;;; cmake-mode.el --- major-mode for editing CMake sources -\*- lexical-binding: t; -\*-

;; Package-Requires: ((emacs "24.1"))

; Distributed under the OSI-approved BSD 3-Clause License. See accompanying

; file Copyright.txt or https://cmake.org/licensing for details.

;------------------------------------------------------------------------------

;;; Commentary:

;; Provides syntax highlighting and indentation for CMakeLists.txt and

;; \*.cmake source files.

;;

;; Add this code to your .emacs file to use the mode:

;;

;; (setq load-path (cons (expand-file-name "/dir/with/cmake-mode") load-path))

;; (require 'cmake-mode)

;------------------------------------------------------------------------------

;;; Code:

;;

;; cmake executable variable used to run cmake --help-command

;; on commands in cmake-mode

;;

;; cmake-command-help Written by James Bigler

;;

(require 'rst)

(require 'rx)

(defcustom cmake-mode-cmake-executable "cmake"

"\*The name of the cmake executable.

This can be either absolute or looked up in $PATH. You can also

set the path with these commands:

(setenv \"PATH\" (concat (getenv \"PATH\") \";C:\\\\Program Files\\\\CMake 2.8\\\\bin\"))

(setenv \"PATH\" (concat (getenv \"PATH\") \":/usr/local/cmake/bin\"))"

:type 'file

:group 'cmake)

;; Keywords

(defconst cmake-keywords-block-open '("BLOCK" "IF" "MACRO" "FOREACH" "ELSE" "ELSEIF" "WHILE" "FUNCTION"))

(defconst cmake-keywords-block-close '("ENDBLOCK" "ENDIF" "ENDFOREACH" "ENDMACRO" "ELSE" "ELSEIF" "ENDWHILE" "ENDFUNCTION"))

(defconst cmake-keywords

(let ((kwds (append cmake-keywords-block-open cmake-keywords-block-close nil)))

(delete-dups kwds)))

;; Regular expressions used by line indentation function.

;;

(defconst cmake-regex-blank "^[ \t]\*$")

(defconst cmake-regex-comment "#.\*")

(defconst cmake-regex-paren-left "(")

(defconst cmake-regex-paren-right ")")

(defconst cmake-regex-closing-parens-line (concat "^[[:space:]]\*\\("

cmake-regex-paren-right

"+\\)[[:space:]]\*$"))

(defconst cmake-regex-argument-quoted

(rx ?\" (\* (or (not (any ?\" ?\\)) (and ?\\ anything))) ?\"))

(defconst cmake-regex-argument-unquoted

(rx (or (not (any space "()#\"\\\n")) (and ?\\ nonl))

(\* (or (not (any space "()#\\\n")) (and ?\\ nonl)))))

(defconst cmake-regex-token

(rx-to-string `(group (or (regexp ,cmake-regex-comment)

?\( ?\)

(regexp ,cmake-regex-argument-unquoted)

(regexp ,cmake-regex-argument-quoted)))))

(defconst cmake-regex-indented

(rx-to-string `(and bol (\* (group (or (regexp ,cmake-regex-token) (any space ?\n)))))))

(defconst cmake-regex-block-open

(rx-to-string `(and symbol-start (or ,@(append cmake-keywords-block-open

(mapcar 'downcase cmake-keywords-block-open))) symbol-end)))

(defconst cmake-regex-block-close

(rx-to-string `(and symbol-start (or ,@(append cmake-keywords-block-close

(mapcar 'downcase cmake-keywords-block-close))) symbol-end)))

(defconst cmake-regex-close

(rx-to-string `(and bol (\* space) (regexp ,cmake-regex-block-close)

(\* space) (regexp ,cmake-regex-paren-left))))

(defconst cmake-regex-token-paren-left (concat "^" cmake-regex-paren-left "$"))

(defconst cmake-regex-token-paren-right (concat "^" cmake-regex-paren-right "$"))

;------------------------------------------------------------------------------

;; Line indentation helper functions

(defun cmake-line-starts-inside-string ()

"Determine whether the beginning of the current line is in a string."

(save-excursion

(beginning-of-line)

(let ((parse-end (point)))

(goto-char (point-min))

(nth 3 (parse-partial-sexp (point) parse-end))

)

)

)

(defun cmake-find-last-indented-line ()

"Move to the beginning of the last line that has meaningful indentation."

(let ((point-start (point))

region)

(forward-line -1)

(setq region (buffer-substring-no-properties (point) point-start))

(while (and (not (bobp))

(or (looking-at cmake-regex-blank)

(cmake-line-starts-inside-string)

(not (and (string-match cmake-regex-indented region)

(= (length region) (match-end 0))))))

(forward-line -1)

(setq region (buffer-substring-no-properties (point) point-start))

)

)

)

;------------------------------------------------------------------------------

;;

;; Indentation increment.

