2007 Rexx Language Association Symposium

Walk With a Dinosaur:
The Journey From VM/CMS To A PC

Les Koehler
1 May 2007

Table of Contents

Abstract

The beginning

- **RPQ Manufacturing**
- **System 360**
- **Diagnostics**
- **IBM Voluntary Education**
- **RAS Engineering**
- **Other RAS Projects**

VM/CMS

- **File System**
- **Tools**

Transition to the PC

Existing Tools

- **The Hessling Editor (THE)**
- **KEDIT**

Developed Tools

- **EXTRACT**
- **DIRS**
- **SHOWPF**
- **SHOWALPH**
- **SHOWOTHER**
- **TOSUB**
- **OOFIX**
- **FIXDO**
- **PUSH/POP**
- **RUN**

Support And Encouragement

Mindset Change

Source Code

Abstract

I'll present some of my experience and background at IBM before I became the first person outside of the U.K. to have a running copy of Rexx.

I'll then give an overview of the mainframe VM/CMS environment and its facilities and tools that I worked with before I retired.

With that as background, I'll then show how I made the transition to the PC, including:

- What tools I found
- The support and encouragement of RexxLA members that I received
- The 'mindset change' that helped the transition
- The tools I developed to ease the transition

The beginning

RPQ Manufacturing

Assembly, test, repair. Specialized in 2701 RPQ's, including:

Bi-sync adapter
Low-cost display adapter
Two-processor switch
Autodin adapter

The CE Box was an integral part of the tester's toolkit. It allowed the tester to completely simulate the cpu side of the channel. The two rows of switches toggled all the Bus and Tag lines. The lights were wired to significant logic points on the adapter (or channel interface) board.

By using the CE Box to run preliminary tests, the tester avoided using precious cpu time needlessly.

System 360

System 360 established a new path for future processors from IBM. Prior to this, each new generation of IBM computers was incompatible with the previous line.

An application program written for S/360 that didn't have any privileged instructions can still be run today on any of the follow-on systems (370, 390, ESA, z).

We used a S/360 Model 30 for final testing. Not pictured are the Channel Extender boxes that allowed equipment to be tested to be attached at various points around the raised floor without impacting the test underway.

Diagnostics

All the diagnostics were written in 360 Assembler and were shipped in card boxes, along with the diagnostic supervisor.

Program listings were shipped in logic binders, about 18x27 inches.

A lot of our diagnostic programs were brand new, since we made unique equipment. One of our Quality Control inspectors was assigned to do initial debugging if a diagnostic had problems. This saved the RAS (Reliability and Servicability) programmer in Research Triangle Park from having to drive to our location in Raleigh.

It was this exposure that first got me interested in programming.

IBM Voluntary Education

1. **After hours**
2. **On premises**
3. **Experienced IBMers as instructors**
4. **All materials provided by IBM**
5. **Certificate added to education records**
6. **Free!**

I took a course in 360 Assembler and after I transferred to Quality Control I assumed responsibility for diagnostic debugging.

This led to frequent interaction with the RAS programmers at RTP, culminating in an offer of a temporary 90 day assignment to help them convert some diagnostics to a new diagnostic supervisor and to see if I really liked programming.

RAS Engineering

Assigned to convert three diagnostics: all of them for RPQ's that I used to build!

The first one, Bi-Sync Adapter, needed a complete re-write because it didn't give the tester enough information to quickly determine where the hardware was failing.

I discovered some research done at Yorktown Heights that used a lookup table to calculate the CRC, instead of simulating the CRC logic with register shifting and exlusive ORing.

From the testers point of view, this was a vast improvement! Upon CRC failure, the diagnostics would re-run the test and show all the intermdediate CRC values that should appear in the CE Box lights. The tester could then use the CE Box to emulate the exact sequence that the diagnostic used and quickly find the failure.

