

## 秀丸からEmacsにエディタ環境を移行するためのチートシート

このページでは、秀丸の色々な機能をEmacsではどのように実現するかを記します。

最終更新日：2011年02月21日 (月) 23時02分20秒

### 前置き

#### 秀丸は良いソフトです

秀丸は言わずと知れた有名エディタで、開発業務を行っている人にとってなくてはならないアプリです（若干の誇張有り）。強力かつ高速な編集機能、かゆいところに手が届く機能、などなど。これがなければ皆の生産性は半分以下に落ちてしまうと思います（若干の誇張有り）。使いこなせば使いこなすほど馴染んでくる、というカスタマイズ性の高さも素晴らしい。その辺はおあつらえのIDEじゃ味わえないです。もうね、開発者だったらどんなことがあっても入れて使いこなせるようになっておくべき、とも思うくらい。

バージョンが上がるごとに「この機能がほしかった」というツボな機能が入ってくるところもね。開発者のことを大切にしてくれているな、という気持ちにさせてくれる「分かっている」アプリなのです。

#### でもね・・・

いいところだらけの秀丸ですが、「Windowsでしか動かない\*1」という最大かつ致命的な欠点を持っているソフトでもあります。

僕の場合、職場はWindows、自宅ではMacを使っているわけで、つまり自宅での生産性が半分以下に落ちてしまっているわけです（かなりの誇張有り）。とはいっても帰宅してからもWindowsの汚い画面なんか見たくないので、しょうがないです。え、Vista? UACが嫌いだから却下。

趣味で色々プログラミングをしていると、自宅でも職場と同じような効率で作業したいという心境になってきてしまいます。でもMacに秀丸はない。しばらく悶々としていました。で、色々試したり考えたりした結果、同じように「馴染み方が半端でない」といわれるEmacsを使っていくことに決めました。理由は、

- EmacsならLinuxもWindowsもMacでも同じ操作ができる
- Xcodeは便利だが、Mac専用なので根本的な解決になっていない（覚える量が2倍になるだけ）
- Emacsは「カスタマイズできる環境」なので、自分の好きなようにできる
- viと違いマウスで操作できる

です。

というわけでこのページを使って、環境の引っ越しに向けて秀丸の色々な機能をEmacsでどのように実現するかをまとめることにしたので、

### 注意

EmacsのターゲットはCarbon Emacsとします。つまり、ここでは特別なもの以外はすでにEmacsに組み込まれているとみなし、導入方法を説明したりはしません。あくまで代替手段のコマンド一覧を示すのみです。

## 目次

- まとめる前に
  - よく使っている機能
  - ゴールの設定
- 秀丸のそれ、Emacsではどうやるの？
  - ないと困る機能
    - tagsファイルによるジャンプ
    - Grep
    - 関数一覧表示
    - キー入力の記録&再生
    - 検索履歴による再検索
    - 最後に編集した箇所に移動
    - 検索方向の切り替え
    - キー入力によるファイルの切り替え
    - 文字コード表示&保存
    - マクロ外コードのグレイアウト
    - ファイルを開いた時に、以前編集していた場所に移動
    - 日付の入力
    - 一括コメントアウト
  - 気になる点
    - タブ文字の入力
    - 行番号の表示
  - あったらいいな
    - diff+Mergeを、日本語表示に対応させた上で実現
    - 入力しようとしている名前を補完
    - 選択領域をハイライトまたは反転させる
    - Shift + 矢印キーで領域選択する
    - 矩形領域の選択
  - その他
    - マウスのホイールスクロールを加速させたくない
    - 「最近使ったファイル」をメニューに表示させる
    - 文字列の自動補完
    - Subversionクライアント
    - 対応する括弧に移動
- 補足
  - 外部からダウンロードした.elファイルをCarbonEmacsにインストールする方法()
    - CarbonEmacsアプリ下に保存（一番手っ取り早いが将来性がない）
    - .emacs.elでパスを指定し、そこに保存する（オススメ）
- 参考にした情報
  - Special Thanks
  - 書籍

。 インターネット

国内

▪ 海外

• コメント

**まとめる前に**

さて、まとめ資料を作る前に、自分が秀丸のどういう機能を使っているかを洗い出します。

**よく使っている機能**

まず「この機能はないと困る！」という機能の一覧を考えます。秀丸で馴染んでしまった操作であり外せない機能の一覧です。ただEmacsでどこまで代用できるか分からないので、優先度も付けてみました。

重要度	機能	説明
必須	tagsファイルによるジャンプ	C/C++ファイルで、現在のカーソルが示す関数の実態へとジャンプする。バックジャンプも可能
必須	Grep	複数のソースファイルからキーワードにマッチするキーワードを抜き出して、一覧表示する。ここから更にダイレクトジャンプ&バックジャンプができること
必須	関数一覧表示	ファイル内に記載されたCやC++などの関数一覧を別ダイアログorウィンドウで表示しておき、今はどの関数にいるか、マウスクリックによる関数先頭への移動が行える
高	キー入力の記録&再生	あるキー入力をトグルに動作を記録し、その後再生できる
高	検索履歴による再検索	以前使った検索キーワードを使って、再検索する。ダイアログボックスから履歴を選択できる。これにより置換する前に検索で正規表現を試し、実際に使用、というテストファーストな編集ができる
高	最後に編集した箇所に移動	最後にキー入力した箇所にカーソルを移動する。編集中、別の場所でコピーをしてから本機能で編集中の場所に戻ってペースト、というのが非常に簡単。Wordだとコピーしたときの場所を最後の編集箇所として認識されてしまうが、それは違う！わかってない！
中	検索方向の切り替え	検索方向をキー1つで切り替えられる
中	キー入力によるファイルの切り替え	キー入力によって、編集対象のファイルを切り替える機能。どのファイルを開いているかもすぐに分かる
中	文字コード表示&保存	開いているファイルの文字コードを常に表示しておく。保存する際も文字コードと改行コードを指定できる
中	マクロ外コードのグレイアウト	#if 0や未定義のマクロで括られた部分をグレイアウトする。グレイアウトするかどうかは別途定義しておく必要があるものの、大規模な年老いたソースを読む場合には必須

