりつける。

```
(eval-after-autoload-if-found
 '(word-count-mode) "word-count" "Minor mode to count words." t)
```

5.14.1. キーバインド

```
(global-set-key (kbd "M-+") 'word-count-mode)
```

5.15. [yatex.el] YaTeX モード

```
(when (autoload-if-found 'yatex-mode "yatex" "Yet Another LaTeX mode" t)
  (setq auto-mode-alist
    (cons (cons "\\tex$" 'yatex-mode) auto-mode-alist))
  ;; Disable auto line break
  (add-hook 'yatex-mode-hook
    '(lambda ()
      (setq auto-fill-function nil)))
  (eval-after-load "yatex"
    '(progn
      ;; 1=Shift JIS, 2=JIS, 3=EUC, 4=UTF-8
      (setq YaTeX-kanji-code 4))))

(when (eval-after-autoload-if-found
  '(yatex-mode) "yatex" "Yet Another LaTeX mode" t nil
  '((setq YaTeX-kanji-code 4))) ;; 1=Shift JIS, 2=JIS, 3=EUC, 4=UTF-8

  (setq auto-mode-alist
    (cons (cons "\\tex$" 'yatex-mode) auto-mode-alist))
  ;; Disable auto line break
  (add-hook 'yatex-mode-hook
    '(lambda ()
      (setq auto-fill-function nil)))))
```

5.16. [wclock.el] 世界時計

<http://pastelwill.jp/wiki/doku.php?id=emacs>

```
(eval-after-autoload-if-found 'wclock "wclock" nil t)
```

5.17. [yasnippet.el] Emacs 用のテンプレートシステム

<https://github.com/capitaomorte/yasnippet>

- <http://yasnipet-doc-jp.googlecode.com/svn/trunk/doc-jp/index.html>
- <http://d.hatena.ne.jp/IMAKADO/20080401/1206715770>
- <http://coderepos.org/share/browser/config/yasnippet>
- <https://github.com/RickMoynihan/yasnippet-org-mode>

Org-mode との衝突を避ける

```
(defun yas-org-very-safe-expand ()
  (let ((yas-fallback-behavior 'return-nil)) (yas-expand)))
(when (eval-after-autoload-if-found
  '(yas-minor-mode yas-minor-mode-on yas-new-snippet)
  "yasnipet" nil t nil
  '((setq yas-verbosity 2))
```

```

        (setq yas-snippet-dirs
              '("~/Dropbox/emacs.d/yas-dict"
                "~/devel/git/yasnipet/snippets"))
        ;; (yas/initialize)
;;      (mapc 'yas-load-directory yas-snippet-dirs)
        ))

(dolist (hook (list 'perl-mode-hook 'c-mode-common-hook))
  (add-hook hook 'yas-minor-mode-on))
(add-hook 'emacs-lisp-mode-hook
  '(lambda () (unless (equal "*scratch*" (buffer-name))
                    (yas-minor-mode-on))))

(add-hook 'org-mode-hook
  '(lambda ()
    (yas-minor-mode-on)
    ;; org-cycle (<TAB>) との衝突を避ける
    (setq yas-trigger-symbol [tab])
    (add-to-list 'org-tab-first-hook 'yas-org-very-
safe-expand)
    (define-key yas-keymap [tab] 'yas-next-field))))

```

5.18. [sdic.el] 英辞郎で英単語を調べる

<http://www.namazu.org/~tsuchiya/sdic/index.html>

Emacs から辞書を使う。lookup を使う方法もあるが、Emacs から使うのは英辞郎に限定。

```

(when (autoload-if-found
      '(sdic-describe-word sdic-describe-word-at-point)
      "sdic" nil t)
  (eval-after-load "sdic"
    '(progn
      (setq sdic-face-color "#3333FF")
      (setq sdic-default-coding-system 'utf-8)
      ;; Dictionary (English => Japanese)
      (setq sdic-eiwa-dictionary-list
        '((sdicf-client "~/Dropbox/Dic/EIJIRO6/EIJI-128.sdic"))))
      ;; Dictionary (Japanese => English)
      (setq sdic-waei-dictionary-list
        '((sdicf-client "~/Dropbox/Dic/EIJIRO6/WAEI-128.sdic")))))

(eval-after-autoload-if-found
  '(sdic-describe-word sdic-describe-word-at-point) "sdic" nil t nil
  '((setq sdic-face-color "#3333FF")
    (setq sdic-default-coding-system 'utf-8)
    ;; Dictionary (English => Japanese)
    (setq sdic-eiwa-dictionary-list
      '((sdicf-client "~/Dropbox/Dic/EIJIRO6/EIJI-128.sdic"))))
    ;; Dictionary (Japanese => English)
    (setq sdic-waei-dictionary-list
      '((sdicf-client "~/Dropbox/Dic/EIJIRO6/WAEI-128.sdic")))))

```

5.18.1. キーバインド

```

;; カーソルの位置の英単語の意味を調べる
(global-set-key (kbd "C-M-w") 'sdic-describe-word-at-point)
;; ミニバッファに英単語を入れて英辞郎を使う
(global-set-key (kbd "C-c w") 'sdic-describe-word)

```

5.19. [lookup.el] 辞書

最近使っていません.

```
;; .lookup/cache.el
(setq lookup-init-directory "~/env/dot_files/.lookup")

(setq lookup-search-modules
  ' (("default"
      ("ndeb:/Users/taka/Dropbox/Dic/COBUILD5/cobuild" :priority t)
      ("ndeb:/Users/taka/Dropbox/Dic/COBUILD5/wordbank" :priority t)
      ("ndeb:/Users/taka/Dropbox/Dic/LDOCE4/ldoce4" :priority t)
      ("ndeb:/Users/taka/Dropbox/Dic/LDOCE4/bank" :priority t)
      ("ndeb:/Users/taka/Dropbox/Dic/LDOCE4/colloc" :priority t)
      ("ndeb:/Users/taka/Dropbox/Dic/LDOCE4/activ" :priority t))))

(setq lookup-agent-attributes
  ' (("ndeb:/Users/taka/Dropbox/Dic/COBUILD5"
      (dictionaries "cobuild" "wordbank"))
    ("ndeb:/Users/taka/Dropbox/Dic/LDOCE4"
      (dictionaries "ldoce4" "bank" "colloc" "activ"))))

(setq lookup-dictionary-attributes
  ' (("ndeb:/Users/taka/Dropbox/Dic/COBUILD5/cobuild"
      (title . "COBUILD 5th Edition")
      (methods exact prefix))
    ("ndeb:/Users/taka/Dropbox/Dic/COBUILD5/wordbank"
      (title . "Wordbank")
      (methods))
    ("ndeb:/Users/taka/Dropbox/Dic/LDOCE4/ldoce4"
      (title . "Longman 4th Edition")
      (methods exact prefix))
    ("ndeb:/Users/taka/Dropbox/Dic/LDOCE4/bank"
      (title . "LDOCE4 Examples and Phrases")
      (methods exact prefix menu))
    ("ndeb:/Users/taka/Dropbox/Dic/LDOCE4/colloc"
      (title . "LDOCE4 Collocation")
      (methods exact prefix))
    ("ndeb:/Users/taka/Dropbox/Dic/LDOCE4/activ"
      (title . "Longman Activator")
      (methods exact prefix menu))))

(setq lookup-default-dictionary-options
  ' ( (:stemmer . stem-english)))
(setq lookup-use-kakasi nil)

;;; lookup for dictionary (require EB Library, eblook, and lookup.el)
;;; package download: http://sourceforge.net/projects/lookup
;;; http://lookup.sourceforge.net/docs/ja/index.shtml#Top
;;; http://www.bookshelf.jp/texi/lookup/lookup-guide.html#SEC\_Top

"lookup-autoloads") ; for 1.99
; (load
; (autoload
; (autoload
; (autoload
; (autoload
; (autoload
'lookup "lookup" nil t)
'lookup-region "lookup" nil t)
'lookup-word "lookup" nil t)
'lookup-select-dictionaries "lookup" nil t)
```



```
;; Search Agents
;; ndeb option requires "eblook" command

; Use expand-
file-name!

; (setq
lookup-search-agents `((ndeb , (concat homedir "/Dropbox/Dic/COBUILD5"))
;
(ndeb , (concat homedir "/Dropbox/Dic/LDOCE4"))))
; (setq
lookup-use-bitmap nil)
; (setq ndeb-
program-name "/usr/bin/eblook")
; (when (eq
window-system 'ns)
; (setq
ndeb-program-name "/opt/local/bin/eblook")
; (setq
ndeb-program-arguments '("-q" "-e" "euc-jp"))
; (setq
ndeb-process-coding-system 'utf-8)) ; utf-8-hfs
```

5.19.1. キーバインド

```
(global-set-key (kbd "C-M-c") 'lookup-word)
```

5.20. [cacoo] Cacao で描く

画像をリサイズしてバッファに表示する用途にも使える。

```
(when (autoload-if-found 'toggle-cacoo-minor-mode "cacoo" nil t)
  (global-set-key (kbd "M--") 'toggle-cacoo-minor-mode)
  (eval-after-load "cacoo"
    '(progn
      (require 'cacoo-plugins))))
(when (eval-after-autoload-if-found
      '(toggle-cacoo-minor-mode) "cacoo" nil t nil
      '((require 'cacoo-plugins)))
  (global-set-key (kbd "M--") 'toggle-cacoo-minor-mode))
```

5.21. [iedit] バッファ内の同じ文字列を一度に編集する

[iedit.el](#) を使うと、バッファ内の同じ文字列を一度に編集することができる。部分重複のない変数名を置き換えるときに有用な場合がある。

```
(require 'iedit nil t)
```

6. 表示サポート

6.1. モードラインのモード名を短くする

```
(require 'mode-name-abbrev nil t)
```

6.2. モードラインの **Narrow** を短くする

標準では「Narrow」と表示されますが、「N」に短縮します。

```
(defvar my-narrow-display " N")
(setq mode-line-modes
  (mapcar (lambda (entry)
    (if (and (stringp entry)
      (string= entry "%n"))
      '(:eval (if (and (= 1 (point-min))
        (= (1+ (buffer-size)) (point-max))) ""
        my-narrow-display)) entry))
    mode-line-modes))
```

6.3. **visible-bell** のカスタマイズ

<http://www.emacswiki.org/emacs/MilesBader> を参考にカスタマイズしている。

visibl-bell を使うと、操作ミスで発生するビープ音を、視覚的な表示に入れ替えられる。しかし、デフォルトではバッファ中央に黒い四角が表示されて少々鬱陶しいので、ミニバッファの点滅に変更する

```
(when (autoload-if-found 'echo-area-bell "echo-area-bell" nil t)
  (setq visible-bell t)
  (setq ring-bell-function 'echo-area-bell))
(eval-after-autoload-if-found
  '(echo-area-bell) "echo-area-bell" nil t nil
  '((setq visible-bell t)
    (setq ring-bell-function 'echo-area-bell)))
```

6.4. 常に **scratch** を表示して起動する

```
;; Start Emacs with scratch buffer even though it call session.el/desktop.el
(add-hook 'emacs-startup-hook '(lambda () (switch-to-buffer "*scratch*")))
```

6.5. バッテリー情報をモードラインに表示する

```
;; Show battery information on the mode line.
(display-battery-mode t)
```