Little did I know that the lesson learned on this assignment would carry through for the rest of my career as a programmer. It's a LOT DIFFERENT when you can have the same viewpoint as the end-user!

Other RAS Projects

Brokerage Communication System - As far as I know, the last custom designed processor that IBM developed.
Bring-up diagnostics: cpu and memory
NYSE Stock market simulator
Gas panel - UC0 based. I wrote an EXEC interpreter so the CE could easily write a custom diagnostic to pinpoint a problem.

VM/CMS

The NYSE Stock Market Simulator was destined to run on the customer's system, but we didn't have an OS/MVT system to play with. The solution was to use the only VM/370 system in RTP, which belonged to Quality Control.

Using VM, you can run any operating system 'virtually', so we put our programs on a 2311 disk pack and simply IPLed it from VM!

File System

- **Mini disk mode letter (A-Z) maps to an allocation on real dasd**
- **Filename Filetype Filemode: PROFILE EXEC A**
- **Fileid automatically uppercased. Lower case requires special handling**
- **Record format either Fixed length or Variable length**
- **Flag for executables: EXEC, MODULE**
- **Directory of files for a mini-disk is just another file**
- **Directory Origin Pointer: record 4/5**
- **Current Directory pointed to by record 3 (?)**
- **Atomic commit**

CMS also has a Shared File System (SFS) and a Byte File System (BFS) that uses SFS.

Tools

- **FILELIST**

- **RDRLIST**

- **FULIST**

- **FULRDR**

- **XEDIT**
- **BROWSE**
- **BookMaster**
- **CODEPRTX**
- **NOTE**
- **SCANFILE**
- **FCOPY**
- **PKGEDIT**
- **Rexx Compiler**

Transition to the PC

Existing Tools

The Hessling Editor (THE)

Based on XEDIT and KEDIT, THE is a command line driven full screen text editor that provides key bindings for many operations. It takes full advantage of the pc keyboard and display, thus it is not limited to the 3270 'green screen' architecture. For those adept at mousing, mouse operations are included.

After all my years using XEDIT, and it's predecessor EDIT, as well as EDGAR, WED and RED (internal editors that preceeded XEDIT), discovering THE was like a duck finding water.

THE includes syntax highlighting for many languages and a multitude of Rexx macros to make life easier, including:

- **DIFF - Find the next difference between two files in the ring**
- **TOTAL - Sum numbers in a marked block**
- **NL - Menu of named lines**
- **MATCH - Mark a block with matching keywords**
- **COMM - Add comments to a block of lines**
- **APPEND - Append a string to a target**
- **UNCOMM - Remove comments from a block of lines**
- **L - Prefix macro to lowercase a (block of) lines**

KEDIT

After I started using THE, I discovered that I had a copy of KEDIT from a previous Symposium. That gave me some additional documentation, since THE can also emulate much of KEDIT. Also, I found some macros of interest:

- **RINGLIST - List all the files in the ring in a popup for selection.**

  The following enhancements were made:
- **Added NONE to RingListOrder**
- **Added abbrev? switch to abbreviate long path names if needed to reduce horizontal scrolling.**
- **Walk around the ring to get to DIR.DIR to avoid rebuilding.**
- **Added ALT= to popup to make unsaved changes stand out.**
- **Added flag to expand CANCEL line**
- **Bound to ALT-R**

RINGLIST Source

- **MORE - Add more lines displayed by ALL**
- **LESS - Display fewer lines displayed by ALL**
- **KEXXREXX - Convert Kedit comment to Rexx and back**
- **ALTERED - Display the lines that have been changed**

Developed Tools

These tools are all THE macros, written in Rexx:

- **XTRACT**

  Displays internal settings. When writing a macro, the EXTRACT command is used to retrieve information about the edit session and put it in variables.
  XTRACT Source

  DIRS
- **Display a menu of directories. Favorites are always shown and directories can be excluded. The current ring of files is examined for a DIR.DIR file and all the directories in it are included.**

  A file can be selected by positioning the cursor and pressing ALT-X.

