ユーティリティ関数群

Takaaki Ishikawa

Table of Contents

init.el に記述していた便利関数を，utility.el として分離しています．init.el では autoload を用いて utility.el を遅延読み込みするように設定してます．このようなファイルの分離で 60[ms]ほど起動を高速化できます．

注意：コピペだけでは動かない可能性があります．

1. ユーティリティ関数

1.1. iTerm2.app を呼び出す関数

```
;;;###autoload
(defun my-cmd-to-open-iterm2 ()
  (interactive)
  (shell-command-to-string "open -a iTerm2.app"))
```

1.2. サボっていると Kyoko さんに怒られる

macOS 用の関数です．別途，Kyoko さんの音声をインストールしておく必要があります．Mavericks だと，Otoya さんも使えます．

```
(defvar my-kyoko-mad-mode nil)
;;;###autoload
(defun my-kyoko-mad-mode-toggle ()
  (interactive)
  (setq my-kyoko-mad-mode (not my-kyoko-mad-mode))
  (message (concat "Kyoko mad mode: "
                   (if my-kyoko-mad-mode "ON" "OFF"))))
;;;###autoload
(defun my-kyoko-mad ()
  (interactive)
  (when my-kyoko-mad-mode
    (shell-command-to-string
     "say -v Kyoko おいおまえ，遊んでないで，仕事しろ")))

;; She will be mad if you do nothing within 10 min.
(run-with-idle-timer 600 t 'my-kyoko-mad)
```

1.3. コンソールでカレントバッファのあるディレクトリに移動する

Finder で開きたいだけならば，M-! でミニバッファに open . と打ち込むだけです．

```
(defcustom open-current-directory-console-program "iTerm2.app"
  "Specify a console program"
```

```
  :type 'string
  :group 'takaxp-mac)

;;;###autoload
(defun my-open-current-directory-in-terminal ()
  " Open Current Directory for MacOSX
  0) Put this function in your .emacs
  1) M-x open-current-directory
  2) Terminal will open automatically
  3) Type M-v to paste and move to a path to the current directory in Emacs"
  (interactive)
  (let ((file-path (buffer-file-name (current-buffer))))
    (unless (string= file-path nil)
      (let ((directory
             (substring file-path 0
                        (-
                         (length file-path)
                         (length (buffer-name (current-buffer)))))))
        (message "%s" directory)
        (shell-command-to-string (concat "echo cd " directory " |pbcopy"))
        (shell-command-to-string
         (concat "open -a " open-current-directory-console-program))))))
```