6.6. スクロールバーを非表示にする

スクロールバーを非表示にするには、nil を指定します。 右側に表示したい場合は、'right とします。

```
;; Show scroll bar or not
(set-scroll-bar-mode nil) ; 'right
```

6.7. ツールバーを非表示にする

ツールバーは使わないので非表示にします。

```
;; Disable to show the tool bar.
(tool-bar-mode 0)
```

6.8. 起動時のスプラッシュ画面を表示しない

```
;; Disable to show the splash window at startup
(setq inhibit-startup-screen t)
```

6.9. カーソル行の行数をモードラインに表示する

```
;; Show line number in the mode line.
(line-number-mode t)
```

6.10. カーソル行の関数名をモードラインに表示する

```
;; Show function name in the mode line.
(which-function-mode t)
```

6.11. 時刻をモードラインに表示する

```
;; Show clock in in the mode line
(display-time-mode t)
```

6.12. 対応するカッコをハイライトする

Built-in の paren.el が利用できる。拡張版として mic-paren.el があり、現在はこれを利用している。

```
(when (require 'mic-paren nil t)
  (paren-activate)
  (setq paren-sexp-mode nil)
  (set-face-foreground 'paren-face-match "#FFFFFF")
  ;; Deep blue: #6666CC, orange: #FFCC66
  (set-face-background 'paren-face-match "66CC66"))

(setq show-paren-delay 0)
(show-paren-mode t)
;; (setq show-paren-style 'expression) ; カッコ内も強調
;; (set-face-background 'show-paren-match-face "#5DA4ff") ; カーソルより濃い青
(set-face-background 'show-paren-match-face "#a634ff")
(set-face-foreground 'show-paren-match-face "#FFFFFF")
(set-face-underline-p 'show-paren-match-face nil)
(setq show-paren-style 'parenthesis)
```

6.13. [migemo.el] ローマ字入力で日本語を検索する

<http://0xcc.net/migemo/#download>

以下は、cmigemo を使う設定です。

```
(when (and (autoload-if-found 'migemo-init "migemo" nil t)
  (executable-find "cmigemo"))
  (add-hook 'isearch-mode-hook 'migemo-init)
  (eval-after-load "migemo"
    '(progn
      (setq completion-ignore-case t) ;; case-independent
      (setq migemo-command "cmigemo")
      (setq migemo-options '("-q" "--emacs" "-i" "\a"))
      (setq migemo-dictionary "/usr/local/share/migemo/utf-8/migemo-dict"))
```

```

(setq migemo-user-dictionary nil)
(setq migemo-regex-dictionary nil)
(setq migemo-use-pattern-alist t)
(setq migemo-use-frequent-pattern-alist t)
(setq migemo-pattern-alist-length 1024)
(setq migemo-coding-system 'utf-8-unix)))

(when
  (eval-after-autoload-if-found
    '(migemo-init) "migemo" nil t nil
    '((setq completion-ignore-case t) ;; case-independent
      (setq migemo-command "cmigemo")
      (setq migemo-options '("-q" "--emacs" "-i" "\a"))
      (setq migemo-dictionary "/usr/local/share/migemo/utf-8/migemo-dict")
      (setq migemo-user-dictionary nil)
      (setq migemo-regex-dictionary nil)
      (setq migemo-use-pattern-alist t)
      (setq migemo-use-frequent-pattern-alist t)
      (setq migemo-pattern-alist-length 1024)
      (setq migemo-coding-system 'utf-8-unix)))

  (when (executable-find "cmigemo")
    (add-hook 'isearch-mode-hook 'migemo-init)))

```

6.14. [anything.el] 何でも絞り込みインターフェイス

<http://svn.coderepos.org/share/lang/elisp/anything-c-moccur/trunk/anything-c-moccur.el> <http://d.hatena.ne.jp/IMAKADO/20080724/1216882563>

```

(when (autoload-if-found
  '(anything-other-buffer anything-complete
    anything-M-x anything-c-moccur-occur-by-moccur)
  "anything-startup" nil t)

  (defun my-anything ()
    (interactive)
    (anything-other-buffer
      '(anything-c-source-recentf
        anything-c-source-file-name-history
        anything-c-source-buffers
        anything-c-source-emacs-commands
        anything-c-source-locate)
      " *my-anything*"))

  (defun my-anything-buffer ()
    (interactive)
    (anything-other-buffer
      '(anything-c-source-buffers)
      " *my-anthing-buffer*"))

  (when (eq window-system 'ns)
    (defun my-anything-spotlight ()
      "Spotlight search with anything.el"
      (interactive)
      (anything-other-buffer
        '(anything-c-source-mac-spotlight)
        " *anything-spotlight*")))

  (eval-after-load "anything-startup"

```

```

' (progn

  (require 'anything-c-moccur nil t)
  ;; (setq moccur-split-word t)
  ;; (setq anything-c-locate-options `("locate" "-w"))

  ;; M-x install-elisp-from-emacswiki recentf-ext.el
  ;; http://www.emacswiki.org/cgi-bin/wiki/download/recentf-ext.el
  ;; (autoload-if-found 'recentf-ext "recentf-ext" nil t)
  (require 'recentf-ext nil t)

  (when (require 'migemo nil t)
    (setq moccur-use-migemo t))

  ;; M-x anything-grep-by-name
  (setq anything-grep-alist
    '(("Org-files" ("egrep -Hin %s *.org" "~/Dropbox/org/"))
      (".emacs.d" ("egrep -Hin %s *.el" "~/emacs.d/"))
      ("ChangeLog" ("egrep -Hin %s ChangeLog" "~/")))))
  ;; ("Spotlight" ("mdfind %s -onlyin ~/Dropbox/Documents/Library/" ""))))

  (setq anything-candidate-number-limit 50) ; 50
  (setq anything-input-idle-delay 0.1) ; 0.1
  (setq anything-idle-delay 0.5) ; 0.5
  (setq anything-quick-update nil))) ; nil

(when (eval-after-autoload-if-found
  '(anything-other-buffer anything-complete anything-M-x anything-c-
moccur-occur-by-moccur) "anything-startup" nil t nil
  '((require 'anything-c-moccur nil t)
    ;; (setq moccur-split-word t)
    ;; (setq anything-c-locate-options `("locate" "-w"))

    ;; M-x install-elisp-from-emacswiki recentf-ext.el
    ;; http://www.emacswiki.org/cgi-bin/wiki/download/recentf-ext.el
    ;; (autoload-if-found 'recentf-ext "recentf-ext" nil t)
    (require 'recentf-ext nil t)

    (when (require 'migemo nil t)
      (setq moccur-use-migemo t))

    ;; M-x anything-grep-by-name
    (setq anything-grep-alist
      '(("Org-files" ("egrep -Hin %s *.org" "~/Dropbox/org/"))
        (".emacs.d" ("egrep -Hin %s *.el" "~/emacs.d/"))
        ("ChangeLog" ("egrep -Hin %s ChangeLog" "~/"))))))))
  ;; ("Spotlight" ("mdfind %s -onlyin ~/Dropbox/Documents/Library/" ""))))

(defun my-anything ()
  (interactive)
  (anything-other-buffer
    ' (anything-c-source-recentf
      anything-c-source-file-name-history
      anything-c-source-buffers
      anything-c-source-emacs-commands
      anything-c-source-locate)
    " *my-anything*"))

(defun my-anything-buffer ()
  (interactive)

```

```

(anything-other-buffer
 ' (anything-c-source-buffers)
 " *my-anthing-buffer*"))

(when (eq window-system 'ns)
 (defun my-anything-spotlight ()
  "Spotlight search with anything.el"
  (interactive)
  (anything-other-buffer
   ' (anything-c-source-mac-spotlight)
   " *anything-spotlight*")))

(setq anything-candidate-number-limit 50) ; 50
(setq anything-input-idle-delay 0.1) ; 0.1
(setq anything-idle-delay 0.5) ; 0.5
(setq anything-quick-update nil) ; nil

```

6.14.1. キーバインド

普通に `anything-startup` を呼んでいる場合には、`anything-M-x` を設定する必要はない。

```

;; Show ibuffer powered by anything
(global-set-key (kbd "M-x") 'anything-M-x)
(global-set-key (kbd "C-c o") 'anything-c-moccur-occur-by-moccur)
(global-set-key (kbd "C-M-r") 'my-anything)
(global-set-key (kbd "C-M-s") 'my-anything-spotlight)
(global-set-key (kbd "C-x C-b") 'my-anything-buffer)

```

7. メディアサポート

7.1. [bongo.el] Emacs のバッファで音楽ライブラリを管理する

[iTunes の代わりに Emacs を使う](#)

`autoload` を設定すると、`*.bongo-playlist` や `*.bongo-library` から起動できないので、明示的に `require` している。なお、`bongo-mplayer` を使う場合、`bongo` を先に `require` するとうまく動作しない (`bongo.el` の最後で、`bongo-mplayer` が `provide` されているからだと思われる)。

以下の設定では、`autoload` で使いつつ、`=M-x init-bongo=` でプレイリストを読み込んでいる。これならば、Emacs 起動時は軽量で、かつ、プレイリストの訪問で `Bongo` を開始できる。

```

;; (require 'bongo)
(when (autoload-if-found 'bongo "bongo-mplayer" nil t)
 (defun init-bongo ()
  (interactive)
  (bongo)
  (find-file "~/Desktop/next/Tidy/hoge.bongo-playlist"))
 (eval-after-load "bongo-mplayer"
  '(progn
   ;; Volume control
   (require volume.el nil t)
   (setq bongo-mplayer-extra-arguments '("-volume" "1"))
   ;; Avoid error when editing bongo buffers

```

```

(setq yank-excluded-properties nil)
;; Use mplayer
(setq bongo-enabled-backends '(mplayer))))))

(when (eval-after-autoload-if-found
      '(bongo) "bongo-mplayer" nil t nil
      '(:; Volume control
        ;; (require volume.el nil t)
        (setq bongo-mplayer-extra-arguments '("-volume" "1"))
        ;; Avoid error when editing bongo buffers
        (setq yank-excluded-properties nil)
        ;; Use mplayer
        (setq bongo-enabled-backends '(mplayer))))

      (defun init-bongo ()
        (interactive)
        (bongo)
        (find-file "~/Desktop/next/Tidy/hoge.bongo-playlist")))

```

org-player.el を使えば、org-mode のバッファから Bongo を操作できる。

```
(eval-after-autoload-if-found 'org-mode "org-player" nil t)
```

音量コントロールには、[volume.el](#) が必要です。設定がうまくいかないので保留中

```
(autoload 'volume "volume" "Tweak your sound card volume." t)
```

7.2. [GoogleMaps.el] GoogleMaps を Emacs 内で使う

<http://julien.danjou.info/software/google-maps.el>

M-x gogole-maps で起動します。

```
(require 'google-maps nil t)
(require 'org-location-google-maps nil t)
```

+/- でズーム，矢印 で移動，q で終了します。また，w で URL を取得してコピー，t で地図の種別を変更できます。

Org-mode を使っている場合には，C-c M-L で表示されるプロンプトで検索すると，プロパティにそのキーワードが記録されます。後から C-c M-l すれば，いつでも地図を表示できるようになります。

