

```

; -----
; Yet Another HyperTIES Implementation, This Time In Emacs (YAHTITTIE)
; Don Hopkins
; -----

; -----
; Notes:
; * Make synonyms use the abbrev mechanism
;   use-abbrev-table define-local-abbrev abbrev-mode abbrev-expansion
; * Edit definitions, etc, with edit-in-transient-window
; * Show temporary messages with typeout-text
; * Use (temp-use-buffer ...) instead of (s-e (s-t-b) ...)
; * Use (progn-args 1) for conditional execution of args
; * Use define-hooked-local-abbrev to make .commands prompt for
;   arguments, with completion over known names, in storyboard mode.
;   Look at /usr/unimacs/lib/emacs/maclib/cmacs.ml
; -----

; -----
; Index manager interface

; Array size defaults:

; #master-indices      Initial master index stack size
(declare-global #master-indices)
(setq-default #master-indices 10)

; #articles            Initial article array size
(declare-global #articles)
(setq-default #articles 300)

; #pictures            Initial picture array size
(declare-global #pictures)
(setq-default #pictures 100)

; #targets             Initial target array size
(declare-global #targets)
(setq-default #targets 200)

; #entries             Initial entry array size
(declare-global #entries)
(setq-default #entries (+ #articles #pictures #targets))

; #fields              Initial field array size
(declare-global #fields)
(setq-default #fields 8)

; &master-indices      Master index stack pointer
(declare-global &master-indices)
(setq &master-indices 0)

; master-indices       Array of master index buffer names
(declare-global master-indices)
(setq master-indices (new-array #master-indices))

; (push-master-index buffername) ; pushes master index buffer on stack
(defun (push-master-index
  (save-excursion

```

```

(save-window-excursion
  (switch-to-buffer (arg 1 ": push-master-index (buffer) "))
  ; initialize master index buffer locals if necessary
  (if (! (& (is-bound &entries) &entries))
      (init-master-index)
      )
  (if (<= (array-size master-indices) &master-indices)
      (setq master-indices (grow-array master-indices))
      )
  (setq-array master-indices (++ &master-indices) (current-buffer-name))
)
))

```

```

; (pop-master-index buffername) ; pops named master index buffer from stack
; (pop-master-index) ; pops top master index buffer from stack
(defun (pop-master-index
))

```

```

; dictionary buffer locals:
;
; Initialize the locals of the master index in the current buffer.
(defun (init-master-index

```

```

;   &articles           Article count
(declare-buffer-specific &articles)
(setq &articles 0)

```

```

;   article-names       Array of article names and synonyms
(declare-buffer-specific article-names)
(setq article-names (new-array #articles))

```

```

;   article-indices     Array of corresponding entry indices
(declare-buffer-specific article-indices)
(setq article-indices (new-array #articles))

```

```

;   &pictures           Picture count
(declare-buffer-specific &pictures)
(setq &pictures 0)

```

```

;   picture-names       Array of picture names
(declare-buffer-specific picture-names)
(setq picture-names (new-array #pictures))

```

```

;   picture-indices     Array of corresponding entry indices
(declare-buffer-specific picture-indices)
(setq picture-indices (new-array #pictures))

```

```

;   &targets           Target count
(declare-buffer-specific &targets)
(setq &targets 0)

```

```

;   target-names        Array of target names
(declare-buffer-specific target-names)
(setq target-names (new-array #targets))

```

```

;   target-indices      Array of corresponding entry indices
(declare-buffer-specific target-indices)