1.4. ファイルに含まれるテーブルを使って定時にアラートを表示する

```
;;;###autoload
(defun my-update-alarms-from-file ()
  (interactive)
  (let ((bname (buffer-name)))
    (when (string= bname "daily.org")
      (my-set-alarms-from-file (concat "~/Dropbox/org/db/" bname)))))

(defun my-set-alarms-from-file (file)
  "Make alarms from org-mode tables. If you have an org-mode file
     with tables with the following format:
     |------+-------+--------------------|
     | Flag |  Time | Content            |
     |------+-------+--------------------|
     |      | 07:00 | Wakeup             |
     |      |       | Read papers        |
     | X    | 12:00 | Clean up your desk |
     When it is 7:00 and 12:00, Growl notify with a message which is specified
     content column from the table. \"Read papers\" will be ignored.
     \"Clean up your desk\" will be shown by sticky mode"
  (let
      ((lines (read-line file)))
    (cancel-function-timers 'my-desktop-notify) ;; clear existing timers
    (while lines
      (set-alarm-from-line (decode-coding-string (car lines) 'utf-8))
      (setq lines (cdr lines)))))

(defun set-alarm-from-line (line)
  (let
      ((hour nil)
       (min nil)
       (current-hour nil)
       (current-min nil)
       (action nil))
    (when (string-match "\\([0-2]?[0-9]\\):\\([0-5][0-9]\\)" line)
      (setq hour (substring line (match-beginning 1) (match-end 1)))
```

```
      (setq min (substring line (match-beginning 2) (match-end 2)))
      (when (string-match
             "\|\\s-*\\([^\|]+[^ ]\\)\\s-*\|" line (match-end 2))
        (setq action
              (substring line (match-beginning 1) (match-end 1)))))
    (when (and (and hour min) action)
      ;;         (message "[%s:%s] => %s" hour min action)
      (setq current-hour (format-time-string "%H" (current-time)))
      (setq current-min (format-time-string "%M" (current-time)))
      (when (> (+ (* (string-to-number hour) 60)
                  (string-to-number min))
               (+ (* (string-to-number current-hour) 60)
                  (string-to-number current-min)))
        (let ((s nil))
          (when (string-match "^\|\\s-*X\\s-*\|" line)
            (setq s 'sticky))
          (set-notify-macos hour min action s))))))

(defun set-notify-macos (hour min action sticky)
  "`alerter' is required."
  (run-at-time (format "%s:%s" hour min) nil
               'my-desktop-notify
               "macos" "Org Mode" hour min action sticky))

(declare-function my-desktop-notification "init-org")
(defun my-desktop-notify (type title hour min action sticky)
  "An interface to `my-desktop-notification'."
  (cond
   ((string= type "macos")
    (my-desktop-notification
     title (format "%s:%s %s" hour min action) sticky))))

(defun read-line (file)
  "Make a list from a file, which is divided by LF code"
  (with-temp-buffer
    (insert-file-contents-literally file)
    (split-string
     (buffer-string) "\n" t)))
```

1.5. 頻繁に利用するファイルを ring 形式でたどる

http://d.hatena.ne.jp/rubikitch/20111120/elispbook

```
(defvar my-file-ring nil)

;;;###autoload
(defun my-make-file-ring (files)
  (setq my-file-ring (copy-sequence files)))
;;    (setf (cdr (last my-file-ring)) my-file-ring))
(my-make-file-ring
 '("~/Dropbox/org/tr/work.org" "~/Dropbox/org/db/daily.org"
   "~/Dropbox/org/minutes/wg1.org" "~/Dropbox/org/tr/work.org"
   "~/Dropbox/org/academic.org" "~/Dropbox/org/org2ja.org"
   "~/Dropbox/org/db/article.org" "~/Dropbox/emacs.d/config/init.org"))

;;;###autoload
(defun my-open-file-ring ()
  (interactive)
  (find-file (car my-file-ring))
```

```
  (setq my-file-ring
        (append (cdr my-file-ring)
                (list (car my-file-ring)))))

;;     (setq my-file-ring (cdr my-file-ring)))
```

1.6. 引数の org バッファを開く

```
;;;###autoload
(defun my-show-org-buffer (file)
  "Show an org-file on the current buffer."
  (interactive)
  (if (get-buffer file)
      (let ((buffer (get-buffer file)))
        (switch-to-buffer buffer)
        (message "%s" file))
    (find-file (concat "~/Dropbox/org/" file)))
  (when (fboundp 'my-org-agenda-to-appt)
    (my-org-agenda-to-appt)))
```

1.7. org バッファにいつものヘッダを追加する

```
(declare-function org-end-of-line "org")

;;;###autoload
(defun insert-org-file-header-template ()
  (interactive)
  (when (string= major-mode 'org-mode)
    (let ((title "#+title:\t\n")
          (date "#+date: \t\n")
          (author "#+author:\tTakaaki ISHIKAWA <takaxp@ieee.org>\n")
          (option "#+options:\t\\n:t\n")
          (other "\n"))
      (goto-char 0)
      (save-excursion
        (insert title date author option other))
      (when (require 'org nil t)
        (org-end-of-line)))))
```

1.8. 議事録ひな形を書き入れる

```
;;;###autoload
(defun insert-minutes-template ()
  (interactive)
  (when (string= major-mode 'org-mode)
    (let ((date "日時：\n")
          (place "場所：\n")
          (attendance "出席者：\n")
          (documents "資料：\n\n"))
      (save-excursion
        (insert date place attendance documents)))))
```

