```
/*********************************************************************
 *                                                                   *
 * Macro   complete.the   V1.8 (THE)                                 *
 *                                                                   *
 * Author:   Pablo Garcia-Abia (Pablo.Garcia@ciemat.es)              *
 * Date:     19/08/2003                                              *
 *                                                                   *
 *                                                                   *
 * Syntax:   complete FILE|KEY|* alternative_command                 *
 *                                                                   *
 *                                                                   *
 * Description:                                                      *
 *                                                                   *
 *    This macro is intended to complete  either  filenames typed in the   *
 *    command  line  (Option  FILE) or  keywords  typed in the file body   *
 *    (Option KEY). The option '*' holds for both FILE and KEY.      *
 *                                                                   *
 *                                                                   *
 *                                                                   *
 *    Filename completion:   complete FILE alternative_command       *
 *    -------------------                                            *
 *                                                                   *
 *    Using the  option  FILE, this  macro  completes  filenames  in the   *
 *    command  line,  whenever  the  first  command  typed is one of the   *
 *    following:                                                     *
 *                                                                   *
 *              XEDIT  FFILE  FILE  MACRO     LS                     *
 *              THE    SSAVE  SAVE  OSREDIR   DIRECTORY              *
 *              EDIT   GET    PUTD  STATUS                           *
 *                                                                   *
 *    Otherwise,    an    alternative    command    is    executed.   If   *
 *    alternative_command is HELP, this help is shown.               *
 *                                                                   *
 *    To use this macro define a key in the following way:           *
 *                                                                   *
 *         DEFINE keyname MACRO COMPLETE FILE alternative_command    *
 *                                                                   *
 *    For example:                                                   *
 *                                                                   *
 *         define TAB macro complete file sos tabfieldf              *
 *                                                                   *
 *    In this example, the TAB key will complete filenames when typed IN   *
 *    the  CMDline  (after  XEDIT,  THE or any other  allowed  command).   *
 *    Otherwise it will execute "sos tabfieldf".                     *
 *                                                                   *
 *                                                                   *
 * Completion procedure:                                             *
 *                                                                   *
 *    Let's asume we are typing in the CMDline anything after one of the   *
 *    allowed  commands.  The  following  substitutions  are made before   *
 *    proceeding to the filename completion itself.                  *
 *                                                                   *
 *    A double  equal  sign (==) is  substituted  by the whole  filename   *
 *    (without path) of the file being edited.                       *
 *                                                                   *
 *    A  single  equal  sign  (=) in a given  field  (delimitied  by the   *
 *    selected  separator) is substituted by the corresponding  field of   *
 *    the filename of the file being edited.                         *
 *                                                                   *
 *    The separator is taken as the character  preceding  the equal sign   *
 *    unless it is the first in the string.  In that case, the character   *
```

```
 *    following the '=' is taken as  separator.  The dot '.'  is used as   *
 *    a separator in ill cases.                                            *
 *                                                                         *
 *    The characters =/ or =\ (depending on the OS) are  substituted  by   *
 *    the path of the file being edited.                                   *
 *                                                                         *
 *    More sophisticated completion may be implemented in the future for   *
 *    other commands, eventually.                                          *
 *                                                                         *
 *                                                                         *
 *                                                                         *
 *    Keyword completion:   complete KEY alternative_command               *
 *    ------------------                                                   *
 *                                                                         *
 *    Using the option KEY, this macro  completes  keywords typed in the   *
 *    body of the file. Blah, blah, blah...                                *
 *    Otherwise,    an    alternative    command    is    executed.   If   *