7.3. [org-google-weather.el] org-agenda に天気を表示する

<http://julien.danjou.info/software/google-weather.el>

```
(require 'google-weather nil t)
(when (require 'org-google-weather nil t)
  '(org-google-weather-use-google-icons t))
```

8. 履歴／ファイル管理

8.1. Undo バッファを無限に取る

```
(setq undo-outer-limit nil)
```

8.2. バッファ保存時にバックアップファイルを生成する

バッファが保存されるとき、必ずバックアップを生成する。

```
;; Backup the buffer whenever the buffer is saved
(global-set-key (kbd "C-x C-s") '(lambda () (interactive) (save-buffer 16)))
```

8.3. ミニバッファの履歴を保存しリストアする

```
(savehist-mode 1)
```

8.4. 履歴サイズを大きくする

tで無限大に指定する。

```
(setq history-length 1000)
```

8.5. Emacs 終了時に開いていたバッファを起動時に復元する

Built-in の [desktop.el](#) を使う。

org バッファを CONTENT view で大量に開いていると、再起動が非常に遅くなるので利用を中止した。代替手段として、session.el と recentf の組み合わせがある。最近利用したファイルとそのカーソル位置が保持されるため、最後に訪問していたファイルを比較的簡単に復元できる。頻繁に復元するバッファには、別途キーバインドを割り当てておけば問題ない。

```
(when (autoload-if-found
      '(desktop-save desktop-clear desktop-load-default desktop-remove)
      "desktop" nil t)
  (desktop-save-mode 1)
  (setq desktop-files-not-to-save "\\(^/tmp\\|^/var\\|^/ssh:\\)"))
(eval-after-autoload-if-found
  '(desktop-save desktop-clear desktop-load-default desktop-remove)
  "desktop" nil t nil
  '((desktop-save-mode 1)
    (setq desktop-files-not-to-save "\\(^/tmp\\|^/var\\|^/ssh:\\)")))
```

8.6. 最近開いたファイルリストを保持

Built-in の [recentf.el](#) を使う。

<http://d.hatena.ne.jp/tomoya/20110217/1297928222>

session.el でも履歴管理できるが、anything のソースとして使っているので併用している。

起動直後から有効にするので、autolad-if-load で括る必要はない。

recentf-auto-cleanup を =mode= にすると起動時にファイルのクリーニングが行われるてしまうので、=never= で回避し、アイドルタイマーで対応する。これだけで50[ms]ほど起動を高速化できる。

```
(add-hook 'after-init-hook 'recentf-mode)
(eval-after-load "recentf"
  '(progn
```



```
(setq recentf-max-saved-items 2000)
(setq recentf-save-file (expand-file-name "~/Dropbox/.recentf"))
(setq recentf-auto-cleanup 'never) ; default = 'mode
(run-with-idle-timer 300 t 'recentf-save-list)
(run-with-idle-timer 600 t 'recentf-cleanup)
(setq recentf-exclude
  '("^/tmp\\.*" "^/private\\.*" "^/var/folders\\.*" "/TAGS$"))))
```

8.7. 深夜にバッファを自動整理する

<http://www.emacswiki.org/emacs-zh/CleanBufferList>

```
(when (require 'midnight nil t)
  (setq clean-buffer-list-buffer-names
        (append clean-buffer-list-kill-buffer-names
                  '("note.txt"))))
(setq clean-buffer-list-delay-general 1)
(setq clean-buffer-list-delay-special 10))
```

8.8. [auto-save-buffers.el] 一定間隔でバッファを保存する

<http://0xcc.net/misc/auto-save/>

起動直後から有効にするので、autolad-if-load で括る必要はない。

```
(when (require 'auto-save-buffers nil t)
  (run-with-idle-timer 1.0 t 'auto-save-buffers))
```

8.9. [backup-dir.el] バックアップファイルを一箇所に集める

<http://www.emacswiki.org/emacs/BackupDirectory> <http://www.northbound-train.com/emacs-hosted/backup-dir.el> <http://www.northbound-train.com/emacs.html>

起動直後から有効にするので、autolad-if-load で括る必要はない。

```
(make-variable-buffer-local 'backup-inhibited)
(when (and (require 'backup-dir nil t)
          (file-directory-p "~/env/emacs_backup")))
  ;; backup path
  (setq bkup-backup-directory-info '((t "~/env/emacs_backup" ok-create)))
  ;; generation properties
  (setq delete-old-versions t
        kept-old-versions 0
        kept-new-versions 5
        version-control t))
```

8.10. [session.el] 様々な履歴を保存し復元に利用する

<http://emacs-session.sourceforge.net/>

- 入力履歴の保持（検索語，表示したバッファ履歴）
- 保存時のカーソル位置の保持
- キルリングの保持
- 変更が加えられたファイル履歴の保持

M-x session-save-session

session-undo-check を指定していると、保存時ではなくバッファを閉じるときの状態を保持する。

Org-mode と併用する場合は, `my-org-reveal-session-jump` の設定が必須.

```
(when (autoload-if-found 'session-initialize "session" nil t)
  (add-hook 'after-init-hook 'session-initialize)
  (eval-after-load "session"
    '(progn
      (add-to-list 'session-globals-exclude 'org-mark-ring)
      ;; Change save point of session.el
      (setq session-save-file (expand-file-name "~/Dropbox/.session"))
      (setq session-initialize '(de-saveplace session keys menus places)
        session-globals-include '((kill-ring 100)
                                   (session-file-alist 100 t)
                                   (file-name-history 200)
                                   search-ring regexp-search-ring))
      (setq session-undo-check -1))))

;; FIXME
;; (setq session-set-file-name-exclude-regexp
;;       "^/private/\\.\\.\\.*")
;;       "[/\\\\]\\.overview\\\\\\\\[/\\\\]\\.session\\\\\\\\|News[/\\\\]\\\\\\\\|^/private\\.\\.\\.*\\\\\\\\
^/var/folders\\.\\.\\.*"))

(when (eval-after-autoload-if-found
  'session-initialize "session" nil t nil
  '((add-to-list 'session-globals-exclude 'org-mark-ring)
    ;; Change save point of session.el
    (setq session-save-file (expand-file-name "~/Dropbox/.session"))
    (setq session-initialize '(de-saveplace session keys menus
places)
      session-globals-include '((kill-ring 100)
                                (session-
file-alist 100 t)
                                (file-name-
history 200)
                                search-ring
regexp-search-ring))
    (setq session-undo-check -1)))

  (add-hook 'after-init-hook 'session-initialize))

;; FIXME
;; (setq session-set-file-name-exclude-regexp
;;       "^/private/\\.\\.\\.*")
;;       "[/\\\\]\\.overview\\\\\\\\[/\\\\]\\.session\\\\\\\\|News[/\\\\]\\\\\\\\|^/private\\.\\.\\.*\\\\\\\\
^/var/folders\\.\\.\\.*"))
```

次はテスト中. org バッファを開いたらカーソル位置を org-reveal したいが, timestamp などと組み合わせたり, org-tree-slide と組み合わせていると, うまくいかない. バッファを表示した時に org-reveal (C-c C-r) を打つのをサボりたいだけなのだが, . . .

<http://www.emacswiki.org/emacs/EmacsSession>

```
(when (autoload-if-found 'session-initialize "session" nil t)
  (add-hook 'after-init-hook 'session-initialize)
  (eval-after-load "session"
    '(progn
```

```
;; For Org-mode
(defun my-maybe-reveal ()
  (interactive)
  (when (and (or (memq major-mode '(org-mode outline-mode))
                 (and (boundp 'outline-minor-momminor-de)
                      outline-minor-mode))
            (outline-invisible-p))
    (if (eq major-mode 'org-mode)
        (org-reveal)
        (show-subtree))))

(defun my-org-reveal-session-jump ()
  (message "call!")
  (when (and (eq major-mode 'org-mode)
            (outline-invisible-p))
    (org-reveal)))

;; C-x C-/
(add-hook 'session-after-jump-to-last-change-hook
          'my-maybe-reveal)))
```

9. 開発サポート

9.1. 便利キーバインド

```
(global-set-key (kbd "C-;") 'comment-dwim) ;; M-; is the default
(global-set-key (kbd "C-c c") 'compile)
```

9.2. [gist.el] Gist インターフェイス

```
(eval-after-autoload-if-found '(gist) "gist" nil t)
```

9.3. [doxymacs.el] Doxygen のコメントを簡単に入力する

<http://doxymacs.sourceforge.net/>

```
(when (autoload-if-found 'doxymacs-mode "doxymacs" nil t)
  (add-hook 'c-mode-common-hook 'doxymacs-mode)
  (eval-after-load "doxymacs"
    '(progn
      (setq doxymacs-doxxygen-style "JavaDoc")
      (add-hook 'font-lock-mode-hook
        '(lambda () (interactive)
              (when (or (eq major-mode 'c-mode) (eq major-mode 'c++-mode))
                (doxymacs-font-lock))))
      (define-key doxymacs-mode-map (kbd "C-c C-s") 'ff-find-other-file))))
(when (eval-after-autoload-if-found
      'doxymacs-mode "doxymacs" nil t nil
      '((setq doxymacs-doxxygen-style "JavaDoc")
        (add-hook 'font-lock-mode-hook
          '(lambda () (interactive)
                (when (or (eq major-mode 'c-mode)
                          (eq major-mode 'c++-mode))
                  (doxymacs-font-lock))))
        (define-key doxymacs-mode-map (kbd "C-c C-s") 'ff-find-other-
```

```
file)))  
(add-hook 'c-mode-common-hook 'doxymacs-mode)
```

9.4. [matlab.el] Matlab 用の設定

```
(when (and (eq window-system 'ns) (= emacs-major-version 23))  
  (autoload 'matlab-mode "matlab" "Enter Matlab mode." t)  
  (setq auto-mode-alist (cons '("\\.m\\") . matlab-mode) auto-mode-alist))  
  (autoload 'matlab-shell "matlab" "Interactive Matlab mode." t))
```

9.5. [auto-complete.el] 自動補完機能

<http://cx4a.org/software/auto-complete/manual.ja.html>

- 辞書データを使う (ac-dictionary-directories)
- auto-complete.el, auto-complete-config.el, fuzzy.el, popup.el を使う.
- [日本語マニュアル](#)
- ac-auto-start を 4 にしておけば, 3 文字までは TAB を yasnippet に渡せる.

Org-mode ユーザにとって TAB は非常に重要なコマンド. そこに auto-complete と yasnippet が TAB を奪いに来るので, 住み分けが重要になる. =ac-auto-start= を =4=にすると, <s=TAB= によるソースブロックの短縮入力を yasnippet で実行できる (この目的だけならば=3=を指定してもいい). <sys などと 4 文字入力すると, =auto-complete= が動いて <system>などを補完してくれる. もちろん, 見出しで TAB を押すときには, ツリーの表示/非表示の切り替えになる.