1.9. ランダムの文字列を取得する

引数で桁数を渡すと，ランダムな数値の文字列を取得できます．org-mode で適当なタイト

ルのツリーを生成したい時に使っています．

```
;;;###autoload
(defun my-get-random-string (length)
  "Get a string contain the length digit number with random selection"
  (interactive)
  (random t)
  (cond ((> length 0)
         (let
             ((count length)
              (string nil)
              (tmp nil))
           (while (< 0 count)
             (setq count (1- count))
             (setq tmp string)
             (setq string
                   (concat tmp (number-to-string (random 10)))))
           (message "%s" string)))
        (t "0")))
```

1.10. 行頭に" - "を挿入する

```
;;;###autoload
(defun add-itemize-head (arg)
  "Insert \" - \" at the head of line.
  If the cursor is already at the head of line, it is NOT returned back to the
  original position again. Otherwise, the cursor is moved to the right of the
  inserted string. \" - [ ] \" will be inserted using C-u prefix."
  (interactive "P")
  (let ((item-string " - "))
    (when arg
      (setq item-string " - [ ] "))
    (cond ((= (point) (line-beginning-position))
           (insert item-string))
          (t (save-excursion
               (move-beginning-of-line 1)
               (insert item-string))))))

;;;###autoload
(defun add-itemize-head-checkbox ()
  "Insert \" - [ ] \" at the head of line.
  If the cursor is already at the head of line, it is NOT returned back to the
  original position again. Otherwise, the cursor is moved to the right of the
  inserted string."
  (interactive)
  (let ((item-string " - [ ] "))
    (cond ((= (point) (line-beginning-position))
           (insert item-string))
          (t (save-excursion
               (move-beginning-of-line 1)
               (insert item-string))))))
```

1.11. 定期実行関数

org バッファからカレンダーを生成し，外部サーバに投げます．また，MobileOrg に最新情報を流しています．

```
(defvar ox-icalendar-activate nil)
(with-eval-after-load "org"
  (run-with-idle-timer 180 t 'my-reload-ical-export)
  ;;    (run-with-idle-timer 1000 t 'org-mobile-push)
  ;; FIXME
  (add-hook 'focus-in-hook (lambda () (setq ox-icalendar-activate nil)))
  (add-hook 'focus-out-hook (lambda () (setq ox-icalendar-activate t))))

(declare-function my-ox-upload-icalendar "init-org")
;;;###autoload
(defun my-reload-ical-export ()
  "Export org files as an iCal format file"
  (interactive)
  (when (and (string= major-mode 'org-mode)
             ox-icalendar-activate)
    (my-ox-upload-icalendar)))
```

1.12. ブラウザの設定

```
;; http://stackoverflow.com/questions/4506249/how-to-make-emacs-org-mode-open-
links-to-sites-in-google-chrome
;; http://www.koders.com/lisp/fidD53E4053393F9CD578FA7D2AA58BD12FDDD8EB89.aspx?
s="skim
(when (autoload-if-found
       '(browse-url)
       "browse-url" nil t)
  (with-eval-after-load "browse-url"
    (cond
     ((eq window-system 'ns)
      (custom-set-variables
       '(browse-url-generic-program 'google-chrome)))
     ((eq window-system 'mac)
      (custom-set-variables
       '(browse-url-browser-function 'browse-url-generic)
       '(browse-url-generic-program
         "/Applications/Google Chrome.app/Contents/MacOS/Google Chrome")
       ))
     (t
      nil))))

;;(setq browse-url-browser-function 'browse-url-default-macosx-browser)
;;(setq browse-url-browser-function 'browse-url-default-windows-browser)
;;(setq browse-url-browser-function 'browse-url-chrome)
```