 *    alternative_command is HELP, this help is shown.                     *
 *                                                                         *
 *    To use this macro define a key in the following way:                 *
 *                                                                         *
 *         DEFINE keyname MACRO COMPLETE KEY alternative_command           *
 *                                                                         *
 *    For example:                                                         *
 *                                                                         *
 *         define C-Y macro complete key cmatch                            *
 *                                                                         *
 *    In this example, the C-Y key  combination  will complete  keywords   *
 *    being typed in the file.  Otherwise it will execute "cmatch".        *
 *                                                                         *
 *                                                                         *
 * Completion procedure:                                                   *
 *                                                                         *
 * Version history:                                                        *
 *                                                                         *
 * 1.8  19/08/2003  - Add some DOS support                                 *
 * 1.7  19/11/1999  - Add Keyword completion and 'completion type' option  *
 * 1.6  01/08/1999  - Added =/ (=\) as a tag for PATH completion           *
 *                  - Added list of files to be excluded from filename     *
 *                    completion                                           *
 *                  - Added HELP option                                    *
 * 1.5  22/02/1999  replace exit by return                                 *
 * 1.4  15/01/1998  minor changes                                          *
 * 1.3  14/01/1998  - follow up directory tree when looking for completions*
 *                    UNIX ok, to be checked in OS2.                       *
 *                    Not implemented in other OS.                         *
 *                  - FJW: OS-dependent corrections (thanks)               *
 *                  - substitution of '=' and '=='                         *
 * 1.2  11/12/1997  added list of commands allowed for file completion     *
 * 1.1  10/12/1997  bug on ambiguities output corrected                    *
 * 1.0  10/12/1997  first version                                          *
 *                                                                         *
 * Bugs, comments and/or questions to Pablo.Garcia@ciemat.es               *
 *************************************************************************/

  Parse Upper Arg comp_type alt_command


/* Get help */

  If translate(comp_type) == 'HELP' Then Do
     set rexxoutput file 99999
```

```
      h_st = 0
      Do i=1 Until h_st=1
         If substr(sourceline(i),1,20) = "/*******************" Then h_st=1
      End

      h_end = 0
      Do i=h_st+1 While h_end=0
         If substr(sourceline(i),58,20) = "*******************/" Then h_end=1
         Else say strip(substr(sourceline(i),4,68),'T')
      End
      say

      Return
   End


/* Check input arguments */

   If comp_type <> 'FILE' & comp_type <> 'KEY' & comp_type <> '*' Then Do
      EMSG 'Invalid completion type:' comp_type
      Return
   End


/* Start the job */

  "PRESERVE"

  "EXTRACT /CMDLINE/CURSOR/MSGLINE/LSCREEN"

  flag_file = 0
  flag_key  = 0

  If comp_type == 'FILE' | comp_type == '*' Then Do
     If cursor.3 = -1 & words(cmdline.3) > 0 Then Call COMPLETE
  End

  If comp_type == 'KEY'  | comp_type == '*' Then Do
     Call Get_Parser

     If cursor.3 > 0 & parser <> '' Then Call COMPLETE_CMD
     Else EMSG 'No parser defined.'
  End

  If flag_file == 0 & flag_key == 0 Then alt_command

  "RESTORE"

  Return



/*********************************************************************/
/* Complete command */

COMPLETE_CMD:

      flag_key = 1

      delim = " ,;:'`[]{}()<>|!@%^&?*+="
```

```
      delim = delim||'"'            /* add " to the list */


/* Get list of keywords to complete */

      If datatype(n_tld) <> 'NUM' Then Do

         tld_file = macropath.1'/'||parser||'.tld'

         If state(tld_file) > 0 Then Do
            EMSG 'File' tld_file 'does not exist.'
            Return
         End

         If open(tld_file,'r') = 0 Then Do
            Say 'Problems opening file' tld_file'.'
            Return
         End

         Call Crea_TLD_list

         If close(tld_file) = 0 Then Do
            Say 'Problems closing file' tld_file'.'
            Return
         End
      End

/* get keyword being typed */

      'EXTRACT /LINE'
      ':'cursor.3
      'EXTRACT /CURLINE'

      this_line = strip(substr(curline.3,1,cursor.4-1),'L')

      Do j=length(this_line) To 1 By -1
         char = substr(this_line,j,1)
         idelfo = 0
         Do d=1 To length(delim)
            If char == substr(delim,d,1) Then Do
              idelfo=1
              Leave
            End
         End
         If idelfo == 1 Then Leave
      End

      this_line = substr(this_line,j+1)
      ltl       = length(this_line)
      n_mat     = 0

      If ltl > 0 Then Do
         Do i=1 To n_tld
            If translate(substr(x_cmd.i,1,ltl)) == translate(this_line) Then Do
               n_mat       = n_mat + 1
               cmd_m.n_mat = x_cmd.i
            End
         End

         Select
            When n_mat == 0 Then MSG 'No matching strings.'
            When n_mat == 1 Then Do
```

```
               MSG 'Unique matching string.'