情報源については, [オンラインマニュアル](#)を参照のこと.

```
(when (require 'auto-complete-config nil t)  
  (ac-config-default)  
  (defun ac-org-mode-setup ()  
;;      (message " >> ac-org-mode-setup")  
      (setq ac-sources '  
;;          ac-source-abbrev ; Emacs の略語  
;;          ;;          ac-source-css-property ; heavy  
;;              ac-source-dictionary ; 辞書  
;;              ac-source-features  
;;              ac-source-filename  
;;              ac-source-files-in-current-dir  
;;              ac-source-functions  
;;          ac-source-gtags  
;;          ac-source-imenu  
;;          ac-source-semantic  
;;          ac-source-symbols  
;;          ac-source-variables  
;;          ac-source-yasnippet  
          ))  
  (add-hook 'org-mode-hook 'ac-org-mode-setup)  
  (defun ac-default-setup ()  
;;      (message " >> ac-default-setup")  
      (setq ac-sources '(ac-source-abbrev  
                          ac-source-dictionary  
                          ac-source-words-in-same-mode-  
buffers)))
```

```

;      (setq ac-sources (append '(ac-source-abbrev
;                                ac-source-dictionary
;                                ac-source-words-in-same-mode-buffers)
;                                ac-sources)))
;      (dolist (hook (list 'perl-mode-hook 'objc-mode-hook))
;        (add-hook hook 'ac-default-setup))
;; *scratch* バッファでは無効化
(add-hook 'lisp-mode-hook
          '(lambda () (unless (equal "*scratch*"
(buffer-name))
                              (ac-default-setup))))

;; ac-modes にあるメジャーモードで有効にする
;; lisp, c, c++, java, perl, cperl, python, makefile, sh,
fortran, f90
(global-auto-complete-mode t)
;; 追加のメジャーモードを設定
(add-to-list 'ac-modes 'objc-mode)
(add-to-list 'ac-modes 'org-mode)
;; 辞書
(add-to-list 'ac-dictionary-directories (concat default-path
"ac-dict"))

;; n 文字以上で補完表示する ("<s TAB" の場合 yasnippet が呼ばれる)
(setq ac-auto-start 4)
;; n 秒後にメニューを表示
(setq ac-auto-show-menu 1.0)
;; ツールチップの表示
(setq ac-use-quick-help t)
(setq ac-quick-help-delay 2.0)
(setq ac-quick-help-height 10)
;; C-n/C-p でメニューをたどる
(setq ac-use-menu-map t)
;; TAB で補完 (org-mode でも効くようにする)
(define-key ac-completing-map [tab] 'ac-complete)
;; RET での補完を禁止
(define-key ac-completing-map "\r" nil)
;; 補完メニューの表示精度を高める
(setq popup-use-optimized-column-computation nil)
;; (setq ac-candidate-max 10)

```

9.6. [auto-complete-clang.el] オムニ補完

C++ バッファでメソッドを補完対象とする。try-catch を使っている場合、`-fcxx-exceptions` オプションが必要で、これはプリコンパイルヘッダを生成する時も同じだ。この設定では、`~/Dropbox/emacs.d/` 以下に `stdafx.pch` を生成する必要があるため、以下のコマンドを用いてプリコンパイルヘッダを生成する。ヘッダファイルのパスを適切に与えれば、Boost や自作のライブラリも補完対象に設定できる。

現状では、補完直後にデフォルトの引数がすべて書き込まれてしまう。なんかうまいことしたいものだ。

```
clang -cc1 -x c++-header -fcxx-exceptions ./stdafx.h -emit-pch -o ./stdafx.pch
-I/opt/local/include -I/opt/local/include/netpbm
```

以下の設定は、先に `auto-complete.el` に関する設定を読み込んでいることを前提としている。

```

(when (require 'auto-complete-clang nil t)
  ;; ac-cc-mode-setup のオーバーライド

```

```

      (defun ac-cc-mode-setup ()
;;      (message " >> Auto-complete-clang")
;;      (setq ac-clang-prefix-header "stdafx.pch")
;;      (setq ac-auto-start 0)
      (setq ac-clang-prefix-header "~/.emacs.d/stdafx.pch")
      (setq ac-clang-flags '("-w" "-ferror-limit" "1"
                             "-fcxx-exceptions"))
      (setq ac-sources '(ac-source-clang
                          ac-source-yasnippet
                          ac-source-gtags))
      )
      (add-hook 'c-mode-common-hook 'ac-cc-mode-setup))

```

次のコードを hoge.cpp として保存し, =v= と t について補完できれば, STL と Boost のプリコンパイルヘッダが有効になっていることを確認できる。

```

#include <iostream>
#include <vector>
#include <boost/timer.hpp>

int main() {
    std::vector<int> v;
    v; // ここ
    boost::timer t;
    cout << t; // ここ
    return 1;
}

```

9.6.1. 参考サイト

- <http://d.hatena.ne.jp/kenbell1988/20120428/1335609313>
- <http://d.hatena.ne.jp/whitypig/20110306/1299416655>
- <http://d.hatena.ne.jp/yano-htn/?of=30>
- <http://www.nomtats.com/2010/11/auto-completeelemacs.html>
- <http://www.plugmasters.com.br/plugfeed/post/73768/awesome-cc-autocompletion-in-emacs>

9.7. [hideshowvis.el] 関数の表示／非表示

<http://www.emacswiki.org/emacs/hideshowvis.el>

```

(when (and (eq window-system 'ns) (= emacs-major-version 23))
  (autoload 'hideshowvis-enable "hideshowvis" "Highlight foldable regions")
  (autoload 'hideshowvis-minor-mode "hideshowvis"
            "Will indicate regions foldable with hideshow in the fringe."
            'interactive)
  (add-hook 'emacs-lisp-mode-hook
            '(lambda () (unless (equal "*scratch*" (buffer-name))
                              (hideshowvis-enable))))
  (dolist (hook (list 'perl-mode-hook 'c-mode-common-hook))
    (add-hook hook 'hideshowvis-enable)))

```

9.7.1. キーバインド

Hide or show current block of sources

```
(global-set-key (kbd "C-(") 'hs-hide-block)
(global-set-key (kbd "C-)") 'hs-show-block)
```

10. Org Mode Thinking

10.1. 基本設定

```
(eval-after-autoload-if-found
'org-mode "org" "Org Mode" t nil
' (
;         (require 'org-install)
(require 'org-extension nil t)
(require 'org-habit)
(require 'org-mobile)

(setq auto-mode-alist
      (cons (cons "\\\\.org$" 'org-mode) auto-mode-alist))
(push '("\\\\.txt\\\\" . org-mode) auto-mode-alist)

;; Set checksum program path for windows
(when (eq window-system 'w32)
  (setq org-mobile-checksum-binary "~/Dropbox/do/cksum.exe"))

;; org ファイルの集中管理
(setq org-directory "~/Dropbox/org/")

;; Set default table export format
(setq org-table-export-default-format "orgtbl-to-csv")

;; Toggle inline images display at startup
(setq org-startup-with-inline-images t)

;; dvipng
(setq org-export-with-LaTeX-fragments t)

;; org バッファ内の全ての動的ブロックを保存直前に変更する
(add-hook 'before-save-hook 'org-update-all-dblocks)

;; アーカイブファイルの名称を指定
(setq org-archive-location "%s_archive::")

;; タイムスタンプによるログ収集設定
(setq org-log-done t) ; t ではなく, '(done), '(state) を指定できる

;; ログをドロアーに入れる
(setq org-log-into-drawer t)

;; タイマーの音
;; (lsetq org-clock-sound "");
))
```

10.2. contribution を使う

```
(setq load-path (append '("~/devel/taka/org-mode/contrib/lisp") load-path))
```

10.3. iCal との連携

```
(eval-after-autoload-if-found
'org-mode "org" "Org Mode" t nil
' (
  ;; ~/Dropbox/Public は第三者に探索される可能性があるので要注意
  ;; default = ~/org.ics
  ;; C-c C-e i org-export-icalendar-this-file
  ;; C-c C-e I org-export-icalendar-all-agenda-files
  ;; C-c C-e c org-export-icalendar-all-combine-agenda-files
  ;; (setq org-combined-agenda-icalendar-file "~/Dropbox/Public/orgAgenda.ics")

  ;; iCal の説明文
  (setq org-icalendar-combined-description "OrgMode のスケジュール出力")
  ;; カレンダーに適切なタイムゾーンを設定する (google 用には nil が必要)
  (setq org-icalendar-timezone "Asia/Tokyo")
  ;;; エクスポート後に, AppleScript で新しいカレンダーをリロードさせる
  ;; (add-hook 'org-after-save-icalendar-file-hook
  ;;          (lambda ()
  ;;            (shell-command
  ;;              "osascript -e 'tell application \"iCal\" to reload calendars'"))))
  (add-hook 'org-after-save-icalendar-file-hook
            (lambda ()
              (let ((result
                     (shell-command
                      "scp -o ConnectTimeout=5 ~/Dropbox/org/org-
ical.ics orz:~/public_html/ical"))))
                (if (eq result 0) (message "Uploading ... [DONE]")
                    (message "Uploading ... [MISS]"))))))

  ;; DONE になった TODO はアジェンダから除外する
  (setq org-icalendar-include-todo t)
  ;; (通常は, <>---<> で区間付き予定をつくる. 非改行入力で日付が Note に入らない)
  (setq org-icalendar-use-scheduled '(event-if-todo))
  ;;; DL 付きで終日予定にする: 締め切り日 (スタンプで時間を指定しないこと)
  ;;          (setq org-icalendar-use-deadline '(event-if-todo event-if-not-
todo))
  (setq org-icalendar-use-deadline '(event-if-todo))
))
```

10.4. スピードコマンド

```
(eval-after-autoload-if-found
'org-mode "org" "Org Mode" t nil
' ((setq org-use-speed-commands t)
   (setq org-speed-commands-user
         (quote (("n" . show-next-org)
                 ("t" . show-today-org)))))
  (defun show-next-org () (show-org-buffer "next.org"))
  (defun show-today-org () (show-org-buffer "today.org"))
))
```


10.5. ソースブロックの入力キーをカスタマイズ

```
(eval-after-autoload-if-found
'org-mode "org" "Org Mode" t nil
'((add-to-list 'org-structure-template-alist
              '("C" "#+BEGIN_COMMENT\n?\n#+END_COMMENT" ""))
  (add-to-list 'org-structure-template-alist
              '("S" "#+BEGIN_SRC emacs-lisp\n?\n#+END_SRC" "<src
lang=\"emacs-lisp\">\n\n</src>")))))
```

10.6. Pomodoro

<http://orgmode.org/worg/org-gtd-etc.html>

```
(eval-after-autoload-if-found
'org-mode "org" "Org Mode" t nil
'((
  (add-to-list 'org-modules 'org-timer)
  (setq org-timer-default-timer 25)
;; (add-hook 'org-clock-in-hook
;;           '(lambda ()
;;             (if (not org-timer-current-timer)
;;                 (org-timer-set-timer '(16))))))

  (setq growl-pomodoro-default-task-name "doing the task")
  (setq growl-pomodoro-task-name 'growl-pomodoro-default-task-name)

  (defun set-growl-pomodoro-task-name ()
    (interactive "P")
    (setq growl-pomodoro-task-name
      (read-from-minibuffer "Task Name: " growl-pomodoro-
default-task-name)))
  (add-hook 'org-timer-set-hook 'set-growl-pomodoro-task-name)

  (defun growl-pomodoro-timer ()
    (interactive)
    (shell-command-to-string
      (concat "growlnotify -s -a Emacs -t \"++ Pomodoro ++\" -m \""
        "The end of " growl-pomodoro-task-name "!\n"))
    (shell-command-to-string
      ; (concat "say The end of " growl-pomodoro-task-name)
      (concat "say -v Kyoko " growl-pomodoro-task-name)
    ))
  (add-hook 'org-timer-done-hook 'growl-pomodoro-timer)
))
```