1.13. バックアップファイルの削除

```
;; find ~/.emacs.d/backup  -type f -name '*15-04-24_*' -print0 | while read -r
-d '' file; do echo -n " \"$file\""; done | xargs -0

;;;###autoload
(defun recursive-delete-backup-files (days)
  (if (= days 1)
      1
    (recursive-delete-backup-files (1- days)))
  (delete-backup-files days))

;;;###autoload
(defun delete-backup-files (&optional day-shift)
  "Delete backup files created in yesterday.
```

```
  > find ~/.emacs.d/backup -type f -name '*YY-MM-DD_*' -print0 | xargs -0"
  (interactive)
  (unless day-shift
    (setq day-shift 1))
  (let* ((backup-dir "~/.emacs.d/backup")
         (cmd (concat "find " backup-dir "  -type f -name \'*"
                      (format-time-string
                       "%y-%m-%d_"
                       (time-subtract (current-time)
                                      (seconds-to-time
                                       (* day-shift (* 24 3600)))))
                      "*\' -print0 | while read -r -d \'\' file; "
                      " do echo -n \" \\\"$file\\\"\"; done | xargs -0"))
         (files (shell-command-to-string cmd)))
    ;; (message "%s" cmd)
    (unless (string= files "")
      (message "%s" (chomp files))
      (shell-command-to-string (concat "mv -v " (chomp files) " ~/.Trash")))))
```

1.14. chomp

改行コードを削除した文字列を返す．

```
;;;###autoload
(defun chomp (str)
  "Chomp leading and tailing whitespace from STR."
  (while (string-match "\\`\n+\\|^\\s-+\\|\\s-+$\\|\n+\\'"
                       str)
    (setq str (replace-match "" t t str)))
  str)
```

1.15. 特定のファイルを Dropbox 以下にバックアップする

```
(defun my-backup (files &optional dropbox)
  "Backup a file to `Dropbox/backup' directory.
If `dropbox' option is provided then the value is uased as a root directory."
  (interactive "P")
  (let ((system (system-name))
        (rootdir (or dropbox "~/Dropbox")))
    (if (and system
             (stringp rootdir)
             (file-directory-p (or rootdir (expand-file-name rootdir))))
        (mapc
         (lambda (file)
           (if (and (stringp file)
                    (file-readable-p (or file (expand-file-name file))))
               (shell-command-to-string
                (concat "cp -f " file " " rootdir "/backup/" system "/"))
             (message (format "--- backup failure: %s" file))))
         (if (listp files)
             files
           (list files)))
      (user-error (format "--- backup-dir does not exist: %s" rootdir)))))
```

1.16. ゴミ箱を空にする [MACOS]

```
;;;###autoload
```

```
(defun mac:delete-files-in-trash-bin ()
  (interactive)
  (do-applescript
   (concat
    "tell application \"Finder\"\n"
    "set itemCount to count of items in the trash\n"
    "if itemCount > 0 then\n"
    "empty the trash\n"
    "end if\n"
    "end tell\n"))
  (my-desktop-notification "Emacs" "Empty the trash, done."))
```

1.17. 終了シーケンスのデバッグ用関数

```
;;;###autoload
(defun my-kill-emacs ()
    (switch-to-buffer "*Messages*")
    (message "3: %s" kill-emacs-hook)
    (y-or-n-p "Sure? "))

;;;###autoload
(defun my-kill-emacs-hook-show ()
  "Test Emacs killing sequence."
  (add-hook 'after-init-hook
            (lambda () (message "1: %s" kill-emacs-hook)) t)
  (with-eval-after-load "postpone"
    (message "2: %s" kill-emacs-hook))
  (add-hook 'kill-emacs-hook #'my-kill-emacs))
```

1.18. [package.el] MELPA 利用設定

```
;;;###autoload
(defun my-setup-package-el ()
  "Setting up for installing packages via built-in package.el.
Downloaded packages will be stored under ~/.eamcs.d/elpa."
  (when (and (require 'package nil t)
             (boundp 'package-archives))
    (let* ((no-ssl (and (memq system-type '(windows-nt ms-dos))
                        (not (gnutls-available-p))))
           (proto (if no-ssl "http" "https")))
      (add-to-list 'package-archives
                   (cons "melpa" (concat proto "://melpa.org/packages/")) t)
      (add-to-list 'package-archives
                   (cons "takaxp" "~/devel/git/melpa/packages/") t))
    (package-initialize)))
```