               text_new = substr(cmd_m.1,ltl+1)
               'CLOCATE :'cursor.4
               'CINSERT' text_new
               'CLOCATE :0'
               'CURSOR FILE' cursor.3 cursor.4+length(text_new)
            End
            Otherwise Do
               'set msgline ON 2' n_mat 'OVERLAY'
               Do i=1 To n_mat
                  MSG cmd_m.i
               End
            End
         End
      End

      ':'line.1
Return



/* Get parser (tld) */

Get_Parser:

      'EXTRACT /AUTOCOLOR/MACROPATH/FTYPE'

      Do i=1 to autocolor.0 Until ifo=1
         type   = word(autocolor.i,1)
         parser = word(autocolor.i,2)

         Parse var type with . '.' type

         If type = ftype.1 Then ifo = 1
      End

      upper  = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
      lower  = "abcdefghijklmnopqrstuvwxyz"
      parser = translate(parser,lower,upper)

      If type = '' Then parser = ''

Return



/* Create list of TLD entries */

Crea_TLD_list:

/* Search for commands in sections:  ':keyword' and ':function' */

      x_cmd. = ''
      n_tld  = 0

      Do nline = 1 Until word(str,1) == ':keyword' | word(str,1) == ':function'
         str = linein(tld_file)
         If lines(tld_file)==0 Then Leave
      End

      Do nline = 1 Until substr(str,1,1) == ':' & word(str,1) <> ':keyword' & word(str,1) <>
':function'
```

```
         str = linein(tld_file)

         If word(str,1) <> ':keyword' & word(str,1) <> ':function' Then Do
            n_tld = n_tld + 1
            x_cmd.n_tld = word(str,1)
         End

         If lines(tld_file)==0 Then Leave
      End

Return



/*********************************************************************/
/* Filename completion */

COMPLETE:

flag_file = 1

/* List of files to be excluded from completion */
/*           -exe---lib--compressed-- -------- PAW ------------ --------------- graphics
-------------- */
list_excl = ".exe .o .a .gz .tgz .zip .hbook .hsto .hz .ntp .evt .ps .eps .dvi .gif .jpeg .jpg
.xbm .xpm"

os = version.3()      /* Credits for Franz-Josef Wirtz */

Select
   When (os == "OS2") Then Do
      dirsep = "\"
/*      dircmd = "ls -d" /* or "dir /B" if 'ls' is not available */ */
      dircmd = "ls -dp"  /* slash after dir name ??? */
   End

   When (os == "WIN32") Then Do
      dirsep = "\"
      dircmd = "dir /B"
   End

   When (os == "DOS") Then Do
      dirsep = "/"
      dircmd = "dir /B"
   End

   Otherwise Do
      dirsep = "/"
      dircmd = "/bin/ls -dp"
   End
End                   /* Credits for Franz-Josef Wirtz (END) */

ori = delword(cmdline.3,words(cmdline.3))
cmd = translate(word(cmdline.3,1))

If words(cmdline.3) > 1 Then str = word(cmdline.3,words(cmdline.3))
Else                         str = ""


/* Deal with '==' in input string */

fn   = filename.1()
```

```
eqeq = pos( '==', str)     /* '==' is always the whole filename */

If eqeq > 0 Then Do
   str = delstr(str,eqeq,2)
   str = insert(fn,str,eqeq-1)
   'CMSG' ori||str
End


/* Deal with '=/' (or '=\') in input string */

fp     = fpath.1()
dirdir = dirsep||dirsep

eqsl = pos('='dirsep, str)           /* '=/' is always the path */

If eqsl > 0 Then Do
   str = delstr(str,eqsl,2)
   str = insert(fp,str,eqsl-1)

   eqdd = pos(dirdir, str)
   If eqdd > 0 Then str = delstr(str,eqdd,1)

   'CMSG' ori||str
End


/* Deal with '=' in input string */

equ = pos( '=', str)

If equ > 0 Then Do

    /* get separator: character before the '=' sign, or after it if '=' is the
                      first char in str. If not char, dot '.' is assumed */

    If      equ == 1           Then eq_sep = substr(str,equ+1,1)
    Else                            eq_sep = substr(str,equ-1,1)

    If eq_sep = " " Then eq_sep = "."

    /* split filename according to 'eq_sep' */

    ndot = 0
    nstr = 1
    fn.  = ''

    If substr(fn,1,1) == eq_sep Then dot = pos(eq_sep,fn,2)
    Else                             dot = pos(eq_sep,fn)

    Do While dot > 0
       ndot    = ndot+1
       fn.ndot = substr(fn,nstr,dot-nstr)
       nstr    = dot+1
       dot     = pos(eq_sep,fn,nstr)
    End

    ndot    = ndot+1
    fn.ndot = substr(fn,nstr)

    /* substitute '=' by its counterparts */
```

```
    nstr = 1
    nequ = 0
    str. = ''

    If substr(str,1,1) == eq_sep Then dot = pos(eq_sep,str,2)
    Else                              dot = pos(eq_sep,str)

    Do While dot > 0
       nequ     = nequ+1

       If substr(str,nstr,dot-nstr) == '=' Then Do
          str = delstr(str,nstr,1)
          str = insert(fn.nequ,str,nstr-1)
       End

       nstr     = dot+1
       dot      = pos(eq_sep,str,nstr)
    End

    If substr(str,nstr) == '=' Then Do
       nequ     = nequ+1
       str = delstr(str,nstr,1)
       str = insert(fn.nequ,str,nstr-1)
    End

   'CMSG' ori||str
End


/* Match one of these commands */

list_cmd = "XEDIT THE EDIT FFILE FILE SAVE SSAVE LS DIRECTORY GET PUTD MACRO OSREDIR STATUS"
list_abb = "1     3   1    2     4    4    2     2  3         3   3    5     3       4     "

match = 0

Do i=1 To words(list_cmd)
   If 'ABBREV'(word(list_cmd,i),cmd,word(list_abb,i)) Then Do
     match = 1
     Leave
   End
End

/* no command matched */

If match <> 1 Then Do
      alt_command
      return
End

/* one command matched */

slash = lastpos(dirsep,str)

x = run_os(dircmd str"*", ,"stdout.", "stderr.")