10.7. face 関連

```
(eval-after-autoload-if-found
'org-mode "org" "Org Mode" t nil
'((
;; Font lock を使う
(global-font-lock-mode 1)
(add-hook 'org-mode-hook 'turn-on-font-lock)
;; ウィンドウの端で折り返す（想定と逆の振る舞い。どこかにバグがある）
(setq org-startup-truncated nil)
;; サブツリー以下の * を略式表示する
(setq org-hide-leading-stars t)
```

```
;; Color setting for TODO keywords
;; Color for priorities
;; (setq org-priority-faces
;;   '(("?A" :foreground "#E01B4C" :background "#FFFFFF" :weight bold)
;;     ("?B" :foreground "#1739BF" :background "#FFFFFF" :weight bold)
;;     ("?C" :foreground "#575757" :background "#FFFFFF" :weight bold)))
;; Color setting for Tags

;; #CC3333
(setq org-todo-keyword-faces
  '(("FOCUS" :foreground "#FF0000" :background "#FFCC66")
    ("CHECK" :foreground "#FF9900")
    ("ICAL" :foreground "#33CC66")
    ("WAIT" :foreground "#33CC66")
    ("EDIT" :foreground "#FF0033")
    ("MAIL" :foreground "#CC3300")
    ("PLAN" :foreground "#FF6600")
    ("REV1" :foreground "#3366FF")
    ("REV2" :foreground "#FFFFFF" :background "#3366FF")
    ("STOP" :foreground "#9999CC")))

;; (:foreground "#0000FF" :bold t) ; default. do NOT put this bottom
(setq org-tag-faces
  '(("Achievement" :foreground "#66CC66")
    ("Report" :foreground "#66CC66")
    ("Background" :foreground "#66CC99")
    ("Chore" :foreground "#6699CC")
    ("Domestic" :foreground "#6666CC")
    ("Ongoing" :foreground "#CC6666") ; for non
scheduled/reminder
    ("Repeat" :foreground "#CC9999") ; for interval tasks
    ("Mag" :foreground "#9966CC")
    ("buy" :foreground "#9966CC")
    ("note" :foreground "#6633CC")
    ("Implements" :foreground "#CC9999" :weight bold)
    ("Coding" :foreground "#CC9999")
    ("Editing" :foreground "#CC9999" :weight bold)
    ("Duty" :foreground "#CC9999" :weight bold)
    ("Survey" :foreground "#CC9999" :weight bold)
    ("Home" :foreground "#CC9999" :weight bold)
    ("Open" :foreground "#CC9999" :weight bold)
    ("Test" :foreground "#FF0000" :weight bold)
    ("DEBUG" :foreground "#FFFFFF" :background "#9966CC")
    ("EVENT" :foreground "#FFFFFF" :background "#9966CC")
    ("Thinking" :foreground "#FFFFFF" :background "#96A9FF")
    ("Schedule" :foreground "#FFFFFF" :background "#FF7D7D")
    ("OUTPUT" :foreground "#FFFFFF" :background "#66CC99")
    ("Log" :foreground "#008500"))))
;; #5BDF8D
```

10.8. TODO キーワードのカスタマイズ

キーワードには日本語も使えます。

```
(eval-after-autoload-if-found
 'org-mode "org" "Org Mode" t nil
 '(setq org-todo-keywords
  '(sequence "TODO(t)" "FOCUS(f)" "CHECK(C)" "|" "DONE(d)")
    (sequence "ICAL(c)" "EDIT(e)" "MAIL(m)" "PLAN(p)" "|"))
```

```

(sequence "REV1(r)" "REV2(R)" "WAIT(w)" "STOP(s)" "|"))

;; Global counting of TODO items
(setq org-hierarchical-todo-statistics nil)
;; Global counting of checked TODO items
(setq org-hierarchical-checkbox-statistics nil)

;; block-update-time
(defun org-dblock-write:block-update-time (params)
  (let ((fmt (or (plist-get params :format) "%Y-%m-%d")))
    (insert "" (format-time-string fmt (current-time)))))

;; すべてのチェックボックスの cookies を更新する
(defun do-org-update-statistics-cookies ()
  (interactive)
  (org-update-statistics-cookies 'all))
))

```

10.9. [org-agenda]

```

(eval-after-autoload-if-found
 'org-agenda "org" "Org Mode" t nil
 '(
  ;; Set the view span as day in an agenda view, the default is week
  (setq org-agenda-span 'day)
  ;; アジェンダに警告を表示する期間
  (setq org-deadline-warning-days 7)
  ;; アジェンダビューで FOLLOW を設定
  ;; (setq org-agenda-start-with-follow-mode t)
  ;; Customized Time Grid
  (setq org-agenda-time-grid
    '((daily today require-timed)
      "-----"
      (800 1000 1200 1400 1600 1800 2000 2200 2400 2600)))
  ;; アジェンダ作成対象（指定しないと agenda が生成されない）
  ;; ここを間違くと, MobileOrg, iCal export もうまくいかない
  (setq org-agenda-files
    '("~/Dropbox/org/org-ical.org" "~/Dropbox/org/next.org"
      "~/Dropbox/org/today.org"
      "~/Dropbox/org/work.org" "~/Dropbox/org/research.org"))))

```

10.10. [appt.el] アラーム設定

```

(eval-after-autoload-if-found
 'org-mode "org" "Org Mode" t nil
 '(
  ;; アラーム表示を有効にする
  (appt-activate 1)
  ;; window を フレーム内に表示する
  (setq appt-display-format 'window)
  ;; window を継続表示する時間[s]
  (setq appt-display-duration 3)
  ;; ビープ音の有無
  (setq appt-audible t)
  ;; 何分前から警告表示を開始するか[m]
  (setq appt-message-warning-time 3)
  ;; モードラインにアラームを表示する
  (setq appt-display-mode-line t)

```

```
;; org-agenda の内容をアラームに登録する
;; (org-agenda-to-appt t '((headline "TODO")))
;; 保存時にアラームに登録
;; (add-hook 'org-mode-hook
;;   (lambda () (add-hook 'before-save-hook
;;     'org-agenda-to-appt t '((headline "TODO")))))
;;
;;
))
```

10.11. [org-capture] 高速にメモを取る

```
(eval-after-autoload-if-found
'org-capture "org-capture" "Org Mode" t nil
' (
  ;; 2010-06-13 の形式では、タグとして認識されない
  (defun get-current-date-tags () (format-time-string "%Y%m%d"))
  (setq org-default-notes-file (concat org-directory "next.org"))
  (defvar org-capture-words-notes-file (concat org-directory "words.org"))
  (defvar org-capture-notes-file (concat org-directory "note.org"))
  (defvar org-capture-research-file (concat org-directory "research.org"))
  (defvar org-capture-buffer-file (concat org-directory "buffer.org"))
  (defvar org-capture-today-file (concat org-directory "today.org"))
  (defvar org-capture-ical-file (concat org-directory "org-ical.org"))

  ;; see org.pdf:p73
  (setq org-capture-templates
    `(("t" "TODO 項目を INBOX に貼り付ける" entry
      (file+headline nil "INBOX") "*** TODO %?\n\t")
      ("c" "同期カレンダーにエントリー" entry
        (file+headline ,org-capture-ical-file "Schedule")
        "*** TODO %?\n\t")
      ("d" "DRAFT 項目を INBOX に貼り付ける" entry
        (file+headline nil "INBOX") "*** DRAFT %?\n\t")
      ("l" "本日のチェックリスト" entry
        (file+headline ,org-capture-today-file "Today")
        "*** FOCUS 本日のチェックリスト %T\n (起床時間の記録)
[[http://www.hayaoki-seikatsu.com/users/takaxp/] [早起き日記]] \n (朝食) \n - [ ]
%?\n (昼食) \n (帰宅 / 夕食) \n ---- \n (研究速報) \n - [ ] \n")
      ("i" "アイデアを書き込む" entry (file+headline nil "INBOX")
        "*** %?\n - \n\t%U")
      ("b" "Bug タグ付きの TODO 項目を貼り付ける" entry
        (file+headline nil "INBOX")
        "*** TODO %? :bug:\n %i\n %a %t")
      ("w" , (concat "英単語を " org-capture-words-notes-file
        " に書き込む") entry
        (file+headline ,org-capture-words-
notes-file "WORDS")
        "*** %? :%(get-current-date-
tags):\n 「」 \n - ")
      ("g" , (concat "英語ノートを " org-capture-words-notes-file
        " に書き込む")
        entry (file+headline ,org-capture-words-notes-file
"GRAMMER")
        "*** %? :%(get-current-date-tags):\n\n%U")
      ("T" "時間付きエントリー" entry (file+headline nil "INBOX")
        "*** %? %T--\n")
      ("n" "ノートとして INBOX に貼り付ける" entry
        (file+headline nil "INBOX")
        "*** %? :note:\n\t%U")
      ("D" "「ドラッカー365の金言」をノートする" entry
```

```

        (file+headline ,org-capture-notes-file "The Daily Drucker")
        "*** [%?] \nDrucker) \n - \n - \nACTION POINT:\n -
\nQUESTION:\n - \n")
        ("r" , (concat "研究ノートを " org-capture-research-file
                        " に書き込む")
          entry (file+headline ,org-capture-research-file "Survey")
                "*** %? :note:\n# \n - \n\t%U")
          ("`" , (concat "ノートをバッファ " org-capture-buffer-file
                        " に書き込む")
            entry (file+headline ,org-capture-buffer-file "Buffer")
                  "*** %(get-random-string 16) %U\n\n%?\n\n----"))
    ))

```

10.12. [org-refile]

```

(eval-after-autoload-if-found
 'org-refile "org" "Org Mode" t nil
 '( (setq org-refile-targets
      (quote (("org-ical.org" :level . 1)
              ("next.org" :level . 1)
              ("sleep.org" :level . 1))))
    ))

```

10.13. [org-babel]

```

(eval-after-autoload-if-found
 'org-mode "org" "Org Mode" t nil
 '( (setq org-confirm-babel-evaluate nil)
      (setq org-src-fontify-natively t)
      (setq org-src-tab-acts-natively t)))

```

10.14. [MobileOrg] iOS との連携

<http://orgmode.org/manual/Setting-up-the-staging-area.html>

```

(eval-after-autoload-if-found
 'org-mode "org" "Org Mode" t nil
 '(
  ; (setq org-mobile-files '("~/Dropbox/org/next.org" "1.org" "2.org"))
  (setq org-mobile-files '("~/Dropbox/org/next.org"))
  ; (setq org-mobile-force-id-on-agenda-items nil)

  ;; Set a file to capture data from iOS devices
  (setq org-mobile-inbox-for-pull (concat org-directory "captured.org"))

  ; Upload location stored org files (index.org will be created)
  (setq org-mobile-directory "~/Dropbox/MobileOrg/")

  ;;; Menu to push or pull org files using MobileOrg
  (defun org-mobile-sync ()
    (interactive)
    (let
      (org-mobile-sync-type
       (read-from-minibuffer "How do you sync the org files? (pull or push)
"))
      (message "%s" org-mobile-sync-type)
      (cond
        ((string= "pull" org-mobile-sync-type) (org-mobile-pull))