中	前回編集箇所への移動	ファイルを開いた時に、以前編集していた場所に移動する
低	日付の入力	カーソルの位置に今日の日付を入力する。秀丸ではマクロで実現
低	一括コメントアウト	選択した領域のコメントアウト。秀丸ではマクロで実現

一方でEmacsを少しかじってみると、Emacsのデフォルト設定では次のような振る舞いが気になります。これも直せないか調べてみようと思います。

気になる度	機能	気になる点（どうなってほしいか）
中	タブ文字の入力	Tabキーを入力したときのインデントが空白文字で行われる。これはこれで綺麗だけれど、他のWindows開発者に合わせて、タブ文字でタブを入力したいときもある
中	行番号の表示	Emacsでは行番号をウィンドウ左側に表示できない？
中	選択領域の明示	選択領域をハイライトまたは反転させる。どこが選択領域の最初かなんて覚えていません。
中	PCライクな操作	Shift + 矢印キーで領域選択する。記憶のすみでは[始点選択] [終点選択]という前時代的なインターフェースだった記憶があるので。
中	矩形選択	矩形領域の選択。Emacsで出来るのは知っているけど、入力キーが面倒だし・・・

さらに欲目を出して、こんなことが出来たらという点も挙げてみましょう。

ほしさ度	機能	説明
高	diff	Diff+Mergeを、日本語表示に対応させた上で実現する
中	ソースコード補完	tagsファイルなどから、入力しようとしている関数名を補完する機能

でも高望みはしません。まずは秀丸でよく使う機能をまとめてから、です。

## ゴールの設定

上記の一覧表から、次をゴールに設定します。

- それぞれの表で出た機能を実現する方法を示す
  - 秀丸でよく使う機能は必ず探し出す
  - 気になる点も必ず
  - できたらいいなはできるだけ
- Emacsのキーバインドをできるだけ維持する
  - きっと秀丸と同じ操作性にすることもできるんですが、それだと何か違うと思う

## 秀丸のそれ、Emacsではどうやるの？

### ないと困る機能

#### tagsファイルによるジャンプ

etagsでTAGSファイルを作成して、そして・・・とか考えていたけれど、どうやらgtagsというのがヨサゲ、GNU GLOBALと言う、いわばetagsやctagsの高機能版（統合版？）のようなもの、BerkleyDBに関数や変数の定義を登録して、情報を出力してくれるみたい。

注意：gtagsといってもGoogle tagsのことではないので気を付けて（間違えないと思うけどね）。

インストールは簡単、MacPortsを使って、

```
$ sudo port install global
```

とするだけ。

そうすると「/opt/local/var/macports/software/global/5.6.2\_0/opt/local/share/gtags」ディレクトリの下に「gtags.el」が置かれているので、これをCarbon Emacsのsite-lispディレクトリにコピー。あとは.emacs.elに

```
;; GNU global (gtags) の設定
(global-locate-library "gtags") (require 'gtags))
(global-set-key "\M-t" 'gtags-find-tag) ;関数の定義元へ
(global-set-key "\M-r" 'gtags-find-rtag) ;関数の参照先へ
(global-set-key "\M-s" 'gtags-find-symbol) ;変数の定義元/参照先へ
(global-set-key "\M-p" 'gtags-find-pattern)
(global-set-key "\M-f" 'gtags-find-file) ;ファイルにジャンプ
(global-set-key [?\C-,] 'gtags-pop-stack) ;前のバッファに戻る
(add-hook 'c-mode-common-hook
  '(lambda ()
    (gtags-mode 1)
    (gtags-make-complete-list)))
```

と記述しておく。キーバインドは上記のコマンドを参照。

GNU GLOBALの便利なところは、更新したぶんだけTAGファイルを作り直してくれるところ。だから2回目以降は速い、速い。

```
$ global -u
```

とするだけです。

ただ、現時点ではなぜかcppファイルで「extern "C" {...}」内に定義した関数をタグ(GTAGS)として登録してくれない\*2。なぜだろう?? .globalrcと各コマンドのmanページを読んでみたけど分からない...

2008/10/29追記

どうやら既知のバグらしいです。MacPortsでのバージョンが5.6.2だったので、Portfileをゴニョゴニョして最新版の5.7.2(2008/9/30リリース)を試してみたのですが、やはり駄目でした。

2008/11/12追記：

このバグを直すパッチが採用されたようです。将来のリリースでは問題なくなりそうなので嬉しい。GLOBAL最強。

## Grep

これを選ぶのは悩みました。最初はgrep-findをしていたけれど、どうも使いづらい（遠回りに作業をしている）感じがしてダメで、しばらくは[color-moccur](#)を使っていました。

```
;; color-moccurの利用宣言と、その設定
;; (moccur-editもインストールしました)
;; Ref. URL: http://www.bookshelf.jp/soft/meadow_50.html#SEC740
;; - .svnは対象外とする
;; - カーソル付近の単語をデフォルトの検索語にする
;; - 複数の単語を、記入順序に関係なく検索する
(load "color-moccur")
(setq dmoccur-recursive-search t)
(setq moccur-grep-default-word-near-point t)
(setq moccur-split-word t)
(setq dmoccur-exclusion-mask (append ' (" \\ ~$" " \\ .svn \\ / \\ *" " \\ .o$"
                                       "GPATH" "GRTAGS" "GSYMS" "GTAGS")
                                     dmoccur-exclusion-mask))
```