```

```

(setq target-indices (new-array #targets))

; &entries          Entry count
(declare-buffer-specific &entries)
(setq &entries 0)

; entry-titles      Array of entry titles
(declare-buffer-specific entry-titles)
(setq entry-titles (new-array #entries))

; entry-filenames   Array of entry file names
(declare-buffer-specific entry-filenames)
(setq entry-filenames (new-array #entries))

; entry-buffers     Array of "instantiated" entry buffer names
(declare-buffer-specific entry-buffers)
(setq entry-buffers (new-array #entries))

(instantiate-master-index)

))

; Entry buffer locals:
;
; Initialize the locals of the entry in the current buffer.
(defun (initialize-entry

; entry-title       Title of entry
(declare-buffer-specific entry-title)
(setq entry-title "Unknown Title")

; entry-type        Type of entry
(declare-buffer-specific entry-type)
(setq entry-type "Unknown Type")

; entry-class       Class of entry
(declare-buffer-specific entry-class)
(setq entry-class "Unknown Class")

; &fields           Field count
(declare-buffer-specific &fields)
(setq &fields 0)

; field-names       Array of field names
(declare-buffer-specific field-names)
(setq field-names (new-array #fields))

; field-tops        Array of field top marks
(declare-buffer-specific field-tops)
(setq field-tops (new-array #fields))

; field-bottoms     Array of field bottom marks
(declare-buffer-specific field-bottoms)
(setq field-bottoms (new-array #fields))

(instantiate-entry)

))

```

```

; Master index stuff:
;

(defun (instantiate-master-index $the-title $the-file
  (save-restriction
    (chomp-sub-index "ARTICLES" (push-article $the-title &entries))
    (chomp-sub-index "PICTURES" (push-picture $the-title &entries))
    (chomp-sub-index "TARGETS" (push-target $the-title &entries))
  )
))

(defun (chomp-sub-index
  (narrow-to-sub-index (arg 1))
  (setq $the-title "*bogus*")
  (while (! (error-occurred (search-forward "\"")))
    (set-mark)
    (search-forward "\"")
    (backward-character)
    (setq $the-title (canonicalize (region-to-string)))
    (forward-character)
    (if (! (looking-at "\n"))
      (progn
        (while (looking-at "[ \t]") (forward-character))
        (set-mark)
        (end-of-line)
        (setq $the-file (region-to-string))
        (push-entry $the-title $the-file)
      )
    )
  )
  (arg 2)
)
))

(defun (narrow-to-sub-index $name
  (setq $name (arg 1 ": narrow-to-sub-index (name) "))
  (widen-region)
  (beginning-of-file)
  (if (error-occurred (search-forward (concat "----- " $name " -----\n")))
    (progn ; make new sub-index & narrow to it
      (end-of-file)
      (insert-string "\n----- " $name " -----\n")
    )
  )
  (set-mark)
  (if (error-occurred (search-forward "\n----- "))
    (end-of-file)
    (beginning-of-line)
  )
  (narrow-region)
  (beginning-of-file)
))

; Pass second argument to keep it from folding case
(defun (canonicalize $string &fold
  (setq $string (arg 1 ": canonicalize (string) "))
  (setq &fold (< (nargs) 2))
  (save-window-excursion

```

```

        (switch-to-buffer "*canonicalize*")
        (setq needs-checkpointing 0)
        (erase-buffer)
        (insert-string $string)
        (set-mark)
        (beginning-of-file)
        (if &fold (case-region-lower))
        (error-occurred
         (re-replace-string "[ \\n\\t<>~]+" " "))
        (beginning-of-file)
        (if (looking-at " ") (delete-next-character))
        (end-of-file)
        (error-occurred
         (backward-character)
         (if (looking-at " ") (delete-next-character))))
        (buffer-to-string "*canonicalize*")
    )
))

; Write the arrays back out into the master-index buffer.
(defun (outstantiate-master-index
)

; Index manager interface:

(defun (search-master-indices $title $subindex
    $i $names $indices &last $j &entry $entry-buffer
    (save-excursion
      (save-window-excursion
        (setq $title (canonicalize (arg 1 ": search-master-indices (title) ")))
        (setq $subindex (arg 2 ": search-master-indices (subindex) "))
        (setq &entry 0)
        (setq $entry-buffer "")
        (setq $i (+ &master-indices 1))
        (while (> (-- $i) 0)
          (switch-to-buffer (array master-indices $i))
          (setq $names (execute-mlisp-string (concat $subindex "-names")))
          (setq $indices (execute-mlisp-string (concat $subindex "-indices")))
          (setq &last (execute-mlisp-string (concat "&" $subindex "s")))
          (setq $j 0)
          (while (<= (++ $j) &last)
            (if (= $title (array $names $j))
              (progn (setq &entry (array $indices $j))
                     (if (= (setq $entry-buffer
                                (array entry-buffers &entry))
                            "")
                         (progn
                          (save-excursion
                            (find-entry-file
                             (array entry-filenames &entry))
                            (setq $entry-buffer (current-buffer-name))
                          )
                        )
                     )
              (setq-array entry-buffers &entry
                          $entry-buffer)
            )
            (setq $j &last)
          )
        (setq $i 0)