;;

(defcustom cmake-tab-width 2

"Number of columns to indent cmake blocks"

:type 'integer

:group 'cmake)

;;

;; Line indentation function.

;;

(defun cmake-indent ()

"Indent current line as CMake code."

(interactive)

(unless (cmake-line-starts-inside-string)

(if (bobp)

(cmake-indent-line-to 0)

(let (cur-indent)

(save-excursion

(beginning-of-line)

(let ((point-start (point))

(closing-parens-only (looking-at cmake-regex-closing-parens-line))

(case-fold-search t) ;; case-insensitive

token)

;; Search back for the last indented line.

(cmake-find-last-indented-line)

;; Start with the indentation on this line.

(setq cur-indent (current-indentation))

(if closing-parens-only

(let ((open-parens 0))

(while (re-search-forward cmake-regex-token point-start t)

(setq token (match-string 0))

(cond

((string-match cmake-regex-token-paren-left token)

(setq open-parens (+ open-parens 1)))

((string-match cmake-regex-token-paren-right token)

(setq open-parens (- open-parens 1)))))

;; Don't outdent if last indented line has open parens

(unless (> open-parens 0)

(setq cur-indent (- cur-indent cmake-tab-width))))

;; Skip detailed analysis if last indented line is a 'closing

;; parens only line'

(unless (looking-at cmake-regex-closing-parens-line)

;; Search forward counting tokens that adjust indentation.

(while (re-search-forward cmake-regex-token point-start t)

(setq token (match-string 0))

(when (or (string-match cmake-regex-token-paren-left token)

(and (string-match cmake-regex-block-open token)

(looking-at (concat "[ \t]\*" cmake-regex-paren-left))))

(setq cur-indent (+ cur-indent cmake-tab-width)))

(when (string-match cmake-regex-token-paren-right token)

(setq cur-indent (- cur-indent cmake-tab-width)))

))

(goto-char point-start)

;; If next token closes the block, decrease indentation

(when (looking-at cmake-regex-close)

(setq cur-indent (- cur-indent cmake-tab-width))

)

)

)

)

;; Indent this line by the amount selected.

(cmake-indent-line-to (max cur-indent 0))

)

)

)

)

(defun cmake-point-in-indendation ()

(string-match "^[ \\t]\*$" (buffer-substring (line-beginning-position) (point))))

(defun cmake-indent-line-to (column)

"Indent the current line to COLUMN.

If point is within the existing indentation it is moved to the end of

the indentation. Otherwise it retains the same position on the line"

(if (cmake-point-in-indendation)

(indent-line-to column)

(save-excursion (indent-line-to column))))

;------------------------------------------------------------------------------

;;

;; Helper functions for buffer

;;

(defun cmake-unscreamify-buffer ()

"Convert all CMake commands to lowercase in buffer."

(interactive)

(save-excursion

(goto-char (point-min))

(while (re-search-forward "^\\([ \t]\*\\)\\\_<\\(\\(?:\\w\\|\\s\_\\)+\\)\\\_>\\([ \t]\*(\\)" nil t)

(replace-match

(concat

(match-string 1)

(downcase (match-string 2))

(match-string 3))

t))

)

)

;------------------------------------------------------------------------------

;;

;; Navigation / marking by function or macro

;;

(defconst cmake--regex-defun-start

(rx line-start

(zero-or-more space)

(or "function" "macro")

(zero-or-more space)

"("))

(defconst cmake--regex-defun-end

(rx line-start

(zero-or-more space)

"end"

(or "function" "macro")

(zero-or-more space)

"(" (zero-or-more (not-char ")")) ")"))

(defun cmake-beginning-of-defun ()

"Move backward to the beginning of a CMake function or macro.

Return t unless search stops due to beginning of buffer."

(interactive)

(when (not (region-active-p))

(push-mark))

(let ((case-fold-search t))

(when (re-search-backward cmake--regex-defun-start nil 'move)

t)))

(defun cmake-end-of-defun ()

"Move forward to the end of a CMake function or macro.

Return t unless search stops due to end of buffer."

(interactive)

(when (not (region-active-p))

(push-mark))

(let ((case-fold-search t))

(when (re-search-forward cmake--regex-defun-end nil 'move)

(forward-line)

t)))

;------------------------------------------------------------------------------

;;

;; Keyword highlighting regex-to-face map.