だけど、除外ファイルも一度検索リストに入れられてしまうので遅いという、けっこう重大な欠点があったので次第に使わなくなってしまいました。

で、紆余曲折のすえに現在使っているのは以下の機能。

まずMacPortsで"p5-app-ack"をインストールしてackコマンドが使えるようにしておきます。そして、それと[color-grep.el](#)を併用して、

- grepを高速に。しかも.svnなどの管理ファイルは最初から無視してくれる
- grep結果バッファでカーソルを移動するだけで別バッファに該当ファイルの行を表示してくれる

という機能を満たすことができました。

```
;; grep結果バッファでのカーソル移動でダイナミックにファイルを開いてくれる
(require 'color-grep)
(setq color-grep-sync-kill-buffer t)
;; M-x grep-findでPerlのackコマンドを使うよう変更
(setq grep-find-command "ack --nocolor --nogroup ")
```

あとは上記機能にmoccur-grep-default-word-near-pointに相当する、コマンド実行時のカーソル近辺の単語を自動入力してくれる機能があれば完璧です。改造して、作れないものだろうか。

2008/11/12追記：

「コマンド実行時のカーソル近辺～」が自作できました。emacs.elに以下の定義を追加して、M-x grep-by-ackまたはF6キーで実行できるようにしました。

```
;; M-x grep-by-ack
;; Perlのackコマンドを使ったgrep(カーソル付近の単語をデフォルトの検索語に)
(defun grep-by-ack ()
  "grep the whole directory for something defaults to term at cursor position"
  (interactive)
  (setq default-word (thing-at-point 'symbol))
  (setq needle1 (or (read-string (concat "ack for <" default-word ">: ")) default-word))
  (setq needle1 (if (equal needle1 "") default-word needle1))
  (setq default-dir default-directory)
  (setq needle2 (or (read-string (concat "target directory <" default-dir ">: ")) default-dir))
  (setq needle2 (if (equal needle2 "") default-dir needle2))
  (grep-find (concat "ack --nocolor --nogroup " needle1 " " needle2)))
```

これでOK, 実行すると

1. ファイル名 (デフォルトはカーソル上の単語)
2. 検索パス (デフォルトはコマンドを実行したバッファのカレントディレクトリ)

を順番に聞いてきます。

便利、便利、ちなみに参考にしたのは米Slashdotの[こちら](#)です。

## 関数一覧表示

[M-x speedbar]

これ、予想以上でした。VisualStudioやEclipseのようなソースツリービューを表示してくれる。[f11]などの最高位のキーに設定してしまう価値は十分にあります。

と思ったら、[Emacs Code Browser](#)などというすごそうなElispも発見。Carbon Emacsには含まれていないので調査が必要かな・・・なんて思ったけれど、[スクリーンショット](#)を見る限りちょっとやり過ぎな感があったので触らないことにした。

2009/04/15追記：[Googleから「nav」というのが出ていた><http://google-opensource.blogspot.com/2009/03/nav-ide-like-navigation-pane-for-emacs.html>]。プロジェクトホームにあるスクリーンショット(下に転載)を見てみると、speedbarと異なり同じウィンドウ内でファイルを表示してくれるのが何だかヨサゲ。早速インストールしてみた。起動は「M-x nav」。

うーん・・・ダメだこれ。ファイルを表示してくれるだけで関数一覧は表示してくれない。1キーで色々できるのは良いんだけど、ちょっとボクの目的からは外れているのでパス。

## キー入力の記録&再生

記録開始 : [F3]or[C-x (] / 記録終了 : [F4]or[C-x )] / 再生 : [F4]or[C-x e]

[F3]や[F4]はEmacs 22より古いバージョンでは使用できないとのこと。  
Emacsではマクロと呼ぶ。直前のマクロを編集する場合は[C-x C-k e]と入力してからリターン([C-x e]も可)。編集後は[C-c C-c]で確定。マクロを保存して再利用するのは若干面倒で、

1. マクロに名前を付ける
  1. [C-x C-k n]でマクロ名を付加 (セッションを閉じるまでは[M-x マクロ名]で実行可能)
2. 名前を付けたマクロをファイルに展開、保存する
  1. .emacsが適当なファイルにて[M-x insert-kbd-macro Enter マクロ名 Enter]と入力し、マクロをelispで展開
  2. ファイルを保存。 .emacs以外のファイルに保存した場合は、.emacsにて「(load-file "パス付きのファイル名")」を記述しておく
3. [M-x マクロ名]で実行

という手順を踏む。キー割当も行いたい場合は、同じく.emacs上で「(global-set-key "\C-x\C-kT" マクロ名)」と記述しておく(Tの場合)。

ちなみにマクロに割り当てられるキーは、[C-x C-k 0~9またはA~Z]のみなので要注意。

## 検索履歴による再検索

検索やファイルオープンの際に[M-p]を押すだけ

[イヤなブログ著者のサイト](#)にある、minibuf-isearch.elやsession.elが使えるかどうか調査したりもしたのですが、そんなことをしなくても大丈夫でした。

## 最後に編集した箇所へ移動

とりあえず簡単な方法は、[C-] [C-g] [C-]でUndoしてからRedoで戻せば何とかなるけれど、一度操作を取り消すのがちょっとリスク。

そこで色々検索したら[goto-chg.el](#)というEmacsLispを発見。ファイルを開いてから変更した箇所へ移動できる(しかも辿っていける)とのこと。

ただ、同サイトでのキーアサインはGNU Globalとバッティングしてしまうので、普段の自分のアサイン[F5](逆方向へ辿る時はShift+[F5])を試してみました。

```
;; [F5]で最後に編集した箇所へ辿る
;; Shift-[F5]で編集箇所を逆方向に辿る
;; http://www.emacswiki.org/cgi-bin/wiki/goto-chg.el
(require 'goto-chg)
(global-set-key [f5] 'goto-last-change)
(global-set-key [S-f5] 'goto-last-change-reverse)
```