1.19. org バッファを評価

org-buffer を評価して Emacs の設定ファイルを生成 / 読み込みまでを自動化します．この設定では，`init.org` と `utility.org` の 2 つのバッファでのみ評価されるようになっています．

```
(declare-function org-babel-tangle "org-babel")

;;;###autoload
(defun my-eval-org-buffer ()
```

```
  "Load init.org/utility.org and tangle init.el/utility.el."
  (interactive)
  (if (and (require 'org nil t)
           (eq major-mode 'org-mode)
           (member (buffer-name) '("init.org" "utility.org")))
      (progn
        (org-babel-tangle)
        (let ((tangled-file
               (concat (file-name-sans-extension (buffer-file-name)) ".el")))
          (when (file-exists-p tangled-file)
            (byte-compile-file tangled-file))))
    (message "Nothing to do for this buffer.")))
```

1.20. TODO リージョン内のブリッツを操作する

```
;;;###autoload
(defun my-org-list-insert-items (begin end)
  (interactive "r")
  (when mark-active
    (let* ((bullet " - ")
           (len (string-width bullet)))
      (goto-char begin)
      (while (and (re-search-forward (concat "\\(^[ \t]*\\)") end t)
                  (not (equal (point) end)))
        (replace-match (concat "\\1" bullet) nil nil)
        (setq end (+ end len)))
      (goto-char begin))))

;;;###autoload
(defun my-org-list-delete-items (begin end)
  (interactive "r")
  (when mark-active
    (let* ((bullet "- ")
           (len (string-width bullet)))
      (goto-char begin)
      (while (re-search-forward
              (concat "\\(^[ \t]*\\)" bullet) end t)
        (replace-match "" nil nil)
        (setq end (- end len)))
      (goto-char begin))))

;;;###autoload
(defun my-org-list-insert-checkbox-into-items (begin end)
  (interactive "r")
  (when mark-active
    (let* ((bullet "- ")
           (checkbox "[ ] ")
           (len (string-width checkbox)))
      (goto-char begin)
      (while (re-search-forward (concat "\\(^[ \t]*\\)" bullet) end t)
        (replace-match (concat "\\1" bullet checkbox) nil nil)
        (setq end (+ end len)))
      (goto-char begin))))

;;;###autoload
(defun my-org-list-delete-checkbox-from-items (begin end)
  (interactive "r")
  (when mark-active
    (let ((bullet "- ")
          (len (string-width "[ ] ")))
```

```
      (goto-char begin)
      (while (re-search-forward
              (concat "\\(^[ \t]*\\)" bullet "\\[.\\][ \t]+") end t)
        (replace-match (concat "\\1" bullet) nil nil)
        (setq end (- end len)))
      (goto-char begin))))

;;;###autoload
(defun my-org-list-insert-itms-with-checkbox (begin end)
  (interactive "r")
  (when mark-active
    (let* ((bullet " - ")
           (checkbox "[ ] ")
           (blen (string-width bullet))
           (clen (string-width checkbox)))
      (goto-char begin)
      (while (and (re-search-forward (concat "\\(^[ \t]*\\)") end t)
                  (not (equal (point) end)))
        (replace-match (concat "\\1" bullet checkbox) nil nil)
        (setq end (+ end blen clen)))
      (goto-char begin))))

;;;###autoload
(defun my-org-list-delete-items-with-checkbox (begin end)
  (interactive "r")
  (when mark-active
    (let* ((bullet "- ")
           (checkbox "[ ] ")
           (blen (string-width bullet))
           (clen (string-width checkbox)))
      (goto-char begin)
      (while (re-search-forward
              (concat "\\(^[ \t]*\\)" bullet "\\[.\\][ \t]+") end t)
        (replace-match "" nil nil)
        (setq end (- end blen clen)))
      (goto-char begin))))
```