len_stdout = length(stdout.1)

If substr(stdout.1,len_stdout) == dirsep Then ,
     lout = lastpos(dirsep,substr(stdout.1,1,len_stdout-1))
Else lout = lastpos(dirsep,stdout.1)
```

```
/* No ambiguities */

If stderr.0 > 0 Then EMSG stderr.1
Else If stdout.0 = 1 Then
      If slash > 0 Then 'CMSG' ori||substr(str,1,slash)||substr(stdout.1,lout+1)
      Else              'CMSG' ori||stdout.1
Else Do


/* Exclude files from exclusion list */

  If words(list_excl) > 0 Then Do

     exclu.0 = words(list_excl)
     Do iw=1 To exclu.0
        exclu.iw = word(list_excl,iw)
     End

     flag.  = 0
     nexclu = 0

     Do jj=1 To stdout.0
        Do iw=1 To exclu.0
           If flag.jj = 0 Then Do
              la = lastpos(exclu.iw,stdout.jj)
              le = length(stdout.jj)-length(exclu.iw)+1
              If la>0 & la >= le Then flag.jj = 1
           End
        End
     End

     nskip = 0

     Do jj=1 To stdout.0
        If flag.jj = 1 Then
           nskip = nskip+1
        Else Do
           k = jj-nskip
           flag.k   = flag.jj
           stdout.k = stdout.jj
        End
     End

     stdout.0 = stdout.0-nskip

  End


/* List ambiguities */

  trunc = 1
  Do itr=5 By 5 Until trunc = 0 | itr > lscreen.1-10

     nmax   = min(itr,lscreen.1-10)
     ncols  = format(stdout.0/nmax-0.5,,0)+1
     nlines = min(nmax,stdout.0)
     long.  = 0
     short  = 9999
     colu   = 0
     line   = 0
     line.  = ""
     width  = lscreen.2
```

```
      "set msgline ON 2" nmax "OVERLAY"

      Do i=1 To stdout.0
         line = line+1
         If line > nmax Then Do ; line = 1 ; colu = colu+1 ; End

         len_stdout = length(stdout.i)

         If substr(stdout.i,len_stdout) == dirsep Then ,
              last = lastpos(dirsep,substr(stdout.i,1,len_stdout-1))+1
         Else last = lastpos(dirsep,stdout.i)+1

         new.i = substr(stdout.i,last)

         short     = min(short,     length(new.i))
         long.colu = max(long.colu, length(new.i))

      End

      colu = 0
      line = 0
      Do i=1 To stdout.0
         line = line+1
         If line > nmax Then Do ; line = 1 ; colu = colu+1 ; End
         line.line = line.line||substr(new.i,1,long.colu+4)
      End

      trunc = 0
      Do i=1 To nlines
         If length(line.i) > width Then trunc = trunc+1
      End

    End

    Do i=1 To nlines
       MSG substr(line.i,1,width)
    End

    If trunc > 0 Then Do
       "set msgline ON 2" nmax+2 "OVERLAY"
        MSG
        MSG "Too many files to be displayed:" trunc "lines truncated..."
    End

  /* Partial Completion (up to ambiguity) */

    If slash > 0 Then len = length(substr(str,slash+1))
    Else len = length(str)

    Do j=len+1 To short
      char = substr(new.1,j,1)
      inall = 1
      Do k=2 To stdout.0
        If substr(new.k,j,1) <> char Then Do
           inall = 0
           Leave
        End
      End
      If inall = 1 Then
        If slash > 0 Then 'CMSG' ori||substr(str,1,slash)||substr(new.1,1,j)
        Else              'CMSG' ori||substr(new.1,1,j)
```

```
    Else Leave
  End

End

return
```