```

```
((string= "push" org-mobile-sync-type) (org-mobile-push))))
))
```

10.15. [org-tree-slide.el] Org でプレゼンテーション

<http://pastelwill.jp/wiki/doku.php?id=emacs:org-tree-slide>

```
;; Org-tree-slide
(when (eval-after-autoload-if-found
      'org-tree-slide-mode "org-tree-slide" nil t nil
      ' ( ;; <f8>/<f9>/<f10>/<f11> are assigned to control org-tree-slide
          (define-key org-tree-slide-mode-map (kbd "<f9>")
            'org-tree-slide-move-previous-tree)
          (define-key org-tree-slide-mode-map (kbd "<f10>")
            'org-tree-slide-move-next-tree)
          (define-key org-tree-slide-mode-map (kbd "<f11>")
            'org-tree-slide-content)
          ;; reset the default setting
          (define-key org-tree-slide-mode-map (kbd "<left>") 'backward-
char)
          (define-key org-tree-slide-mode-map (kbd "<right>") 'forward-
char)
          (org-tree-slide-narrowing-control-profile)
          ;; (org-tree-slide-presentation-profile)
          (setq org-tree-slide-skip-outline-level 4)
          (setq org-tree-slide-skip-done nil)))
      (global-set-key (kbd "<f8>") 'org-tree-slide-mode)
      (global-set-key (kbd "S-<f8>") 'org-tree-slide-skip-done-toggle)))
```

10.16. [org-fstree] ディレクトリ構造を読み取る

```
(eval-after-autoload-if-found
  'org-mode "org" nil t nil
  ' ((require 'org-fstree nil t)))
```

10.17. [calfw-org] calfw に org の予定を表示する

```
(eval-after-autoload-if-found
  'cfw:open-org-calendar "calfw-org" "Rich calendar for org-mode" t nil
  ' (
    ;; icalendar との連結
    (setq cfw:org-icalendars '("~/Dropbox/org/org-ical.org"))

    ;; org で使う表にフェイスを統一
    (setq cfw:fchar-junction ?+
          cfw:fchar-vertical-line ?|
          cfw:fchar-horizontal-line ?-
          cfw:fchar-left-junction ?|
          cfw:fchar-right-junction ?|
          cfw:fchar-top-junction ?+
          cfw:fchar-top-left-corner ?|
          cfw:fchar-top-right-corner ?| )))

;; (add-hook 'window-configuration-change-hook 'cfw:resize-calendar)
;; (defun cfw:resize-calendar ())
;; (interactive)
```

```
;; (when (eq major-mode 'cfw:calendar-mode)
;; (cfw:refresh-calendar-buffer nil)
;; (message "Calendar resized.)))

;; (defun open-calfw-agenda-org ()
;; (interactive)
;; (cfw:open-org-calendar))

;; (setq org-agenda-custom-commands
;; '(("w" todo "FOCUS")
;; ("G" open-calfw-agenda-org "Graphical display in calfw"))))
```

10.18. [org-export-generic] エクスポート機能を拡張する

org-set-generic-type を使うことで、エクスポート機能を好みに拡張できる。
contrib の中の org-export-generic.el が必要なので注意する。

org-set-generic-type を .emacs に追記した後、C-c C-e g <key-binding>
とすればよい。<key-binding> は org-set-generic-type で設定する値である。2
つ目は、Markdown へのエクスポーターである。

```
(eval-after-autoload-if-found
'org-export-generic "org-mode" nil t nil
' (
  (org-set-generic-type
    "textile"
    ' (:file-suffix
      ".textile"
      :key-binding ?T
      :title-format "Title: %s\n\n"
      ;; :date-format "Date: %s\n"
      :date-export nil
      :toc-export nil
      :author-export nil
      :tags-export nil
      :drawers-export nil
      :date-export t
      :timestamps-export t
      :priorities-export nil
      :todo-keywords-export t
      :body-line-fixed-format "\t%s\n"
      ;;:body-list-
prefix "\n"
      :body-list-format "* %s"
      :body-list-suffix "\n"
      :body-bullet-list-prefix ("* " "** " "*** " "**** " "***** " "***** ")
      :body-number-list-format "# %s"
      :body-number-list-suffix "\n"
      :header-prefix (" " " " "### " "#### " "#####" " "##### ")
      :body-section-header-prefix ("h1. " "h2. " "h3. " "h4. " "h5. " "h6. ")
      :body-section-header-format "%s"
      :body-section-header-suffix ("\n\n")
      :body-header-section-numbers nil
      :body-header-section-number-format "%s) "
      :body-line-format "%s\n"
      :body-newline-paragraph "\n"
      :bold-format "%s*"
      :italic-format " %s "
```

```

:underline-format "+%s+"
:strikethrough-format "-%s-"
:verbatim-format "`%s`"
:code-format "@%s@"
:body-line-wrap 75
:blockquote-start "\n<pre>\n"
:blockquote-end "\n</pre>\n"
))

(org-set-generic-type
"markdown"
'(:file-suffix
".markdown"
:key-binding ?M
:title-format "Title: %s\n"
:date-format "Date: %s\n"
:toc-export nil
:author-export t
:tags-export nil
:drawers-export nil
:date-export t
:timestamps-export t
:priorities-export nil
:todo-keywords-export t
:body-line-fixed-format "\t%s\n"
;;:body-list-prefix "\n"
:body-list-format "- %s"
:body-list-suffix "\n"
:header-prefix (" " " " "### " "#### " "##### " "##### ")
:body-section-header-prefix (" " " " "### " "#### " "##### " "##### ")
:body-section-header-format "%s\n"
:body-section-header-suffix "(?= ?- " ")
:body-header-section-numbers nil
:body-header-section-number-format "%s) "
:body-line-format "%s\n"
:body-newline-paragraph "\n"
:bold-format "***%s**"
:italic-format "_%s_"
:verbatim-format "`%s`"
:code-format "@%s@"
:body-line-wrap 75
))
))

```

10.19. [org-odt] ODT 形式に出力

```

(eval-after-autoload-if-found
'(org-odt) "org-odt" nil t nil
'((setf org-export-odt-styles-file
      (concat (getenv "HOME") "/Dropbox/org/style.odt")))
  (setf org-export-odt-preferred-output-format "pdf")
  (setf org-export-odt-convert-processes
        '(("LibreOffice"
           "/Applications/LibreOffice.app/Contents/MacOS/soffice
--headless --convert-to %f%x --outdir %d %i")
          ("unoconv" "unoconv -f %f -o %d %i")))))

```


10.20. README を常に org-mode で開く

```
(eval-after-autoload-if-found
'org-mode "org" "Org Mode" t nil
'((push '("[rR][eE][aA][dD][mM][eE]" . org-mode) auto-mode-alist)))
```

10.21. その他

```
(eval-after-autoload-if-found
'org-mode "org" "Org Mode" t nil
'((setq alarm-table "~/Dropbox/org/today.org")
(run-at-time "00:00" nil 'set-alarms-from-file alarm-table)))
```

10.21.1. キーバインド

```
(when (eval-after-autoload-if-found
'org-mode "org" "Org Mode" t nil
'((when (require 'org-icalendar)
(defun my-org-export-icalendar ()
(interactive)
(org-export-icalendar nil "~/Dropbox/org/org-ical.org"))
;; (org-transpose-element) が割り当てられているので取り返す.
(org-defkey org-mode-map "\C-\M-t" 'beginning-of-buffer)

;; (define-key org-mode-map (kbd "C-c 1")
;; 'org-export-icalendar-combine-agenda-files)
(define-key org-mode-map (kbd "C-c 1") 'my-org-export-icalendar)
(define-key org-mode-map (kbd "C-c 2") 'do-org-update-statistics-
cookies)

(define-key org-mode-map (kbd "C-c m") 'org-mobile-sync)
(define-key org-mode-map (kbd "<f5>") 'org-narrow-to-subtree)
(define-key org-mode-map (kbd "S-<f5>") 'widen)))

(global-set-key (kbd "C-M-o") '(lambda () (interactive)
(show-org-buffer
"next.org"))))
(global-set-key (kbd "C-M-c") '(lambda () (interactive)
(show-org-buffer "org-
ical.org"))))
(global-set-key (kbd "C-M-9") '(lambda () (interactive)
(show-org-buffer
"buffer.org"))))
(global-set-key (kbd "C-M-0") '(lambda () (interactive)
(show-org-buffer
"today.org"))))
(global-set-key (kbd "C-c l") 'org-store-link)
(global-set-key (kbd "C-c a") 'org-agenda)
(global-set-key (kbd "C-c r") 'org-capture))
```

10.22. org-mode の latex エクスポート関数をオーバーライド

```
;;; Tex export (org-mode -> tex with beamer class) ;;;;;;;;;;;;;;;;;;;;;;;;;;
;; (setq org-export-latex-classes
;; '(("article"
;; "\documentclass[11pt]{article}
;; \usepackage[AUTO]{inputenc}
;; \usepackage[T1]{fontenc}
```

```

;; \usepackage{graphicx}
;; \usepackage{longtable}
;; \usepackage{float}
;; \usepackage{wrapfig}
;; \usepackage{soul}
;; \usepackage{amssymb}
;; \usepackage{hyperref}"
;;     ("\\section{%s}" . "\\section*{%s}")
;;     ("\\subsection{%s}" . "\\subsection*{%s}")
;;     ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
;;     ("\\paragraph{%s}" . "\\paragraph*{%s}")
;;     ("\\subparagraph{%s}" . "\\subparagraph*{%s}"))
;; ("report"
;;  "\\documentclass[11pt]{report}
;; \usepackage[AUTO]{inputenc}
;; \usepackage[T1]{fontenc}
;; \usepackage{graphicx}
;; \usepackage{longtable}
;; \usepackage{float}
;; \usepackage{wrapfig}
;; \usepackage{soul}
;; \usepackage{amssymb}
;; \usepackage{hyperref}"
;;     ("\\part{%s}" . "\\part*{%s}")
;;     ("\\chapter{%s}" . "\\chapter*{%s}")
;;     ("\\section{%s}" . "\\section*{%s}")
;;     ("\\subsection{%s}" . "\\subsection*{%s}")
;;     ("\\subsubsection{%s}" . "\\subsubsection*{%s}"))
;; ("book"
;;  "\\documentclass[11pt]{book}
;; \usepackage[AUTO]{inputenc}
;; \usepackage[T1]{fontenc}
;; \usepackage{graphicx}
;; \usepackage{longtable}
;; \usepackage{float}
;; \usepackage{wrapfig}
;; \usepackage{soul}
;; \usepackage{amssymb}
;; \usepackage{hyperref}"
;;     ("\\part{%s}" . "\\part*{%s}")
;;     ("\\chapter{%s}" . "\\chapter*{%s}")
;;     ("\\section{%s}" . "\\section*{%s}")
;;     ("\\subsection{%s}" . "\\subsection*{%s}")
;;     ("\\subsubsection{%s}" . "\\subsubsection*{%s}"))
;; ("beamer"
;;  "\\documentclass{beamer}
;; \usepackage[AUTO]{inputenc}
;; \usepackage{graphicx}
;; \usepackage{longtable}
;; \usepackage{float}
;; \usepackage{wrapfig}
;; \usepackage{amssymb}
;; \usepackage{hyperref}"
;;  org-beamer-sectioning)))