```

```

    )
  )
)
$entry-buffer
)
)
))

(defun (find-entry-file $relname
  (setq $relname (arg 1 ": find-entry-file (relative name) "))
  (find-file (concat (file-directory-name (current-file-name))
    $relname))
; entry gets instantiated according to file name extension...?
))

; Entry stuff:
;

(defun (push-entry $title $filename
  (setq $title (arg 1 ": push-entry (title) "))
  (setq $filename (arg 2 ": push-entry (filename) "))
  (if (<= (array-size entry-titles) &entries)
    (progn
      (setq entry-titles (grow-array entry-titles))
      (setq entry-filenames (grow-array entry-filenames))
    )
  )
  (setq-array entry-titles (++ &entries) $title)
  (setq-array entry-filenames &entries $filename)
  (setq-array entry-buffers &entries ""))
))

; Article stuff:
;

(defun (instantiate-article
))

(defun (push-article $title &index
  (setq $title (arg 1 ": push-article (title) "))
  (setq &index (arg 2 ": push-article (index) "))
  (if (<= (array-size article-names) &articles)
    (progn
      (setq article-names (grow-array article-names))
      (setq article-indices (grow-array article-indices))
    )
  )
  (setq-array article-names (++ &articles) $title)
  (setq-array article-indices &articles &index))
))

; Picture stuff:
;

(defun (instantiate-picture
))

```

```

(defun (push-picture $title &index
  (setq $title (arg 1 ": push-picture (title) "))
  (setq &index (arg 2 ": push-picture (index) "))
  (if (<= (array-size picture-names) &pictures)
    (progn
      (setq picture-names (grow-array picture-names))
      (setq picture-indices (grow-array picture-indices))
    )
  )
  (setq-array picture-names (++ &pictures) $title)
  (setq-array picture-indices &pictures &index)
))

; Target stuff:
;

(defun (instantiate-target
)

(defun (push-target $title &index
  (setq $title (arg 1 ": push-tartget (title) "))
  (setq &index (arg 2 ": push-tartget (index) "))
  (if (<= (array-size target-names) &targets)
    (progn
      (setq target-names (grow-array target-names))
      (setq target-indices (grow-array target-indices))
    )
  )
  (setq-array target-names (++ &targets) $title)
  (setq-array target-indices &targets &index)
))

; Command interpreter interface:
;
; Commands are defined in a file with the following syntax, which
; specified the number of arguments for each commands, prompts for
; the arguments, which may specify a function to call to read the
; arguments with completion over whatever name space is appropriate.
;
; .link <Title or synonym of link destination:(get-article-name)>
; .picture <Title of picture:(get-picture-name)>
; .target <Title of target:(get-picture-name)>
;      <Title or synonym of link destination:(get-article-name)>

; Utilities:

(defun (grow-array $array $new-array $size $i
  (setq $array (arg 1 ": grow-array (array) "))
  (setq $size (array-size $array))
  (setq $new-array (new-array (+ $size $size)))
  (setq $i 0)
  (while (<= (++ $i) $size)
    (setq-array $new-array $i (array $array $i)))
  $new-array
))

```