;;

(defconst cmake-font-lock-keywords

`((,(rx-to-string `(and symbol-start

(or ,@cmake-keywords

,@(mapcar #'downcase cmake-keywords))

symbol-end))

. font-lock-keyword-face)

(,(rx symbol-start (group (+ (or word (syntax symbol)))) (\* blank) ?\()

1 font-lock-function-name-face)

(,(rx "${" (group (+(any alnum "-\_+/."))) "}")

1 font-lock-variable-name-face t)

)

"Highlighting expressions for CMake mode.")

;------------------------------------------------------------------------------

(defun cmake--syntax-propertize-until-bracket-close (syntax end)

;; This function assumes that a previous search has matched the

;; beginning of a bracket\_comment or bracket\_argument and that the

;; second capture group has matched the equal signs between the two

;; opening brackets

(let\* ((mb (match-beginning 2))

(me (match-end 2))

(cb (format "]%s]" (buffer-substring mb me))))

(save-match-data

(if (search-forward cb end 'move)

(progn

(setq me (match-end 0))

(put-text-property

(1- me)

me

'syntax-table

(string-to-syntax syntax)))

(setq me end)))

(put-text-property

(match-beginning 1)

me

'syntax-multiline

t)))

(defconst cmake--syntax-propertize-function

(syntax-propertize-rules

("\\(#\\)\\[\\(=\*\\)\\["

(1

(prog1 "!" (cmake--syntax-propertize-until-bracket-close "!" end))))

("\\(\\[\\)\\(=\*\\)\\["

(1

(prog1 "|" (cmake--syntax-propertize-until-bracket-close "|" end))))))

;; Syntax table for this mode.

(defvar cmake-mode-syntax-table nil

"Syntax table for CMake mode.")

(or cmake-mode-syntax-table

(setq cmake-mode-syntax-table

(let ((table (make-syntax-table)))

(modify-syntax-entry ?\( "()" table)

(modify-syntax-entry ?\) ")(" table)

(modify-syntax-entry ?# "<" table)

(modify-syntax-entry ?\n ">" table)

(modify-syntax-entry ?$ "'" table)

table)))

;;

;; User hook entry point.

;;

(defvar cmake-mode-hook nil)

;;------------------------------------------------------------------------------

;; Mode definition.

;;

;;;###autoload

(define-derived-mode cmake-mode prog-mode "CMake"

"Major mode for editing CMake source files."

;; Setup jumping to beginning/end of a CMake function/macro.

(set (make-local-variable 'beginning-of-defun-function) #'cmake-beginning-of-defun)

(set (make-local-variable 'end-of-defun-function) #'cmake-end-of-defun)

; Setup font-lock mode.

(set (make-local-variable 'font-lock-defaults) '(cmake-font-lock-keywords))

; Setup indentation function.

(set (make-local-variable 'indent-line-function) 'cmake-indent)

; Setup comment syntax.

(set (make-local-variable 'comment-start) "#")

;; Setup syntax propertization

(set (make-local-variable 'syntax-propertize-function) cmake--syntax-propertize-function)

(add-hook 'syntax-propertize-extend-region-functions #'syntax-propertize-multiline nil t))

; Help mode starts here

;;;###autoload

(defun cmake-command-run (type &optional topic buffer)

"Runs the command cmake with the arguments specified. The

optional argument topic will be appended to the argument list."

(interactive "s")

(let\* ((bufname (if buffer buffer (concat "\*CMake" type (if topic "-") topic "\*")))

(buffer (if (get-buffer bufname) (get-buffer bufname) (generate-new-buffer bufname)))

(command (concat cmake-mode-cmake-executable " " type " " (if topic (shell-quote-argument topic) topic)))

;; Turn of resizing of mini-windows for shell-command.

(resize-mini-windows nil)

)

(shell-command command buffer)

(save-selected-window

(select-window (display-buffer buffer 'not-this-window))

(cmake-mode)

(read-only-mode 1)

(view-mode 1))

)

)

;;;###autoload

(defun cmake-command-run-help (type &optional topic buffer)

"`cmake-command-run' but rendered in `rst-mode'."

(interactive "s")

(let\* ((bufname (if buffer buffer (concat "\*CMake" type (if topic "-") topic "\*")))

(buffer (if (get-buffer bufname) (get-buffer bufname) (generate-new-buffer bufname)))

(command (concat cmake-mode-cmake-executable " " type " " (if topic (shell-quote-argument topic) topic)))

;; Turn of resizing of mini-windows for shell-command.

(resize-mini-windows nil)

)

(shell-command command buffer)

(save-selected-window

(select-window (display-buffer buffer 'not-this-window))

(rst-mode)

(read-only-mode 1)

(view-mode 1))

)

)

;;;###autoload

(defun cmake-help-list-commands ()

"Prints out a list of the cmake commands."