1.21. TODO " - "と" - [ ] "をサイクルさせる

- ブリッツ行でないときは，ブリッツ化する．
- ブリッツ行の場合は，チェックボックス付きに変更する．
- チェックボックス付きブリッツ行の場合は，チェックボックスを取る．
- 引数付きで呼び出すと，チェックボックスの有無に依らずブリッツを取る．

```
;;;###autoload
(defun my-cycle-bullet-at-heading (arg)
  "Add a bullet of \" - \" if the line is NOT a bullet line."
  (interactive "P")
  (save-excursion
    (beginning-of-line)
    (let ((bullet "- ")
          (point-at-eol (point-at-eol)))
```

```
      (cond
       ((re-search-forward
         (concat "\\(^[ \t]*\\)" bullet "\\[.\\][ \t]+") point-at-eol t)
        (replace-match (if arg "" (concat "\\1" bullet)) nil nil))
       ((re-search-forward
         (concat "\\(^[ \t]*\\)" bullet) point-at-eol t)
        (replace-match (if arg "" (concat "\\1" bullet "[ ] ")) nil nil))
       ((re-search-forward
         (concat "\\(^[ \t]*\\)") point-at-eol t)
        (replace-match
         (concat "\\1 " bullet) nil nil))
       (t nil)))))
```

1.22. その他

```
;;; Test function from GNU Emacs (O'REILLY, P.328)
;;;###autoload
(defun count-words-buffer ()
  "Count the number of words in the current buffer"
  (interactive)
  (save-excursion
    (let ((count 0))
      (goto-char (point-min))
      (while (< (point) (point-max))
        (forward-word 1)
        (setq count (1+ count)))
      (message "buffer contains %d words." count))))

;;; Test function for AppleScript
;;; Cite: http://sakito.jp/emacs/emacsobjectivec.html
;;;###autoload
(defun do-test-applescript ()
  (interactive)
  (do-applescript
   (format
    (concat
     "display dialog \"Hello world!\" \r"))))

;;;###autoload
(defun describe-timer ()
  "see http://masutaka.net/chalow/2009-12-05-1.html"
  (interactive)
  (let ((tl timer-list)
        (timer nil))
    (pop-to-buffer (get-buffer-create "*timer*"))
    (erase-buffer)
    (insert
     "TIME           FUNCTION\n"
     "-------------- ---------------------\n")
    (while tl
      (setq timer (car tl))
      (insert
       (concat
        (format-time-string "%m/%d %T"
                            (list (aref timer 1)
                                  (aref timer 2)
                                  (aref timer 3)))
        " "
        (symbol-name (aref timer 5))
        "\n"))
```

```
      (setq tl (cdr tl)))
    (read-only-mode 1)))
```

1.23. XHTML を利用したガントチャート生成 [OBSOLETE]

最近使っていません．

```
(defcustom my-auto-install-batch-list-el-url nil
  "URL of a auto-install-batch-list.el"
  :type 'string
  :group 'takaxp-utility)

;; Publish an xml file to show a Gantt Chart
(defcustom default-timeline-csv-file nil
  "source.csv"
  :type 'string
  :group 'takaxp-utility)

(defcustom default-timeline-xml-business-file nil
  "XML file for business schedule"
  :type 'string
  :group 'takaxp-utility)

(defcustom default-timeline-xml-private-file nil
  "XML file for private schedule"
  :type 'string
  :group 'takaxp-utility)

(defcustom default-timeline nil
  "a template index.html"
  :type 'string
  :group 'takaxp-utility)

(defun export-timeline-business ()
  "Export schedule table as an XML source to create an web page"
  (interactive)
  (when (and default-timeline
             (and default-timeline-csv-file
                  default-timeline-xml-business-file))
    (shell-command-to-string (concat "rm -f " default-timeline-csv-file))
    (org-table-export default-timeline-csv-file "orgtbl-to-csv")
    (shell-command-to-string (concat "org2gantt.pl > "
                                     default-timeline-xml-business-file))
    (shell-command-to-string (concat "open " default-timeline))))

(defun export-timeline-private ()
  "Export schedule table as an XML source to create an web page"
  (interactive)
  (when (and default-timeline
             (and default-timeline-csv-file
                  default-timeline-xml-private-file))
    (shell-command-to-string (concat "rm -f " default-timeline-csv-file))
    (org-table-export default-timeline-csv-file "orgtbl-to-csv")
    (shell-command-to-string (concat "org2gantt.pl > "
                                     default-timeline-xml-private-file))
    (shell-command-to-string (concat "open " default-timeline))))
```