そしたら、おお、まさに秀丸のと同じかそれ以上の機能が！素晴らしい！

## 検索方向の切り替え

順方向 : [C-s]、逆方向 : [C-r] かみむらさんにM-sではなくC-rだと指摘してもらいました。THX!  
単語検索 : [C-s Enter C-w] ただし、次の候補を探しにいけない！

ただ、この検索方法だと大文字小文字の区別無く、単語かどうかの判定もできません。秀丸だと検索用ダイアログボックスのチェックボックスで、

- 大文字/小文字を区別する
- 単語として検索
- 正規表現の適用

を切り替えられる(検索している途中でも)ので、これだけだとちょっと不便。調査中。

と、ここで問題発生。単語として検索したとき、いったん検索すると次候補に[C-s C-s]してしまうと普通の検索モードになってしまう。この回避方法はないみたいで、さてどうしたものか。

- - 2chで聞いたところ、「ないよ」とか「そんなあなたにmoccure」「Redoは？」と答をくれた。Redo, Redo+はダメだったけど、moccureは調べれば何とかかなりそう？ [moccure関連全部入りのcolor-moccureはこちら](#)。

2008/8/6：単語検索を連続する方法に気が付いた。なんだ、正規表現で単語を表せばいいんだ。

```
単語検索：[C-M-s] \ <hogehoge \ >
検索を続けるとき：[C-s], [C-r]
```

気付けば簡単でした。

ちなみにインクリメンタルサーチ中にバックスペースキーを入力するとメイン画面の文字が削除されてしまうのは、以下の設定を.emacs.elに記述しておけば良いことが判明。

```
;; backspaceキーをインクリメンタルサーチ中のミニバッファで有効にする
(define-key isearch-mode-map [backspace] 'isearch-delete-char)
```

参考にしたのは[こちらのサイト](#)。ありがとうございました。

## キー入力によるファイルの切り替え

```
[C-x C- ]または[C-x C- ] (矢印は矢印キーを表す)
```

[C-x b]や[C-x C-b]もいいけれど、一番手軽。ちなみにウィンドウやフレーム間の移動は、

```
[C-x 2]などで作成したウィンドウ間の移動：[C-x o]
[C-x 5 2]などで作成したフレーム間の移動：[C-x 5 o]
```

と入力する。・・・けれど、フレーム間の移動はちょっと面倒くさい。そこで調べてみたところ(other-frame x)でxに渡す値でアクティブなフレームを切り替えてくれることが判明。逆戻りは負数を与える。

というわけで、[C-x C- ][C-x C- ]に登録してみた。

```
;; C-x <C-left/right>でバッファの切り替え
;; C-x <C-up/down>でフレームの切り替え
(global-set-key [?\ C-x C-up] '(lambda () "" (interactive) (other-frame -1)))
(global-set-key [?\ C-x C-down] '(lambda () "" (interactive) (other-frame 1)))
```

あ、これ便利だ。

## 文字コード表示 & 保存

どうやるの？

## マクロ外コードのグレイアウト

どうやるの？

## ファイルを開いた時に、以前編集していた場所へ移動

以下を.emacs.elに記述するだけです。saveplace.elはCarbonEmacsに最初から含まれています。

```
(load "saveplace")
(setq-default save-place t)
```

これによって「最後にカーソルのあった場所 \*3」に移動してくれます。

## 日付の入力

どうやるのだろう？ EmacsLispで簡単にできそうな気がするけれど、

2008/10/29現在、ATOKをインストールしてしまったので「きょう」や「いま」を変換すれば日付や時刻が入力できるようになってしまいました。

## 一括コメントアウト

```
事前に範囲を指定しておいて[M-;] (セミコロン) かみむらさんTHX!
```

```
範囲指定していない場合はカーソルのある行の末尾にコメントが追加され、そこにカーソルが移動して
```

このM-;は結構インテリジェントで、範囲選択していない状態では行末にコメントを追加し、その真ん中にカーソルを持ってってくれる。便利！

## 気になる点

### タブ文字の入力

[M-i]でハードタブ（空白記号ではない、`\t`）が入力される。

ただしバックスペースで削除する時はタブ文字から空白になって消されていくし、また自動インデントの種類によってはタブではなく2文字の空白で表されてしまうので、本格的にタブだけで使用するには結構なカスタマイズが必要っぽい。

2008/9/9追記：

空白になって消されていくのは、Backspaceキーにbackward-delete-char-untabifyというように「非タブ化しつつバックスペース」というコマンドがバインドされていたことが原因だった。なので

```
(global-set-key [backspace] 'backward-delete-char)
```

