[redacted]

Raw Blame

304 lines (260 sloc) | 11.8 KB

```
;;; lang/cc/config.el --- c, c++, and obj-c -*- lexical-binding: t; -*-

(defvar +cc-default-include-paths
  (list "include"
        "includes")
  "A list of default relative paths which will be searched for up from the
current file, to be passed to irony as extra header search paths. Paths can be
absolute. This is ignored if your project has a compilation database.

This is ignored by ccls.")

(defvar +cc-default-header-file-mode 'c-mode
  "Fallback major mode for .h files if all other heuristics fail (in
`+cc-c-c++-objc-mode').")

(defvar +cc-default-compiler-options
  `((c-mode . nil)
    (c++-mode
     . ,(list "-std=c++1z" ; use C++17 draft by default
              (when IS-MAC
                ;; NOTE beware: you'll get abi-inconsistencies when passing
                ;; std-objects to libraries linked with libstdc++ (e.g. if you
                ;; use boost which wasn't compiled with libc++)
                "-stdlib=libc++")))
    (objc-mode . nil))
  "A list of default compiler options for the C family. These are ignored if a
compilation database is present in the project.

This is ignored by ccls.")


;;
;;; Packages

(use-package! cc-mode
  :mode ("\\.mm\\'" . objc-mode)
  ;; Use `c-mode'/`c++-mode'/`objc-mode' depending on heuristics
  :mode ("\\.h\\'" . +cc-c-c++-objc-mode)
  ;; Ensure find-file-at-point recognize system libraries in C modes. It must be
  ;; set up before the likes of irony/lsp are initialized. Also, we use
  ;; local-vars hooks to ensure these only run in their respective major modes,
  ;; and not their derived modes.
  :hook ((c-mode-local-vars c++-mode-local-vars objc-mode-local-vars) . +cc-init-ffap-integration-h)
  ;;; Improve fontification in C/C++ (also see `modern-cpp-font-lock')
  :hook (c-mode-common . rainbow-delimiters-mode)
  :hook ((c-mode c++-mode) . +cc-fontify-constants-h)
  :config
  (set-docsets! 'c-mode "C")
  (set-docsets! 'c++-mode "C++" "Boost")
```

```
  (set-electric! '(c-mode c++-mode objc-mode java-mode) :chars '(?\n ?\} ?\{))
  (set-rotate-patterns! 'c++-mode
    :symbols '(("public" "protected" "private")
               ("class" "struct")))
  (set-ligatures! '(c-mode c++-mode)
    ;; Functional
    ;; :def "void "
    ;; Types
    :null "nullptr"
    :true "true" :false "false"
    :int "int" :float "float"
    :str "std::string"
    :bool "bool"
    ;; Flow
    :not "!"
    :and "&&" :or "||"
    :for "for"
    :return "return"
    :yield "#require")

  ;; HACK Suppress 'Args out of range' error in when multiple modifications are
  ;;      performed at once in a `c++-mode' buffer, e.g. with `iedit' or
  ;;      multiple cursors.
  (undefadvice! +cc--suppress-silly-errors-a (orig-fn &rest args)
    :around #'c-after-change-mark-abnormal-strings
    (ignore-errors (apply orig-fn args)))

  ;; Custom style, based off of linux
  (setq c-basic-offset tab-width
        c-backspace-function #'delete-backward-char)

  (c-add-style
   "doom" '((c-comment-only-line-offset . 0)
            (c-hanging-braces-alist (brace-list-open)
                                    (brace-entry-open)
                                    (substatement-open after)
                                    (block-close . c-snug-do-while)
                                    (arglist-cont-nonempty))
            (c-cleanup-list brace-else-brace)
            (c-offsets-alist
             (knr-argdecl-intro . 0)
             (substatement-open . 0)
             (substatement-label . 0)
             (statement-cont . +)
             (case-label . +)
             ;; align args with open brace OR don't indent at all (if open
             ;; brace is at eolp and close brace is after arg with no trailing
             ;; comma)
             (brace-list-intro . 0)
             (brace-list-close . -)
             (arglist-intro . +)
             (arglist-close +cc-lineup-arglist-close 0)
             ;; don't over-indent lambda blocks
             (inline-open . 0)
             (inlambda . 0)
             ;; indent access keywords +1 level, and properties beneath them
             ;; another level
             (access-label . -)
             (inclass +cc-c++-lineup-inclass +)
             (label . 0))))

  (when (listp c-default-style)
    (setf (alist-get 'other c-default-style) "doom"))

  (after! ffap
    (add-to-list 'ffap-alist '(c-mode . ffap-c-mode))))


(use-package! modern-cpp-font-lock
  :hook (c++-mode . modern-c++-font-lock-mode))

```

```
(use-package! irony
  :unless (featurep! +lsp)
  :commands irony-install-server
  ;; Initialize compilation database, if present. Otherwise, fall back on
  ;; `+cc-default-compiler-options'.