  DIRS Source
- **SHOWPF**

  Displays all PF key settings
  SHOWPF Source
- **SHOWALPH**

  Displays the alpha key bindings
  SHOWALPH Source
- **SHOWOTHER**

  Display the other key bindings
  SHOWOTHER Source

TOSUB

- **Position to the CALLed subroutine, or return to the original line**

TOSUB Source

- **OOFIX**

Fix Classic Rexx (CMS/TSO) disallowed characters for OORexx

OOFIX Source

- **FIXDO**

Fix standalone DO to join it with the previous line

FIXDO Source

- **PUSH/POP**

Save and restore a specified EXTRACT setting

PUSH/POP Source

- **RUN**

Execute the file being edited

RUN Source

Support And Encouragement

I'm deeply indebted to Mark Hessling for writing the RUN macro for me and his tireless answering of all my questions as I got up to speed on using THE. Without THE, I would probably never started this journey.

Lee Peedin graciously wrote the program to solve the first requirement that got me started on this journey and provided continuous support and encouragement in improving my understanding of ooRexx.

Mindset Change

Using THE and ooRexx has lowered the enormous wall that I thought existing between me and using the pc for anything other than canned applications.

Source Code

RINGLIST Source

```
/*
* RINGLIST.THE
*
* Display a popup menu containing a sorted list of all files currently in
* the ring.  Select the file you want to get the focus.
*
* See RINGLIST.TXT for more detailed documentation.
*
* Works with:
*
*    KEDIT for Windows 1.5
*
* Original author: Kent Downs, MSG, 11/97
* Modified for THE by Les Koehler
*  Enhancements:
*   - Added NONE to RingListOrder
*   - Added abbrev? switch to abbreviate long path names if needed
*     to reduce horizontal scrolling.
*   - Walk around the ring to get to DIR.DIR to avoid rebuilding.
*   - Add ALT count to popup
*/
/*
if \(version.1() = "KEDIT/WINDOWS" & version.2() \= "1.00") then do
   prompt = "/This macro requires KEDIT for Windows 1.5/"
   "dialog" prompt "title /Error/ ok"
   exit 99
   end
*/
/* Any files in the ring? */
if ring.0()<2 then do
  'msg No files in the ring to select.'
  exit
end

/* Retrieve the list formatting choice (set in the profile)*/
"EDITV GET RINGLISTORDER"
--trace or
RingListOrder = strip(translate(RingListOrder))
valid=''
valid=valid 'NAME_FIRST' /* Sort by name. File name is to the left of the file path, e.g.,*/
                         /*   hello.txt (in) d:\work\stuff*/
valid=valid 'NAME' /* Sort by name. File is listed as a complete fileid, e.g.,*/
                   /*- d:\work\stuff\hello.txt*/
```

```
valid=valid 'FILEID' /* Sort by fileid. File is listed as a complete fileid*/
                     /*- d:\work\stuff\hello.txt*/
valid=valid 'NONE' /* No sort */
If wordpos(ringlistorder,valid)=0 then ringlistorder=word(valid,1)
--trace o
/* Setup long strings to abbreviate when needed */
abbrev?=1     /* 1-Abbreviate, 0=Don't abbreviate */
/* Long.1 is calculated at DISPLAY_LIST: */
abbrev.1=' '
long.2='\Documents and Settings\'
abbrev.2='\Docs...\'
long.3='\Les\My Documents\'
abbrev.3='\My...\'
long.0=3
canmsg?=1      /* Show RINGLISTORDER and ABBREV with CANCEL? */
call BuildList
call DisplayList
exit 0

/* Construct the sorted list as a compound variable (array). Create a temporary*/