と.emacs.elに記述して問題解決。

### 行番号の表示

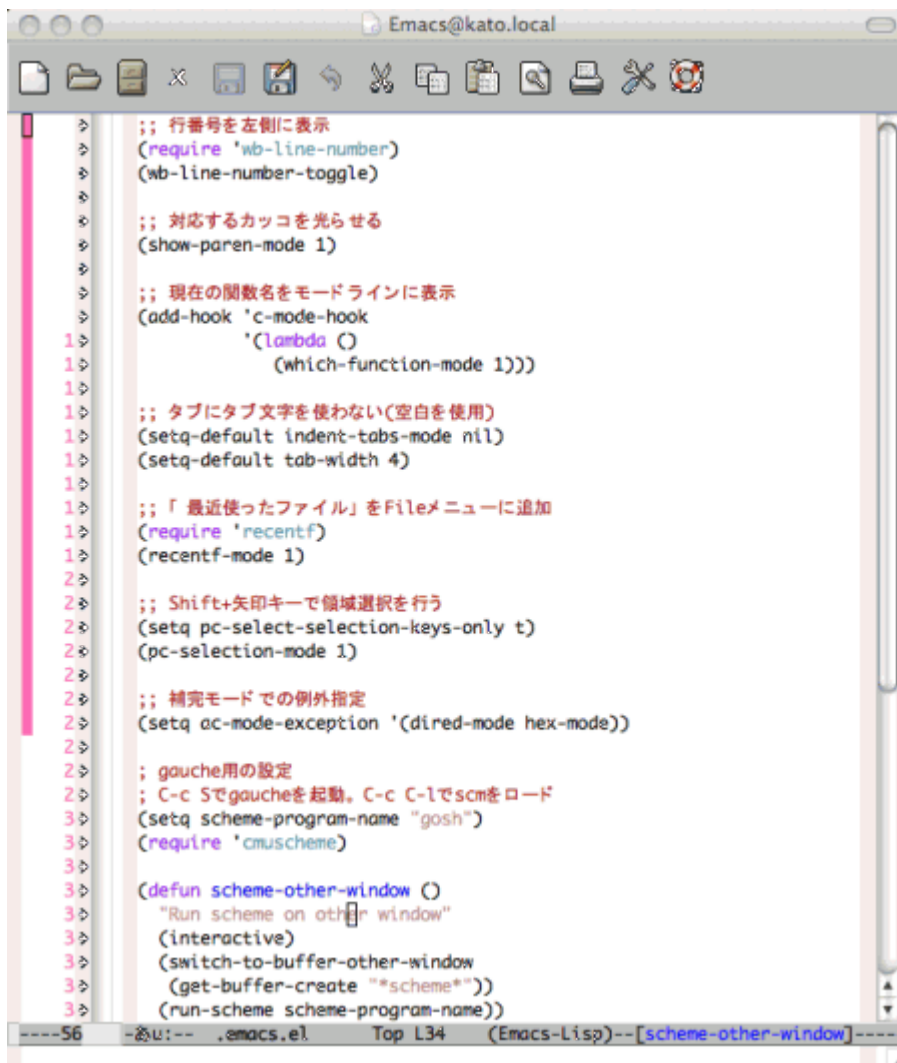
他所様の作ったものを使うしかない模様。Webを漁ってみると、どうも有名どころとして3つあるみたい。

- [wb-line-number.el](#)
- [setnu.el\(&setnu+.el\)](#)
- [linum.el](#)

まずwb-line-number.elは、ダウンロードしたwb-line-number.elをrequireして、適宜ONにすればOK。

```
(require 'wb-line-number) ;; これを.emacs.elに追加
(wb-line-number-toggle) ;; こちらは適宜呼んであげる
```

でも、Carbon Emacs上で試したら下図のように行を正しく認識してくれなかった。なぜだろう？？



```
Emacs@kato.local
;; 行番号を左側に表示
(require 'wb-line-number)
(wb-line-number-toggle)

;; 対応するカッコを光らせる
(show-paren-mode 1)

;; 現在の関数名をモードラインに表示
(add-hook 'c-mode-hook
  '(lambda ()
    (which-function-mode 1)))

;; タブにタブ文字を使わない(空白を使用)
(setq-default indent-tabs-mode nil)
(setq-default tab-width 4)

;; 「最近使ったファイル」をFileメニューに追加
(require 'recentf)
(recentf-mode 1)

;; Shift+矢印キーで領域選択を行う
(setq pc-select-selection-keys-only t)
(pc-selection-mode 1)

;; 補完モードでの例外指定
(setq ac-mode-exception '(dired-mode hex-mode))

; gauche用の設定
; C-c Sでgaucheを起動。C-c C-lでscmをロード
(setq scheme-program-name "gosh")
(require 'cmuscheme)

(defun scheme-other-window ()
  "Run scheme on other window"
  (interactive)
  (switch-to-buffer-other-window
   (get-buffer-create "**scheme*")))
(run-scheme scheme-program-name))

---56 --- .emacs.el Top L34 (Emacs-Lisp)-- [scheme-other-window] ---
```

そこでsetnu.elを調べてみると、これは・古い、・バグが多い、・修正版として用意されたsetnu+.elがリンク切れ、というように全然使えなかったのでパス。  
結局、最後のlinumが一番シンプルで良かったのでこれを使うことにした。使い方はsetnu.elとほぼ同じ。