1.24. Auto-install をセットアップする [OBSOLETE]

いつも auto-install を使うわけではないので，必要時に `init-auto-install` を実行してパラメータを設定してから auto-install でパッケージを取得するようにしています．cask+pallet 環境に移行してからは使っていません．

```
(defun init-auto-install ()
  "Setup auto-install.el.
1. Set my-auto-install-batch-list-el-url
2. M-x init-auto-install
3. M-x auto-install-batch hoge"
  (interactive)
  (when (and (require 'auto-install nil t)
             my-auto-install-batch-list-el-url)
    (setq auto-install-batch-list-el-url my-auto-install-batch-list-el-url)
    (setq auto-install-directory default-path)
    (setq auto-install-wget-command "/opt/local/bin/wget")
    (auto-install-update-emacswiki-package-name t)
    ;; compatibility
    (auto-install-compatibility-setup))) ; for install-elisp users
```

1.25. org-buffer を dokuwiki 形式に変換し，kill-ring に格納 [OBSOLETE]

外部プログラム org2dokuwiki.pl を使います．

```
;;;###autoload
(defun org2dokuwiki-cp-kill-ring ()
  "Convert the current org-file to dokuwiki text, and copy it to kill-ring."
  (interactive)
  (when (eq major-mode 'org-mode)
    (cond (buffer-file-name
           (kill-new
            (shell-command-to-string
             (concat "cat " buffer-file-name "| perl "
                     (expand-file-name "~/Dropbox/scripts/org2dokuwiki.pl"))))
           (message "Copying %s ... done" buffer-file-name)
           (sit-for 1.5)
           (message ""))
          (t (message "There is NOT such a file.")))))
```

1.26. ミニバッファに日時を表示 [OBSOLETE]

- hydra に移行しました．

```
;;;###autoload
(defun my-date ()
  (interactive)
  (message "%s" (concat
                 (format-time-string "%Y-%m-%d") " ("
                 (format-time-string "%a.") ") "
                 (format-time-string "W:%W @")
                 (format-time-string "%H:%M"))))
```

1.27. 日付などを簡単に挿入する *[OBSOLETE]*

- hydra に移行しました．

http://www.fan.gr.jp/~ring/doc/elisp_20/elisp_38.html#SEC608

```
(defun insert-formatted-current-date (arg)
  "Insert a timestamp at the cursor position. C-u will add [] brackets."
  (interactive "p")
  (case arg
    (4 (if (equal major-mode 'org-mode)
           (org-time-stamp-inactive)
         (insert (format-time-string "[%Y-%m-%d]"))))
    (t (insert (format-time-string "%Y-%m-%d")))))

(defun insert-formatted-current-time ()
  (interactive)
  (insert (format-time-string "%H:%M")))

(defun insert-formatted-signature ()
  (interactive)
  (insert (concat (format-time-string "%Y-%m-%d") "  " user-full-name
                  "  <" user-mail-address ">")))
```

1.27.1. キーバインド

```
(global-set-key (kbd "C-c 0") 'insert-formatted-current-date)
(global-set-key (kbd "C-c 9") 'insert-formatted-current-time)
```