/* file to work in and do our sorting.*/
BuildList:
   SortedList. = ""
   /* Remember the current file so we can return to it after working with*/
   /* a temporary file.*/
   ThisFile = efileid.1()

   /* Preserve the cursor position within the file*/
   call SavePosition

   "extract /ring/"
   /* Open the temporary file*/
   "nomsg the zzzzzzzz.zzz"
   'top'
   'nomsg del *'
   "set autosave off"

   /* Track the longest file name and file extension and path*/
   wFName = 0
   wFExt = 0
   wPath=0
   walt=0

   firstid=1                          /*For COMPAT KEDIT*/
   if ring.0-1=ring.1 then firstid=2 /*For COMPAT XEDIT*/
   do i = firstid to ring.0
      Fileid = ring.i
      alt=word(fileid,words(fileid))
```

```
   n = pos("Size=", Fileid)
   Fileid = strip(substr(Fileid, 1, n - 1))

   /* Get the path*/
   n = lastpos("\", Fileid)
   NextFPath = substr(Fileid, 1, n - 1)
   /* If it's in the root dir there won't be a trailing backslash*/
   if substr(NextFPath, length(NextFPath)) = ":" ,
      then NextFPath = NextFPath"\"

   /* Extract the name and extension*/
   Fileid = substr(Fileid, n + 1)
   n = lastpos(".", Fileid)
   if n = 0 then do
      NextFExt = ""
      NextFName = Fileid
   end
   else do
      NextFExt = substr(Fileid, n)
      NextFName = substr(Fileid, 1, n - 1)
   end

   wFName = max(wFName, length(NextFName))
   wFExt = max(wFExt, length(NextFExt))
   wPath=max(wpath,length(nextfpath))
   walt=max(walt,length(alt))

   /* Replace the line with the parts separated by tab chars*/
   "input" NextFName || d2c(9) || NextFExt || d2c(9) || NextFPath,
    || d2c(9) alt
end

/* Line up the columns and sort*/
"set tabs" (wFName + 2) (wFName + wFExt + 4),
  (wFName + wFExt +wpath + 6)
"expand all"

if ringlistorder\='NONE' then do
  if RingListOrder = "NAME_FIRST" | RingListOrder = "NAME",
   then "nomsg sort all 1" wFName+2
   else "nomsg sort all" (wFName + wFExt + 4) ,
    wFName+wFName + 2+wpath ,
    "1" (wFName) (wFName + 2) (wFName + wFExt + 3)
end

/* Collect the sorted pieces into an array*/
do i = 1 to size.1()
   ":"i
```

```
      line = curline.3()
      alt.i=word(line,words(line))
      line=delword(line,words(line))
      SortedList.i = strip(substr(line, 1, wFName))
      SortedList.i =  SortedList.i ,
      || strip(substr(line, wFName + 2, wFExt))
      path = strip(substr(line, wFName + wFExt + 4,wpath))
      parse var path drive+1 rest
      path=translate(drive)||rest
      if RingListOrder \= "NAME_FIRST" then do
         if substr(path, length(path)) \= "\" then path = path"\"
         SortedList.i =  path || SortedList.i
         copylist.i=sortedlist.i /* Keep for later */
      end
      else do
         copylist.i=path /* Keep a complete copy for later */
         if substr(path, length(path)) \= "\" then ,
          copylist.i=copylist.i"\"
         copylist.i=copylist.i||sortedlist.i
         SortedList.i =  left(SortedList.i,wfname+wfext+0),
          '(in)' left(path,wpath+1)
      end
      sortedlist.i=overlay(alt.i,sortedlist.i,wfname+wfext+wpath,
       +8,walt)
   end

   /* Exit the temp file and return to the original*/
   "qquit"
--   'nomsg edit' ThisFile
/* 'redraw'*/
return

/* Given an array of sorted fileids, construct and display a popup menu*/