  :hook (irony-mode . +cc-init-irony-compile-options-h)
  ;; Only initialize `irony-mode' if the server is available. Otherwise fail
  ;; quietly and gracefully.
  :hook ((c-mode-local-vars c++-mode-local-vars objc-mode-local-vars) . +cc-init-irony-mode-maybe-h)
  :preface (setq irony-server-install-prefix (concat doom-etc-dir "irony-server/"))
  :config
  (defun +cc-init-irony-mode-maybe-h ()
    (if (file-directory-p irony-server-install-prefix)
        (irony-mode +1)
      (message "Irony server isn't installed")))

  (setq irony-cdb-search-directory-list '("." "build" "build-conda"))

  (use-package! irony-eldoc
    :hook (irony-mode . irony-eldoc))

  (use-package! flycheck-irony
    :when (featurep! :checkers syntax)
    :config (flycheck-irony-setup))

  (use-package! company-irony
    :when (featurep! :completion company)
    :init (set-company-backend! 'irony-mode '(:separate company-irony-c-headers company-irony))
    :config (require 'company-irony-c-headers)))


;;
;; Major modes

(after! cmake-mode
  (set-docsets! 'cmake-mode "CMake"))

(use-package! company-cmake  ; for `cmake-mode'
  :when (featurep! :completion company)
  :after cmake-mode
  :config (set-company-backend! 'cmake-mode 'company-cmake))


(use-package! demangle-mode
  :hook llvm-mode)


(use-package! company-glsl  ; for `glsl-mode'
  :when (featurep! :completion company)
  :after glsl-mode
  :config (set-company-backend! 'glsl-mode 'company-glsl))


;;
;; Rtags Support

(use-package! rtags
  :unless (featurep! +lsp)
  ;; Only initialize rtags-mode if rtags and rdm are available.
  :hook ((c-mode-local-vars c++-mode-local-vars objc-mode-local-vars) . +cc-init-rtags-maybe-h)
  :preface (setq rtags-install-path (concat doom-etc-dir "rtags/"))
  :config
  (defun +cc-init-rtags-maybe-h ()
    "Start an rtags server in c-mode and c++-mode buffers.
If rtags or rdm aren't available, fail silently instead of throwing a breaking error."
    (and (require 'rtags nil t)
         (rtags-executable-find rtags-rdm-binary-name)
         (rtags-start-process-unless-running)))

  (setq rtags-autostart-diagnostics t
```

```
        rtags-use-bookmarks nil
        rtags-completions-enabled nil
        rtags-display-result-backend
        (cond ((featurep! :completion ivy)  'ivy)
              ((featurep! :completion helm) 'helm)
              ('default))
        ;; These executables are named rtags-* on debian
        rtags-rc-binary-name
        (or (cl-find-if #'executable-find (list rtags-rc-binary-name "rtags-rc"))
            rtags-rc-binary-name)
        rtags-rdm-binary-name
        (or (cl-find-if #'executable-find (list rtags-rdm-binary-name "rtags-rdm"))
            rtags-rdm-binary-name)
        ;; If not using ivy or helm to view results, use a pop-up window rather
        ;; than displaying it in the current window...
        rtags-results-buffer-other-window t
        ;; ...and don't auto-jump to first match before making a selection.
        rtags-jump-to-first-match nil)

  (set-lookup-handlers! '(c-mode c++-mode)
    :definition #'rtags-find-symbol-at-point
    :references #'rtags-find-references-at-point)

  ;; Use rtags-imenu instead of imenu/counsel-imenu
  (define-key! (c-mode-map c++-mode-map) [remap imenu] #'+cc/imenu)

  ;; Ensure rtags cleans up after itself properly when exiting Emacs, rather
  ;; than display a jarring confirmation prompt for killing it.