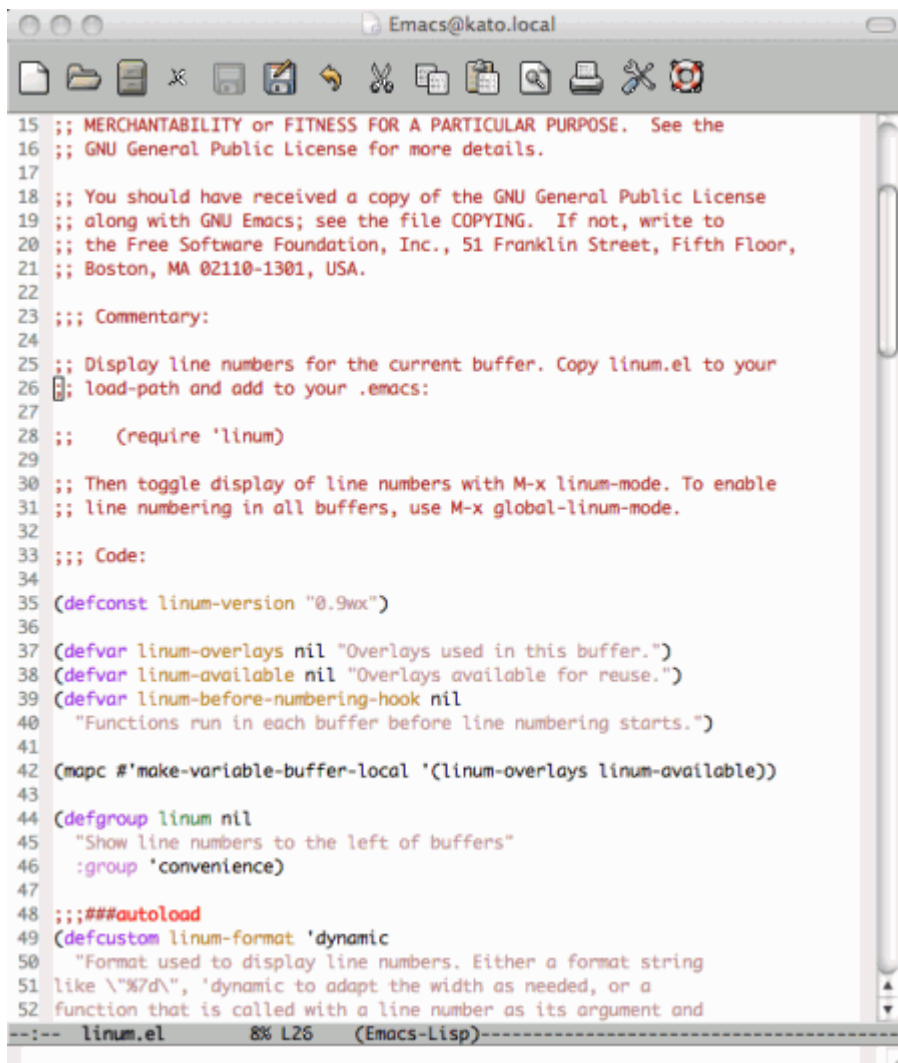
.emacs.elに

```
(require 'linum.el)
```

と記述しておき、表示するときは

```
M-x linum-mode
```

とするだけ。



```
15 ;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16 ;; GNU General Public License for more details.
17
18 ;; You should have received a copy of the GNU General Public License
19 ;; along with GNU Emacs; see the file COPYING. If not, write to
20 ;; the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor,
21 ;; Boston, MA 02110-1301, USA.
22
23 ;;; Commentary:
24
25 ;; Display line numbers for the current buffer. Copy linum.el to your
26 ;; load-path and add to your .emacs:
27
28 ;; (require 'linum)
29
30 ;; Then toggle display of line numbers with M-x linum-mode. To enable
31 ;; line numbering in all buffers, use M-x global-linum-mode.
32
33 ;;; Code:
34
35 (defconst linum-version "0.9wv")
36
37 (defvar linum-overlays nil "Overlays used in this buffer.")
38 (defvar linum-available nil "Overlays available for reuse.")
39 (defvar linum-before-numbering-hook nil
40 "Functions run in each buffer before line numbering starts.")
41
42 (mapc #'make-variable-buffer-local '(linum-overlays linum-available))
43
44 (defgroup linum nil
45 "Show line numbers to the left of buffers"
46 :group 'convenience)
47
48 ;;;##autoload
49 (defcustom linum-format 'dynamic
50 "Format used to display line numbers. Either a format string
51 like \"%7d\", 'dynamic to adapt the width as needed, or a
52 function that is called with a line number as its argument and
--:-- linum.el 8% L26 (Emacs-Lisp)-----
```

ちなみにemacs.elによく書かれている(line-number-mode t)はモードラインに現在行を表示させるだけで、linum-modeのように画面の左側に行番号情報を表示してくれるわけじゃない。(column-number-mode t)も同じで、モードラインに桁数を表示するだけで違うから注意。

## あったらいいな

### diff+Mergeを、日本語表示に対応させた上で実現

どうやるの？ ediffはいいけれど、ファイルの文字コードを自動判別してくれるわけではないみたい(Windowsと開発環境が混在しているとShift-JISも混じってきて面倒)。さて、どうしたものか。

### 入力しようとしている名前を補完

どうやるの？ ac-modeが使えるそうだけれど、今後消えゆく運命にありそうなEmacsLispなので、代替物を探さねば。

### 選択領域をハイライトまたは反転させる

emacs.elに以下の式を記述しておく

```
(transient-mark-mode 1)
```

Emacsのデフォルト設定だと、領域を選択している最中の選択箇所はハイライトされず、「今どこを選んでいるの？」というのが往々にして分からなくなりがち。この場合は上記のとおり記述してお

けばOK, 領域を選択している際の色を変えられます。次の「Shift+矢印キー」と組み合わせで設定しておくとう便利かな。

### Shift + 矢印キーで領域選択する

.emacs.elに以下の式を記述しておく

```
(setq pc-select-selection-keys-only t)
(pc-selection-mode 1)
```

これで大丈夫です。Mac OSXのCarbon EmacsはこれがデフォルトでONになっているみたいだけど、Meadow3.00ではOFFだったので記録しておこう。

### 矩形領域の選択

.emacs.elに以下を追加しておき、

```
(autoload 'sense-region-on "sense-region"
          "System to toggle region and rectangle." t nil)
(sense-region-on)
```

範囲選択している状態で

```
C-SPC
```

を入力。

整形テキスト、テキストで書かれた表を取り扱うときに矩形選択機能があると便利です。秀丸ではこの機能があって、ボクはC-bに割り当てて多用していました。で、Emacsにも当然同じ機能があるわけですが・・・

- 入力が面倒くさい(デフォルトのキーバインド(C-x rk, C-x ry)が有り得ない。替えればいいけどさ)
- 矩形選択中なのに画面は通常選択のままイメージが掴みづらい

というのが気になります。特に後者、常用するにはちと厳しい。

そこで見つけたのが冒頭に紹介した[session.el><http://taiyaki.org/elisp/sense-region/>]。これは便利で、範囲選択中にC-SPCすると通常選択と矩形選択を切り替えてくれます。しかも矩形選択中は矩形の選択領域が表示されますし、切り取りや貼り付けもC-w, C-yで出来ます \*4。

```
;; 範囲選択中にC-SPCで文字列選択⇔矩形選択を切り替えられる
;; http://taiyaki.org/elisp/sense-region/
;; src: http://taiyaki.org/elisp/sense-region/src/sense-region.el
(autoload 'sense-region-on "sense-region"
          "System to toggle region and rectangle." t nil)
(sense-region-on)
```

before

```
;; 範囲選択中にC-SPCで文字列選択⇔矩形選択を切り替えられる
;; http://taiyaki.org/elisp/sense-region/
;; src: http://taiyaki.org/elisp/sense-region/src/sense-region.el
(autoload 'sense-region-on "sense-region"
          "System to toggle region and rectangle." t nil)
(sense-region-on)
```

after!!