```

11. フォント／配色関連

11.1. 正規表現を見やすくする

```
(set-face-foreground 'font-lock-regexp-grouping-backslash "#66CC99")
(set-face-foreground 'font-lock-regexp-grouping-construct "#9966CC")
```

11.2. カーソル行に色をつける

```
;; Color of the current line
;; Cite: http://murakan.cocolog-nifty.com/blog/2009/01/emacs-tips-1d45.html
;; see also http://www.emacswiki.org/cgi-bin/emacs/highlight-current-line.el
(global-hl-line-mode t)
(set-face-background 'hl-line "#DEEDFF")
```

11.3. カーソルの色

```
;; Cursor (see also takaxp-mac.el)
;;(add-to-list 'default-frame-alist '(cursor-type . (hbar . 5)))
;;(add-to-list 'default-frame-alist '(cursor-type . bar))

;;(add-hook 'window-configuration-change-hook
(defun update-cursor-color ()
  (interactive)
  (if current-input-method (set-cursor-color "#91C3FF")
    (set-cursor-color "#AAAAAA")))
(update-cursor-color)
(run-with-idle-timer 10 t 'update-cursor-color)

(add-hook 'input-method-activate-hook
  (lambda () (set-cursor-color "#91C3FF")))
(add-hook 'input-method-inactivate-hook
  (lambda () (set-cursor-color "#AAAAAA")))

(when (and (eq window-system 'ns) (= emacs-major-version 23))
  ;; when IME is ON
  (mac-set-input-method-parameter
    "com.google.inputmethod.Japanese.base" 'title "G"))

(when (and (eq window-system 'ns) (= emacs-major-version 24))
  ;; when IME is ON
  (mac-set-input-method-parameter
    "com.google.inputmethod.Japanese.base" 'title "グ"))
```

11.4. カーソルを点滅させない

```
;; Disable cursor blink
(blink-cursor-mode -1)
```

11.5. フォント設定

表のような利用環境に対して、個別に設定を施しています。WindowsとLinuxは安定版のEmacs23で、Macは開発版のCocoaEmacs23です。MacではEmacs24でもうまく表示できています。

	ASCII	日本語
Mac	Monaco	ヒラギノ丸ゴ
Windows	Inconsolata	メイリオ
Linux	Inconsolata	MigMix

<http://d.hatena.ne.jp/setoryohei/20110117/1295336454>

```
(defun my-ja-font-setter (spec)
  (set-fontset-font nil 'japanese-jisx0208 spec)
  (set-fontset-font nil 'katakana-jisx0201 spec)
  (set-fontset-font nil 'japanese-jisx0212 spec)
  (set-fontset-font nil '(#x0080 . #x024F) spec)
  (set-fontset-font nil '(#x0370 . #x03FF) spec)
  (set-fontset-font nil 'mule-unicode-0100-24ff spec))

(defun my-ascii-font-setter (spec)
  (set-fontset-font nil 'ascii spec))

(cond
 ;; CocoaEmacs
 ((eq window-system 'ns)
  (when (or (= emacs-major-version 23) (= emacs-major-version 24))
    (let
      ;; 1) Monaco, Hiragino/Migu 2M : font-size=12, -apple-
      ;; 2) Inconsolata, Migu 2M      : font-size=14,
      ;; 3) Inconsolata, Hiragino     : font-size=14, -apple-
      hiragino=1.2
      hiragino=1.0
      ;; Fonts

      ((font-size 12)
       ; ((font-size 28) ; for mirroring presentation (1440x900)
       ; (ascii-font "Inconsolata")
       ; (ascii-font "Monaco")
       ; (ja-font "Migu 2M"))
       ;; (ja-font "Hiragino Maru Gothic Pro"))
      (my-ascii-font-setter (font-spec :family ascii-font :size font-
size))
      (my-ja-font-setter (font-spec :family ja-font :size font-size)))

      ;; Fix ratio provided by set-face-attribute for fonts display
      (setq face-font-rescale-alist
        '(("^-apple-hiragino.*" . 1.0) ; 1.2
          (".*Migu.*" . 1.2)
          (".*Inconsolata.*" 1.0)
          (".*osaka-bold.*" . 1.0)      ; 1.2
          (".*osaka-medium.*" . 1.0)    ; 1.0
          (".*courier-bold-.*-mac-roman" . 1.0) ; 0.9
          ;; (".*monaco cy-bold-.*-mac-cyrillic" . 1.0)
          ;; (".*monaco-bold-.*-mac-roman" . 1.0) ; 0.9
          ("cdac$" . 1.0))) ; 1.3

      ;; Space between lines
```

```

(set-default 'line-spacing 1)
;; Anti aliasing with Quartz 2D
(setq mac-allow-anti-aliasing t)))

((eq window-system 'w32) ; windows7
  (let
    ((font-size 14)
     (font-height 100)
     (ascii-font "Inconsolata")
     ;; (ja-font "Meiryo UI"))
     (ja-font "メイリオ"))
    (my-ja-font-setter
     (font-spec :family ja-font :size font-size :height font-height))
    (my-ascii-font-setter (font-spec :family ascii-font :size font-size)))
    (setq face-font-rescale-alist '((".*Inconsolata.*" . 1.0))) ; 0.9
    (set-default 'line-spacing 1))

(window-system ; for SuSE Linux 12.1
  (let
    ((font-size 14)
     (font-height 100)
     (ascii-font "Inconsolata")
     ;; (ja-font "MigMix 1M")
     (ja-font "Migu 1M"))
    (my-ja-font-setter
     (font-spec :family ja-font :size font-size :height font-height))
    (my-ascii-font-setter (font-spec :family ascii-font :size font-size)))
    (setq face-font-rescale-alist '((".*MigMix.*" . 2.0)
    (".*Inconsolata.*" . 1.0))) ; 0.9
    (set-default 'line-spacing 1)))

```

11.5.1. フォントのインストール方法

Linux では次のように処理するだけでよく、意外と簡単.

1. ~/.fonts を作成する
2. フォントを 1.のディレクトリに置く
3. fc-cache -fv を実行
4. fc-list でインストールされているかを確認.

なお, Windows では, フォントファイルを右クリックして, インストールを選択するだけで OK.

11.5.2. フォントチェック用コード

サンプルの [org ファイル](#) を作って, 見た目をチェックしています. バッファ内の桁数チェックや, ASCII が漢字の半分の幅になっているかのチェックが楽になります.

11.6. パッチをカラフルに表示する

Built-in の [diff-mode.el](#) をカスタマイズします.

現在試験中.

```
(setq ediff-window-setup-function 'ediff-setup-windows-plain)
```

<http://d.hatena.ne.jp/syohex/20111228/1325086893>

```
(eval-after-autoload-if-found
'diff-mode "diff-mode" nil t nil
'((set-face-attribute 'diff-added-face nil
                      :background nil :foreground "green"
                      :weight 'normal)
  (set-face-attribute 'diff-removed-face nil
                      :background nil :foreground "firebrick1"
                      :weight 'normal)

  (set-face-attribute 'diff-file-header-face nil
                      :background nil :weight 'extra-bold)

  (set-face-attribute 'diff-hunk-header-face nil
                      :foreground "chocolate4"
                      :background "white" :weight 'extra-bold
                      :inherit nil)))
```

11.7. 背景を黒系色にする

```
(custom-set-faces
'(default ((t
            (:background "black" :foreground "#55FF55")
            ))))
```

11.8. [rainbow-mode.el] 配色のリアルタイム確認

M-x rainbow-mode とすると、色指定のコードの背景色を、その指定色にリアルタイム変換してくれる。

<http://elpa.gnu.org/packages/rainbow-mode.html>

```
(eval-after-autoload-if-found 'rainbow-mode "rainbow-mode" nil t)
```

11.8.1. 色一覧

0, 6, 9, C, F の組み合わせ

```
#000000 #000033 #000066 #000099 #0000CC #0000FF
#003300 #003333 #003366 #003399 #0033CC #0033FF
#006600 #006633 #006666 #006699 #0066CC #0066FF
#009900 #009933 #009966 #009999 #0099CC #0099FF
#00CC00 #00CC33 #00CC66 #00CC99 #00CCCC #00CCFF
#00FF00 #00FF33 #00FF66 #00FF99 #00FFCC #00FFFF

#330000 #330033 #330066 #330099 #3300CC #3300FF
#333300 #333333 #333366 #333399 #3333CC #3333FF
#336600 #336633 #336666 #336699 #3366CC #3366FF
#339900 #339933 #339966 #339999 #3399CC #3399FF
#33CC00 #33CC33 #33CC66 #33CC99 #33CCCC #33CCFF
#33FF00 #33FF33 #33FF66 #33FF99 #33FFCC #33FFFF

#660000 #660033 #660066 #660099 #6600CC #6600FF
#663300 #663333 #663366 #663399 #6633CC #6633FF
#666600 #666633 #666666 #666699 #6666CC #6666FF
```

```
#669900 #669933 #669966 #669999 #6699CC #6699FF
#66CC00 #66CC33 #66CC66 #66CC99 #66CCCC #66CCFF
#66FF00 #66FF33 #66FF66 #66FF99 #66FFCC #66FFFF

#990000 #990033 #990066 #990099 #9900CC #9900FF
#993300 #993333 #993366 #993399 #9933CC #9933FF
#996600 #996633 #996666 #996699 #9966CC #9966FF
#999900 #999933 #999966 #999999 #9999CC #9999FF
#99CC00 #99CC33 #99CC66 #99CC99 #99CCCC #99CCFF
#99FF00 #99FF33 #99FF66 #99FF99 #99FFCC #99FFFF

#CC0000 #CC0033 #CC0066 #CC0099 #CC00CC #CC00FF
#CC3300 #CC3333 #CC3366 #CC3399 #CC33CC #CC33FF
#CC6600 #CC6633 #CC6666 #CC6699 #CC66CC #CC66FF
#CC9900 #CC9933 #CC9966 #CC9999 #CC99CC #CC99FF
#CCCC00 #CCCC33 #CCCC66 #CCCC99 #CCCCCC #CCCCFF
#CCFF00 #CCFF33 #CCFF66 #CCFF99 #CCFFCC #CCFFFF

#FF0000 #FF0033 #FF0066 #FF0099 #FF00CC #FF00FF
#FF3300 #FF3333 #FF3366 #FF3399 #FF33CC #FF33FF
#FF6600 #FF6633 #FF6666 #FF6699 #FF66CC #FF66FF
#FF9900 #FF9933 #FF9966 #FF9999 #FF99CC #FF99FF
#FFCC00 #FFCC33 #FFCC66 #FFCC99 #FFCCCC #FFCCFF
#FFFF00 #FFFF33 #FFFF66 #FFFF99 #FFFFCC #FFFFFF
```

12. Macintosh 専用

12.1. [igrep]

```
;; igrep (M-x grep Override)
;(when (require 'igrep nil t)
;  (igrep-define lgrep (igrep-use-zgrep nil) (igrep-regex-option "-n -Ou8"))
;  (igrep-find-define lgrep
;    (igrep-use-zgrep nil) (igrep-regex-option "-n -Ou8")))
;
```