(interactive)

(cmake-command-run-help "--help-command-list")

)

(defvar cmake-commands '() "List of available topics for --help-command.")

(defvar cmake-help-command-history nil "Command read history.")

(defvar cmake-modules '() "List of available topics for --help-module.")

(defvar cmake-help-module-history nil "Module read history.")

(defvar cmake-variables '() "List of available topics for --help-variable.")

(defvar cmake-help-variable-history nil "Variable read history.")

(defvar cmake-properties '() "List of available topics for --help-property.")

(defvar cmake-help-property-history nil "Property read history.")

(defvar cmake-help-complete-history nil "Complete help read history.")

(defvar cmake-string-to-list-symbol

'(("command" cmake-commands cmake-help-command-history)

("module" cmake-modules cmake-help-module-history)

("variable" cmake-variables cmake-help-variable-history)

("property" cmake-properties cmake-help-property-history)

))

(defun cmake-get-list (listname)

"If the value of LISTVAR is nil, run cmake --help-LISTNAME-list

and store the result as a list in LISTVAR."

(let ((listvar (car (cdr (assoc listname cmake-string-to-list-symbol)))))

(if (not (symbol-value listvar))

(let ((temp-buffer-name "\*CMake Temporary\*"))

(save-window-excursion

(cmake-command-run-help (concat "--help-" listname "-list") nil temp-buffer-name)

(with-current-buffer temp-buffer-name

; FIXME: Ignore first line if it is "cmake version ..." from CMake < 3.0.

(set listvar (split-string (buffer-substring-no-properties (point-min) (point-max)) "\n" t)))))

(symbol-value listvar)

))

)

(require 'thingatpt)

(defun cmake-symbol-at-point ()

(let ((symbol (symbol-at-point)))

(and (not (null symbol))

(symbol-name symbol))))

(defun cmake-help-type (type)

(let\* ((default-entry (cmake-symbol-at-point))

(history (car (cdr (cdr (assoc type cmake-string-to-list-symbol)))))

(input (completing-read

(format "CMake %s: " type) ; prompt

(cmake-get-list type) ; completions

nil ; predicate

t ; require-match

default-entry ; initial-input

history

)))

(if (string= input "")

(error "No argument given")

input))

)

;;;###autoload

(defun cmake-help-command ()

"Prints out the help message for the command the cursor is on."

(interactive)

(cmake-command-run-help "--help-command" (cmake-help-type "command") "\*CMake Help\*"))

;;;###autoload

(defun cmake-help-module ()

"Prints out the help message for the module the cursor is on."

(interactive)

(cmake-command-run-help "--help-module" (cmake-help-type "module") "\*CMake Help\*"))

;;;###autoload

(defun cmake-help-variable ()

"Prints out the help message for the variable the cursor is on."

(interactive)

(cmake-command-run-help "--help-variable" (cmake-help-type "variable") "\*CMake Help\*"))

;;;###autoload

(defun cmake-help-property ()

"Prints out the help message for the property the cursor is on."

(interactive)

(cmake-command-run-help "--help-property" (cmake-help-type "property") "\*CMake Help\*"))

;;;###autoload

(defun cmake-help ()

"Queries for any of the four available help topics and prints out the

appropriate page."

(interactive)

(let\* ((default-entry (cmake-symbol-at-point))

(command-list (cmake-get-list "command"))

(variable-list (cmake-get-list "variable"))

(module-list (cmake-get-list "module"))

(property-list (cmake-get-list "property"))

(all-words (append command-list variable-list module-list property-list))

(input (completing-read

"CMake command/module/variable/property: " ; prompt

all-words ; completions

nil ; predicate

t ; require-match

default-entry ; initial-input

'cmake-help-complete-history

)))

(if (string= input "")

(error "No argument given")

(if (member input command-list)

(cmake-command-run-help "--help-command" input "\*CMake Help\*")

(if (member input variable-list)

(cmake-command-run-help "--help-variable" input "\*CMake Help\*")

(if (member input module-list)

(cmake-command-run-help "--help-module" input "\*CMake Help\*")

(if (member input property-list)

(cmake-command-run-help "--help-property" input "\*CMake Help\*")

(error "Not a know help topic.") ; this really should not happen

))))))

)

;;;###autoload

(progn

(add-to-list 'auto-mode-alist '("CMakeLists\\.txt\\'" . cmake-mode))

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

; This file provides cmake-mode.

(provide 'cmake-mode)

;;; cmake-mode.el ends here