Expliquer ce code: function! s:print(group) " (: let l:color = g:novum#colors[a:group] exec printf('hi novum_%s_hi guifg=%s ctermfg=%d', a:group, g:novum#xterm_colors[l:color], l:color) if a:group ==# 'bg' exec printf('hi novum_%s_hi guibg=white ctermbg=255', a:group) endif exec printf('echohl novum_%s_hi', a:group) let l:def = g:novum#defaults[a:group] let l:i = get(g:, 'novum_'.a:group, l:def) echom printf('%s[%d] -> %s %s -> default%s', a:group, l:i, l:color, g:novum#xterm_colors[l:color], l:i ==# l:def ? " : printf(': %s[%d]', a:group, l:def)) exec printf('hi clear novum_%s_hi', a:group) echohl NONE endfunction " :) function! novum#color_set(bang, ...) " (: if !a:0 let l:group = " if a:bang for l:key in keys(g:novum#colors) if exists('g:novum_'.l:key) unlet g:['novum_'.l:key] endif endfor else echohl ErrorMsg echom 'USAGE: :Novum[!] [group [[+-]num]]' echohl NONE return endif else let l:bang = a:bang let l:args = split(a:1) let l:group = l:args[0] if l:group[-1:] ==# '!' let l:bang = 1 let l:group = l:group[:-2] endif if len(l:args) > 1 let l:val = l:args[1] if l:val[0] =~# '[+-]' let l:num = get(g:, 'novum_'.l:group, g:novum#defaults[l:group]) + eval(l:val[0] . str2nr(l:val[1:])) else let l:num = str2nr(l:val) endif let g:['novum_'.l:group] = min([g:novum#maxes[l:group] - 1, max([0, l:num])]) else if l:bang if exists('g:novum_'.l:group) unlet g:['novum_'.l:group] endif else call s:print(l:group) return endif endif endif let l:top = line('w0') redraw! mkview colorscheme novum if expand('%:p:t') ==# 'novum.vim' silent edit normal! zv endif loadview let l:newtop = line('w0') if l:top !=# l:newtop execute 'normal! '.repeat("\<c-y>", l:newtop - l:top) endif redraw! if empty(l:group) echom 'Novum defaults reset' else call s:print(l:group) endif endfunction " :) function! novum#color_set_compl(arglead, cmdline, curpos) abort " (: let l:args = split(a:cmdline, '\s\+') let l:argn = len(l:args) if l:argn ==# 1 return keys(g:novum#maxes) elseif l:argn ==# 2 if a:cmdline =~# '\s\+$' return map(range(0, g:novum#maxes[l:args[1]]), { _,v -> string(v) }) endif return filter(keys(g:novum#maxes), { _,v -> stridx(v, a:arglead) ==# 0 }) elseif l:argn ==# 3 if a:cmdline !~# '\s\+$' return map(range(0, g:novum#maxes[l:args[1]]), { _,v -> string(v) }) endif endif return [] endfunction " :) function! novum#setup_colors(init) " (: if expand('%:p') !~# '\/colors\/novum\.vim$' || g:colors_name !=# 'novum' || get(b:, 'novum_no_setup') return endif nnoremap <buffer> <silent>) <c-a>:update<cr> nnoremap <buffer> <silent> (<c-x>:update<cr> if !get(b:, 'novum_colors_setup') autocmd BufWinEnter <buffer> call novum#setup_colors(0) autocmd BufWinLeave <buffer> call clearmatches() autocmd BufWritePost <buffer> if !get(b:, 'novum_no_setup')|try|colo novum|call novum#setup_colors(0)|catch|redraw!|echoerr 'Error loading novum.vim'|endtry|endif let b:novum_colors_setup = 1 endif call clearmatches() let l:linenr = 0 for l:line in getline(1, '$') let l:linenr += 1 if l:line =~# '^H\s\+\w' call matchaddpos(split(l:line, '\s\+')[1], [[l:linenr]]) elseif l:line =~# '^\s*call\s\+s:\%(set\|link\)Color\s*(' let l:name = substitute(l:line, '^\s*call\s\+s:\%(set\|link\)Color(\(["'']\)\(.\{-}\)\1.*', '\2', '') let l:group = 'novum_'.l:name let l:col = get(g:novum#colors, l:name, -1) if l:col ==# -1 continue endif exec printf('hi %s guifg=%s ctermfg=%d', l:group, g:novum#xterm_colors[l:col], l:col) if l:name ==# 'bg' exec printf('hi %s guibg=%s ctermbg=%d', l:group, g:novum#xterm_colors[g:novum#colors.fg], g:novum#colors.fg) endif let l:p1 = stridx(l:line, """) let l:p2 = stridx(l:line, """, l:p1 + 1) call matchaddpos(l:group, [[l:linenr, l:p1 + 1, l:p2 - l:p1 + 1]]) endif endfor " if if &termguicolors let [l:linenr,l:colnr] = searchpos('let\s\+g:novum#xterm_colors\s*=\s*\[\s*\zs\S', 'n') let l:colnr += 1 let l:line = split(getline(l:linenr)[l:colnr-1 : -4], "', "') for l:i in range(256) let l:col = 'xterm_col_'.l:i execute printf('hi %s guifg=%s', l:col, l:line[l:i]) call matchaddpos(l:col, [[l:linenr, l:colnr + 11 * l:i, 7]]) let l:nums = printf('%3d', l:i) call matchaddpos('Conceal', [[l:linenr, l:colnr + 11 * l:i - 4, 1]], 11, -1, { 'conceal': l:nums[0] }) call matchaddpos('Conceal', [[l:linenr, l:colnr + 11 * l:i - 3, 1]], 11, -1, { 'conceal': l:nums[1] }) call matchaddpos('Conceal', [[l:linenr, l:colnr + 11 * l:i - 2, 1]], 11, -1, { 'conceal': l:nums[2] }) call matchaddpos('Conceal', [[l:linenr, l:colnr + 11 * l:i - 1, 1]], 11, -1, { 'conceal': ':' })

endfor endif if !a:init if exists('#User#NovumColorUpdate') doautocmd User NovumColorUpdate endif endif endfunction " :) " vim: fdm=marker fmr=\(\:,\:\) fdl=0:

Ce code est une implémentation d'un thème de couleurs pour Vim. Il définit trois fonctions principales, `s:print`, `novum#color_set` et `novum#color_set_compl`, qui font respectivement les tâches suivantes :

- `s:print` : Cette fonction imprime les informations sur les couleurs actuelles d'un groupe donné (par exemple, le premier plan et le fond).
- `novum#color_set` : Cette fonction permet de définir les couleurs pour un groupe donné (par exemple, premier plan et fond) ou de les remettre à leur valeur par défaut. La fonction peut être appelée avec ou sans arguments, avec ou sans le modificateur `!`. Si aucun argument n'est fourni avec le modificateur `!`, toutes les couleurs sont remises à leur valeur par défaut. Sinon, si un argument est fourni, il peut être utilisé pour définir une nouvelle couleur pour le groupe donné.
- `novum#color_set_compl` : Cette fonction est utilisée pour compléter les arguments pour la fonction `novum#color_set`. Elle fournit des suggestions pour les groupes de couleurs et les valeurs de couleurs disponibles pour le groupe donné.

En général, ce code permet de personnaliser les couleurs d'affichage dans Vim en définissant des groupes de couleurs (par exemple, premier plan et fond) et en leur attribuant des valeurs spécifiques. Les couleurs peuvent être définies manuellement ou remises à leur valeur par défaut.