さらにこのEmacsLispには追加機能があって、領域を選択していないときにCtrl+[space] を繰り返す

と単語がリージョンに追加されていってくれます。便利だ。

上記で紹介したサイトからのソースリンクが切れてしまっているのを載せておきます。  
<http://taiyaki.org/elisp/sense-region/src/sense-region.el> です。ページのリンク先には「www」  
が付いてしまっているようです。

## その他

上の機能を探している途中で見つけた、「これも覚えておいた方がよいな」というコマンドや設定を書いておきます。

### マウスのホイールスクロールを加速させたくない

Emacsのマウスホイールの動き、なんだか他のアプリとは少し違うんですね。少しでも回しすぎると一気に最後まで飛んで行ってしまうような。どうもマウスを連続して回していると、wheel-up/downからdouble/triple-wheel-up/downというイベントに変わってしまっているみたいです。そこで振る舞いをカスタマイズしつつ、なじみやすい設定に落とし込んでみました。

```
;; マウスのホイールスクロールスピードを調節  
;; (連続して回しているととんでもない早さになってしまう。特にLogicoolのマウス)  
(global-set-key [wheel-up] '(lambda () "" (interactive) (scroll-down 1)))  
(global-set-key [wheel-down] '(lambda () "" (interactive) (scroll-up 1)))  
(global-set-key [double-wheel-up] '(lambda () "" (interactive) (scroll-down 1)))  
(global-set-key [double-wheel-down] '(lambda () "" (interactive) (scroll-up 1)))  
(global-set-key [triple-wheel-up] '(lambda () "" (interactive) (scroll-down 2)))  
(global-set-key [triple-wheel-down] '(lambda () "" (interactive) (scroll-up 2)))
```

うわ、コレすごい快適。

使っているマウスがLogicoolのマウスで、例のMicroGear™スクロールホイールを使っていると速攻でdouble/tripleに移行してしまうので、すばらしい!

ちなみに参考にさせてもらったサイトは[こちら](#)です。感謝です。

### 「最近使ったファイル」をメニューに表示させる

```
.emacs.elに以下の式を記述しておく  
(require 'recentf)  
(recentf-mode 1)
```

このメニューの設定は色々と変えられます。

### 文字列の自動補完

```
(M-x ac-mode)
```

上記はかなり賢い補完機能を持った。候補1つ1つがUndoの対象になってしまうところ以外は非の打ち所がないです。Emacsを使う上では必須機能に近い、と見ました。

[公式サイトはこちら](#)。

注意(2008/10/25) : [CarbonEmacsの2008年秋版](#)ではac-modeが無くなるみたいです。

### Subversionクライアント

psvn、ではなくてdsvnが便利。というのも[このサイト](#)に書いてあるとおり、作業中に?キーでキーボード一覧を表示できるから。

```
;; Subversion用の設定 (psvn.elとdsvn.elを併用)
;; (require 'psvn)
(autoload 'svn-status "dsvn" "Run `svn status'." t)
(autoload 'svn-update "dsvn" "Run `svn update'." t)
```

熟練者にはpsvnでも良いんだろうけれど、ボクみたいな初心者にはdsvnがいいのです。

### 対応する括弧に移動

```
C-M-f/b または C-M-n/p
```

前者はS式、後者は括弧。どちらも便利だけれど、Cでは後者かな。ただ、C-M-というのは押しづらいので

```
;; 対応する括弧に移動(C-M-f/p相当)
(global-set-key [? \C-{}] 'backward-list)
(global-set-key [? \C-}] 'forward-list)
```

というようにC-{}にアサインしておきました(本当はC-[]にアサインしたかったけれど、C-[がESCキーと認識されてしまって対応できず、どうすれば良いんだろう?)。

## 補足

### 外部からダウンロードした.elファイルをCarbonEmacsにインストールする方法()

#### CarbonEmacsアプリ下に保存(一番手っ取り早い将来性がない)

手順は以下のとおりです。

1. .elファイルをダウンロード
2. .elファイルを「/Applications/Emacs.app/Contents/Resources/site-lisp/」にコピー

これだけ。ちなみにバイトコンパイルする場合は、同名のファイルを開いて、

```
M-x byte-compile-file
```

で.elcファイルを作成しておけばオーケーです。

この方法は特別な設定が不要なところが利点ですが、CarbonEmacsのバージョンを更新したりすると全てご破算になってしまうのであまりオススメしません。

#### .emacs.elでパスを指定し、そこに保存する(オススメ)

手順は以下のとおり。

1. .elファイルをダウンロード
2. .emacs.elのできるだけ最初の方でadd-to-listを記述
3. そこで指定したディレクトリに.elファイルをコピー

たとえば私の場合はホームディレクトリ下に.emacs.d/site-lispを作り、そこに保存しています。

```
(add-to-list 'load-path "~/.emacs.d/site-lisp")
(let ((default-directory (expand-file-name "~/.emacs.d/site-lisp")))
  (normal-top-level-add-subdirs-to-load-path)
)
```

これだとCarbonEmacsを書き換えても.elファイルが残るのでオススメです。不可視ファイルに保存するのがイヤな人はディレクトリ名を適当に変更してください。

## 参考にした情報

### Special Thanks

---

- かみむらさん、コメント欄で色々とお教えていただきました。ありがとうございます。

### 書籍

---

- [入門 GNU Emacs](#)
  - このページ作成で最も使用した本。これがなければページを作ろうとすら思いませんでした。
- [GNU Emacsデスクトップリファレンス](#)
  - あまり役立たなかったけれど、上の本よりは手元に置いておきやすいのでポチポチ使いました。