2. 未設定 / テスト中

2.1. UUID をファイル名にして所定のディレクトリにコピー / 移動

- すでに org-attach が存在するので用途が微妙に．．．

```
(defvar org-att-global-directory "~/Dropbox/org/attachment/")
(defun copy-file-with-uuid (input)
  (interactive "FFile name: ")
  (if (file-exists-p input)
      (let* ((id (org-id-uuid))
             (filename (expand-file-name input))
             (directory (file-name-directory filename))
             (extension (file-name-extension filename))
             (output (concat org-att-global-directory id "." extension)))
        (copy-file filename output)
        (message "--- Copied as %s " output)
        output)
    (message "--- %s does NOT exist." input)
    nil))

(defun rename-file-with-uuid (input)
  (interactive "FFile name: ")
  (if (file-exists-p input)
      (let* ((id (org-id-uuid))
```

```
              (filename (expand-file-name input))
              (directory (file-name-directory filename))
              (extension (file-name-extension filename))
              (output (concat directory id "." extension)))
         (rename-file filename output)
         (message "--- Renamed as %s " output)
         output)
     (message "--- %s does NOT exist." input)
     nil))

(defun org-link-uuid (input &optional overwrite)
  (interactive "FFile name: ")
  (let ((output
         (if overwrite
             (rename-file-with-uuid input)
           (copy-file-with-uuid input))))
    (when output
      (insert (concat "[[file+sys:" output
                      "][" (file-name-base input) "]]\n")))))
```

2.2. byte-compile の警告を抑制する

```
;; Avoid warning (for sense-region)
;; Warning: 'mapcar' called for effect; use 'mapc' or 'dolist' insted
(setq byte-compile-warnings
      '(free-vars unresolved callargs redefine obsolete noruntime
      cl-functions interactive-only make-local))
```

2.3. [window-resizer.el] 分割したウィンドウサイズを変更する

http://d.hatena.ne.jp/khiker/20100119/window_resize

以下の警告を参考に書き換えた．

```
In my-window-resizer:
utility.el:333:23:Warning: `last-command-char' is an obsolete variable (as of
    Emacs at least 19.34); use `last-command-event' instead.
```

```
;;;###autoload
(defun my-window-resizer ()
  "Control separated window size and position.
   Type {j,k,l,m} to adjust windows size."
  (interactive)
  (let (
;;         (window-obj (selected-window))
;;         (current-width (window-width))
;;         (current-height (window-height))
        (dx (if (= (nth 0 (window-edges)) 0) 1
              -1))
        (dy (if (= (nth 1 (window-edges)) 0) 1
              -1))
        action c)
    (catch 'end-flag
      (while t
        (setq action
              (read-key-sequence-vector (format "size[%dx%d]"
                                                (window-width)
```

```
                                                    (window-height))))
        (setq c (aref action 0))
        (cond ((= c ?l)
               (enlarge-window-horizontally dx))
              ((= c ?h)
               (shrink-window-horizontally dx))
              ((= c ?j)
               (enlarge-window dy))
              ((= c ?k)
               (shrink-window dy))
              ;; otherwise
              (t
               (let ((last-command-event (aref action 0))
                     (command (key-binding action)))
                 (when command
                   (call-interactively command)))
               (message "Quit")
               (throw 'end-flag t)))))))
```

2.4. [idle-requie]

```
(require 'idle-require)
(idle-require-mode 1)
```

2.5. [pdf-preview]

```
(require 'pdf-preview)
```

2.6. [EasyPG]

```
(when (require 'epa-setup nil t)
  (epa-file-enable))
```

2.7. [eblook]

```
;; eblook
(when (require 'eblook nil t)
  (autoload 'edict-search-english "edic"
    "Search for a translation of an English word" t)
  (autoload 'edict-search-kanji "edict"
    "Search for a translation of a Kanji sequence" t)
  (setq *edict-files* '("/Users/taka/Dropbox/Dic/LDOCE4"))
  (setq *edict-files* '("/Users/taka/Downloads/edict/edict")))
```

2.8. [iBuffer]

iBuffer で list-buffers をオーバーライド (C-x C-b で表示)

```
(defalias 'list-buffers 'ibuffer)
```

2.9. キーバインド

```
;; Multiple combination
; 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)
```

3. おわりに

以上です．