  (add-hook! 'kill-emacs-hook (ignore-errors (rtags-cancel-process)))

  (add-hook 'rtags-jump-hook #'better-jumper-set-jump)
  (add-hook 'rtags-after-find-file-hook #'recenter))


;;
;; LSP

(when (featurep! +lsp)
  (add-hook! '(c-mode-local-vars-hook
               c++-mode-local-vars-hook
               objc-mode-local-vars-hook)
             #'lsp!)

  (map! :after ccls
        :map (c-mode-map c++-mode-map)
        :n "C-h" (cmd! (ccls-navigate "U"))
        :n "C-j" (cmd! (ccls-navigate "R"))
        :n "C-k" (cmd! (ccls-navigate "L"))
        :n "C-l" (cmd! (ccls-navigate "D"))
        (:localleader
         :desc "Preprocess file"        "lp" #'ccls-preprocess-file
         :desc "Reload cache & CCLS"    "lf" #'ccls-reload)
        (:after lsp-ui-peek
         (:localleader
          :desc "Callers list"          "c" #'+cc/ccls-show-caller
          :desc "Callees list"          "C" #'+cc/ccls-show-callee
          :desc "References (address)"  "a" #'+cc/ccls-show-references-address
          :desc "References (not call)" "f" #'+cc/ccls-show-references-not-call
          :desc "References (Macro)"    "m" #'+cc/ccls-show-references-macro
          :desc "References (Read)"     "r" #'+cc/ccls-show-references-read
          :desc "References (Write)"    "w" #'+cc/ccls-show-references-write)))

  (when (featurep! :tools lsp +eglot)
    ;; Map eglot specific helper
    (map! :localleader
          :after cc-mode
          :map c++-mode-map
          :desc "Show type inheritance hierarchy" "ct" #'+cc/eglot-ccls-inheritance-hierarchy)

    ;; NOTE : This setting is untested yet
    (after! eglot
```

```
      ;; IS-MAC custom configuration
      (when IS-MAC

        (add-to-list 'eglot-workspace-configuration
                     `((:ccls . ((:clang . ,(list :extraArgs ["-isystem/Library/Developer/CommandLineTools/usr/include/c++/v1"
                                                              "-isystem/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr
                                                              "-isystem/usr/local/include"]
                                                  :resourceDir (cdr (doom-call-process "clang" "-print-resource-dir"))))))))

(use-package! ccls
  :when (featurep! +lsp)
  :unless (featurep! :tools lsp +eglot)
  :hook (lsp-lens-mode . ccls-code-lens-mode)
  :init
  (defvar ccls-sem-highlight-method 'font-lock)
  (after! projectile
    (add-to-list 'projectile-globally-ignored-directories ".ccls-cache")
    (add-to-list 'projectile-project-root-files-bottom-up ".ccls-root")
    (add-to-list 'projectile-project-root-files-top-down-recurring "compile_commands.json"))
  ;; Avoid using `:after' because it ties the :config below to when `lsp-mode'
  ;; loads, rather than `ccls' loads.
  (after! lsp-mode (require 'ccls))
  :config
  (set-evil-initial-state! 'ccls-tree-mode 'emacs)
  ;; Disable `ccls-sem-highlight-method' if `lsp-enable-semantic-highlighting'
  ;; is nil. Otherwise, it appears ccls bypasses it.
  (setq-hook! 'lsp-configure-hook
    ccls-sem-highlight-method (if lsp-enable-semantic-highlighting
                                  ccls-sem-highlight-method))
  (when (or IS-MAC IS-LINUX)
    (let ((cpu-count-command (cond (IS-MAC '("sysctl" "-n" "hw.ncpu"))
                                   (IS-LINUX '("nproc"))
                                   (t (error "unreachable code")))))
      (setq ccls-initialization-options
            `(:index (:trackDependency 1
                      :threads ,(max 1 (/ (string-to-number (cdr (apply #'doom-call-process cpu-count-command))) 2)))))))
  (when IS-MAC
    (setq ccls-initialization-options
          (append ccls-initialization-options
                  `(:clang ,(list :extraArgs ["-isystem/Library/Developer/CommandLineTools/usr/include/c++/v1"
                                              "-isystem/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include"
                                              "-isystem/usr/local/include"]
                                  :resourceDir (cdr (doom-call-process "clang" "-print-resource-dir")))))))
```