### インターネット

---

#### 国内

---

- [秀まるおのホームページ](#)
  - 秀丸のホームページ
- [Wikipedia\(ja\)の記事](#)
  - どちらかといえばEmacsの歴史を知りたい人向け。
- [プリプロセッサで無効化されたコードをグレイアウトするにはどうしたらいいですか？](#)
  - 表題の質問をはてなに出した人がいました。
- [Rubyのメソッドの補完を行うanything-rcodetools.elをリリース](#)
  - 今回対象にしているetagsとは別(多分)のRubyメソッド補完elisp。今度試してみます。

#### 海外

---

- [Slashdotの記事「\(Stupid\) Useful Emacs Tricks?」](#)
  - 米SlashdotでEmacsのトリックで何かある?というトピックに集まったたくさんの記事。これは面白いです。
- [Emacs Nifty Tricks](#)
  - Emacs小技の総本山的サイト。どんだけあるの?

## コメント

もしあれば、どうぞ。

- 通常のコメントは、M-; (私の環境では) -- かみむら (2008-08-01 21:14:22)
- 一括コメントは、リージョンした状態で M-; -- かみむら (2008-08-01 21:14:49)
- grep は、M-x grep -- かみむら (2008-08-01 21:15:46)
- むしろ、M-x find-grep で、再帰的に grep してくれるのが便利です。 -- かみむら (2008-08-01

21:16:32)

- あと、etags があるならば、関数名にカーソルを配置して、M-. で、その関数定義の場所にジャンプしてくれます。 -- かみむら (2008-08-01 21:18:06)
- 保存する文字コードの指定は、M-x ret f です。 -- かみむら (2008-08-01 21:19:37)
- 最後に編集した場所への移動は、私は、Undo してます。あと、C-x 2 で画面を 2 つにして、片方で切り取って、片方で貼り付けとかも便利です。 -- かみむら (2008-08-01 21:20:50)
- ハードタブは、C-q TAB でしょうか。C-q C-i も可能かと。 -- かみむら (2008-08-01 21:23:30)
- 戻る方向への検索は、C-r も使えるかもしれませんが。 -- かみむら (2008-08-01 21:24:25)
- 私は、Emacs を応援してます！ -- かみむら (2008-08-01 21:25:21)
- かみむらさん、色々ありがとうございます！色々知らなかったことがたくさんあり、どれも参考になります。Undoで最後の場所に戻る、というのは発想の転換ですね。すばらしい！このページ、更新が途中で止まっているのでまた再開しますね。かみむらさんのおかげでやる気が出ました。 -- かときち (2008-08-02 13:51:08)
- 初めまして。記事の方参考にさせて頂きました。ac-mode の振り替えは標準添付の hippie-expand でいかがでしょうか? -- すが (2008-11-22 02:31:16)
- すがさん、ありがとうございます！hippie-expandがどういうものか知らなかったのも、ac-mode と比べてどうか、調べてみますね。感謝です。 -- かときち (2008-11-25 21:59:18)
- > だけど、除外ファイルも一度検索リストに入れられてしまうので遅いという、けっこう重大な欠点があったので次第に使わなくなってしまいました。これは改善されたみたい? -- y (2009-03-30 23:21:14)
- 遅い、というのはhippie-expandのことですよ。ac-modeもあまり使わないことに気付いて、今は使用を止めています・・・。コメントありがとうございます。あ、そういえば最近speedbarのようなバッファを同じウィンドウの中で表示する、というEmacs-Lispが公開されていたのを思い出しました。試してみようっと。 -- かときち (2009-03-31 02:00:50)
- [マクロ外コードのグレイアウト] cpp-highlight-buffer が良いみたいです。ただ、起動時にこれを実行させる場合はcondition-caseで例外処理しないとダメみたいです。 -- まつもと (2009-11-14 22:38:49)
- [gtags] gtags-find-tag より gtags-find-tag-from-hereの方が、一気にジャンプ出来るんで良さげかもしれません。 -- まつもと (2009-11-14 22:42:01)
- まつもとさん、ありがとうございます。ちょっと使ってみて、試してみます。 -- かときち (2009-11-16 14:04:08)
- <http://www.bookshelf.jp/soft/meadow.html> の情報は非常に有用です。特にmoccure-edit.elは強力すぎて眩暈がします。 -- こうじ (2010-01-12 13:03:35)
- あと、行番号が要るというのはただの「思い込み」で、まともな行番号表示elispがないのは、必要ないことが判って作者が作るのを止めてしまうからだと思われれます。 -- こうじ (2010-01-12 13:08:13)
- こうじさん、コメントとリンク、ありがとうございます。行番号は...難しいですね。diffやチームでのコードレビューなどでは行番号が識別子になっていますしね。 -- かときち (2010-02-05 11:40:53)
- 「入力しようとしている名前を補完」は dabbrev を使っています。途中まで入力して M-/ で補完。違うのが補完されたら再度 M-/ です。 -- こうぞう (2011-01-25 21:32:33)
- 「最後に編集した箇所に移動」と似た動きに、C-x C-x と C-u C-SPC がある。前者は今の位置とマークした位置を交換する。後者はマーク履歴を順に巡る。 -- こうぞう (2011-01-25 21:42:09)
- Wine1.2.2で hm802\_signed.exe試用中ですが範囲選択の文字消えてません。僕の環境だけかも知れませんが。 -- yoshikazu (2011-02-19 22:52:58)
- その後の試用で矩形選択と横スクロールで文字消えました。秀丸ビミョーに不便です。 -- yoshikazu (2011-02-20 12:26:19)
- <http://itmst.blog71.fc2.com/blog-entry-44.html>のコメントに答えがありました。これで矩形選択や横スクロールでも文字消えません。 -- yoshikazu (2011-02-21 23:02:20)

名前:

コメント:

投稿