13. フレーム/ウィンドウ制御

13.1. 起動時の設定

```
;; To avoid an error setting up the frame width (only for Emacs23)
;(set-frame-width (selected-frame) 81)
;(set-frame-width (selected-frame) 80)

;; Default window position to show a Emacs frame
;; Dynabook UX: top=0, left=0, width=80, height=32
(cond
  ((eq window-system 'ns) ; for Macintosh
   (setq initial-frame-alist
     (append
       '((top . 22) ; Y-pos from (0,0) the height of menu bar is 22pix.
         (left . 0) ; X-pos from (0,0) ; 420 is the center for MBP
         ;; 26 is the setting for Butler's Docklet
         ;; 837 is the setting for right side for MBP
         (width . 80) ; Width : character count
         (height . 35) ; Height : character count
         (alpha . (100 75))
```

```

        (vertical-scroll-bars . nil)
      ) initial-frame-alist)))

((eq window-system 'x) ; for Linux
 (setq initial-frame-alist
  (append
   '((vertical-scroll-bars . nil)
     (top . 0)
     (left . 0)
     (width . 80)
     (height . 38)
     ) initial-frame-alist)))

(t
  ; for Windows
  (setq initial-frame-alist
   (append
    '((vertical-scroll-bars . nil)
      (top . 0)
      (left . 0)
      (width . 80)
      (height . 26)
      ) initial-frame-alist))))

;; Apply the initial setting to default
(setq default-frame-alist initial-frame-alist)

```

13.2. [elscreen.el] Emacs バッファをタブ化

```

;;; ElScreen (require apal) ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Note: change a string in the elscreen.el from "mac" to "ns"
;;; 2011-10-26: e2wm's perspective (two) mode is more useful for me.
(load "elscreen" "ElScreen" t)

```

13.3. [e2wm.el] 二画面表示

1. <http://github.com/kiwanami/emacs-window-manager/raw/master/e2wm.el>
2. <http://github.com/kiwanami/emacs-window-layout/raw/master/window-layout.el>

```

(when (autoload-if-found 'e2wm:dp-two "e2wm" nil t)
  (eval-after-load "e2wm"
    '(progn
      (setq e2wm:c-two-recipe
        '(- (:lower-size 10)
          (| left right)
          sub))
      (setq e2wm:c-two-wininfo
        '(:name left )
          (:name right )
          (:name sub :default-hide t)))
      ;; left, prev
      (setq e2wm:c-two-right-default 'left)

      ;; To avoid rebooting issue when using desktop.el and recentf.el
      (add-hook 'kill-emacs-hook 'e2wm:stop-management))))

```



```
(eval-after-autoload-if-found
'e2wm:dp-two "e2wm" nil t nil
'((setq e2wm:c-two-recipe
      '(- (:lower-size 10)
          (| left right)
          sub))
 (setq e2wm:c-two-wininfo
      '(:name left )
        (:name right )
        (:name sub :default-hide t)))
;; left, prev
(setq e2wm:c-two-right-default 'left)

;; To avoid rebooting issue when using desktop.el and recentf.el
(add-hook 'kill-emacs-hook 'e2wm:stop-management)))
```

13.4. [frame-ctr.el] キーボードでフレームの場所を移す

拙作の [frame-ctr.el](#) を使います. e2wm.el と frame-cmds.el に依存しています.

frame-ctr.el は, frame-cmds, frame-fns と e2wm.el を利用したアドホックなフレーム管理ツールです.

1. <http://www.emacswiki.org/emacs/download/frame-cmds.el>
2. <http://www.emacswiki.org/emacs/download/frame-fns.el>

```
(eval-after-autoload-if-found
'(change-frame-width-single
  change-frame-width-double
  frame-ctr-open-height-ring
  move-frame-with-user-specify move-frame-left move-frame-to-center
  move-frame-right move-frame-to-edge-top move-frame-to-edge-bottom)
"frame-ctr" nil t nil
'((frame-ctr-make-height-ring '(56 20 40))) ; for Emacs24
;; (frame-ctr-make-height-ring '(60 68 20 40))) ; for Emacs23
```

13.4.1. キーバインド

```
;; Move the frame to somewhere (default: 0,0)
(global-set-key (kbd "M-0") 'move-frame-with-user-specify)
;; Move the frame to left side of the current position (require 'frame-cmds)
(global-set-key (kbd "M-1") '(lambda () (interactive) (move-frame-left 200)))
;; Move the frame to the center of the window display (require 'frame-ctr)
(global-set-key (kbd "M-2") 'move-frame-to-center)
;; Move the frame to right side of the current position (require 'frame-cmds)
(global-set-key (kbd "M-3") '(lambda () (interactive) (move-frame-right 200)))
;; Set the frame width single size
;; C-u C-x - => e2wm OFF, single size width and double height, move center
(global-set-key (kbd "C-x -") 'change-frame-width-single)
;; Set the frame width double size
;; C-u C-x = => e2wm ON, double size width and height, move to the center
(global-set-key (kbd "C-x =") 'change-frame-width-double)
;; Move the current frame to the top of the window display
(global-set-key (kbd "<f1>") 'move-frame-to-edge-top)
;; Move the current frame to the bottom of the window display
(global-set-key (kbd "S-<f1>") 'move-frame-to-edge-bottom)
;; Cycle heights
(global-set-key (kbd "<f2>") 'frame-ctr-open-height-ring)
```

13.5. [popwin.el] ポップアップウィンドウの制御

<https://github.com/m2ym/popwin-el/>

popwin:display-buffer を autoload してもうまくいかない。

```
(when (require 'popwin nil t)
  (setq special-display-function 'popwin:special-display-popup-window)
  (setq display-buffer-function 'popwin:display-buffer)
  (setq popwin:special-display-config
    (append
      '(("*Completions*" :height 10 :position bottom :noselect t)
        ("CAPTURE-next.org" :height 10 :position bottom :noselect t)
        ("CAPTURE-org-ical.org" :height 10 :position bottom :noselect
t)
          ("*Org-todo*" :height 10 :position bottom)
          ("*Calendar*" :height 10 :position bottom)
          ("*wclock*" :height 10 :position bottom)
          ("*Org Agenda*" :height 10 :position bottom)
          ("*Agenda Commands*" :height 10 :position bottom)
          ("*Org Select*" :height 10 :position bottom)
          ("*Occur*" :height 10 :position bottom)
          ("*sdc*" :height 10 :position top)
          ("*anything*" :height 10 :position bottom)
          ("*anything M-x*" :height 10 :position bottom)
          ("*anything complete*" :height 10 :position bottom)
          ("*my-anything*" :height 10 :position bottom)
          ("*my-anything-buffer*" :height 10 :position bottom)
          ;; ("*cfw-calendar*" :height 40 :position top)
          ("*eshell*" :height 10 :position bottom))
      popwin:special-display-config)))
```

14. ユーティリティ関数

14.1. [pomodoro.el] ポモドーロの実践

@syohex さん謹製の [pomodoro.el](#) に少しカスタマイズしたおれおれ [pomodoro.el](#) を使っています。以下のように設定すると、ポモドーロの残り時間は表示せず、アイコンだけをモードラインに表示できます。残り時間は M-x pomodoro:mode-line-time-display-toggle すれば、いつでも表示できます。

pomodoro:finish-work-hook , pomodoro:finish-rest-hook ,
pomodoro:long-rest-hook にそれぞれ結びつけてあるのは、[Macのスピーチ機能](#)です。この例では、Kyoko さんが指示を出してくれます。

M-x pomodoro:start すると、ポモドーロが始まり、8時間後に pomodoro:stop が呼ばれてポモドーロが終了します。[pomodoro](#) は機械的に仕事をしたい人にピッタリです。人によっては [GTD](#) よりも取っ付きやすいと思います。

```
(eval-after-autoload-if-found
 'pomodoro:start "pomodoro" nil t nil
 '(:; 作業時間終了後に開くファイルを指定しない
  (setq pomodoro:file nil)

  ;; ●だけで表現する（残り時間表示なし）
  (setq pomodoro:mode-line-time-display nil)
```

```

;; 長い休憩に入るまでにポモドーロする回数
(setq pomodoro:iteration-for-long-rest 8)

;; 作業時間関連
(setq pomodoro:work-time 25      ; 作業時間
      pomodoro:rest-time 5       ; 休憩時間
      pomodoro:long-rest-time 60 ; 長い休憩時間
      pomodoro:max-iteration 16) ; ポモドーロする回数

;; タイマーの表示をノーマルフェイスにする
(set-face-bold-p 'pomodoro:timer-face nil)

;; 作業中（赤），休憩中（青），長い休憩中（緑）にする
(set-face-foreground 'pomodoro:work-face "#F53838")
(set-face-foreground 'pomodoro:rest-face "#3869FA")
(set-face-foreground 'pomodoro:long-rest-face "#00B800")

(defvar my-pomodoro-speak nil)
(defun my-pomodoro-speak-toggle ()
  (interactive)
  (setq my-pomodoro-speak (not my-pomodoro-speak)))

;; Mac ユーザ向け. Kyoko さんに指示してもらう
(add-hook 'pomodoro:finish-work-hook
  (lambda ()
    (let ((script (concat "say -v Kyoko "
                          (number-to-string (floor
pomodoro:rest-time))
                          "分間、休憩しろ"))))
      (if my-pomodoro-speak
          (shell-command-to-string script)
          (message "%s" script)))))

(add-hook 'pomodoro:finish-rest-hook
  (lambda ()
    (let ((script (concat "say -v Kyoko "
                          (number-to-string (floor
pomodoro:work-time))
                          "分間、作業しろ"))))
      (if my-pomodoro-speak
          (shell-command-to-string script)
          (message "%s" script)))))

(add-hook 'pomodoro:long-rest-hook
  (lambda ()
    (let ((script (concat "say -v Kyoko これから"
                          (number-to-string (floor
pomodoro:long-rest-time))
                          "分間の休憩です"))))
      (if my-pomodoro-speak
          (shell-command-to-string script)
          (message "%s" script))))))

```

14.2. [utility.el] 自作してテスト中の便利関数群

関数定義を[別ファイル](#)に分離して，Emacs 起動の高速化を図っています．各関数を autoload の管理下において，必要なときにロードするように設定しています．

```

(when (eval-after-autoload-if-found
      '(eval-org-buffer
        kyoko-mad-mode-toggle org2dokuwiki-cp-kill-ring
        open-current-directory set-alarms-from-file takexp:open-file-ring
        show-org-buffer get-random-string init-auto-install
        add-itemize-head insert-formatted-current-date
        insert-formatted-current-time insert-formatted-signature
        export-timeline-business export-timeline-private reload-ical-
export
      browse-url-chrome count-words-buffer do-test-applescript
      my-window-resizer)
      "utility" nil t)

(global-set-key (kbd "<f12>") 'takexp:open-file-ring)
(global-set-key (kbd "M-4") 'my-window-resizer))

;; Editing with a rectangle region
(global-set-key (kbd "C-x r C-SPC") 'rm-set-mark)
(global-set-key (kbd "C-x r C-x") 'rm-exchange-point-and-mark)
(global-set-key (kbd "C-x r C-w") 'rm-kill-region)
(global-set-key (kbd "C-x r M-w") 'rm-kill-ring-save)

```

15. provide

```
(provide 'init)
```