DisplayList:
  'extract /lscreen/'
  trace or
  maxwidth=lscreen.2-4
  longest=length(sortedlist.1)

 --     call RestorePosition
   --   exit
  abbreved?=0
  if longest>maxwidth & abbrev? then do /* Abbreviate the path so we don't */
    trim=0+(longest-maxwidth) /* have to use the cursor keys */
    long.1=copies(' ',trim+1) /* Look for blanks */
    do i=1 to ring.0
      do l=1 to long.0
        hit=pos(long.l,sortedlist.i)
```

```
      if hit>0 then do
        abbreved?=1
        sortedlist.i=delstr(sortedlist.i,hit,length(long.l))
        sortedlist.i=insert(abbrev.l,sortedlist.i,hit-1)
        ln=length(sortedlist.i)
        if ln<=maxwidth then do /* We've taken out enough */
          if ln<maxwidth then do /* To much? */
            pad=maxwidth-ln      /* Pad it out to width */
            hit=pos('Alt=',sortedlist.i)
            sortedlist.i=insert(copies(' ',pad),sortedlist.i,hit-1)
          end
          leave l
        end
      end
    end
  end
end

 menu="/"
 /* Build the POPUP command*/
 do i = 1 to ring.0
    menu = menu || sortedlist.i"/"
 end
canmsg=''
--canmsg?=0
if canmsg? then do
  canmsg=' (RingListOrder='ringlistorder
  if abbreved? then canmsg=canmsg 'Abbrev)'
  else canmsg=canmsg')'
  ln=length(canmsg)
  if \abbreved? then do
    if length(canmsg)<longest then
     canmsg=right(canmsg,longest-6)
  end
  else canmsg=right(canmsg,maxwidth-6)
end

 menu ='/Cancel'canmsg'/-'|| delstr(menu,length(menu)-1)
 /* Display the menu*/
 'redraw'
 "popup" menu

 if popup.1 = "" | word(translate(popup.1),1) = "CANCEL" then do
    call RestorePosition
    exit
 end
 item=popup.2-2 /* Allow for CANCEL and horizontal line */
 popup.1=copylist.item /* Get full path and fileid that we saved */
```

```
   /*
   if RingListOrder = "NAME_FIRST" then do
      /* Need to remove the extra stuff used to get the paths to line up*/
      parse var popup.1 fid '(in)' path 'Alt=' .
      fid=strip(fid)
      path=strip(path)
      if substr(reverse(path),1,1)\='\' then path=path'\'
      popup.1=path||fid
   end
   else popup.1=subword(popup.1,1,words(popup.1)-1)
   */
   if efileid.1() = popup.1 then do
      call RestorePosition
      exit
   end

   /* Go to the selected file*/
   if pos('DIR.DIR',popup.1)>0 then do
      do r=1 to ring.0 while \dir()
        'nextwindow'
      end
    end
   else 'the' popup.1
   exit
return

/* Preserve the current line and cursor positions*/
SavePosition:

   "preserve"
   SavedLineNumber = line.1()
   "EDITV PUT SAVEDLINENUMBER"
return

/* Restore the original current line and cursor positions*/
RestorePosition:
   "EDITV GET SAVEDLINENUMBER"
   ":"SavedLineNumber
   "restore"
return
```

EXTRACT Source

```
/* Extract the setting and display it */
signal on syntax
arg setting name
'preserve'
```

```
'set linend off'
'extract /'setting name'/'
if rc=0 then do
  'msg' setting'.0='value(setting'.0')
  do s=1 to value(setting'.0')
    'msg' setting'.'s'='value(setting'.s')
  end
end
else do
  'msg Unknown keyword:' setting
end
'restore'
exit
syntax:
  'msg Unknown keyword:' setting
  'restore'
```

DIRS Source

```
/* Display a list of DIRs to pick from */
dir.1='C:\'
dir.2='C:\LesK2007'
dir.3='C:\MyTHEstuff'
dir.4='C:\MyRexxStuff'
dir.5='C:\THE'
dir.6='C:\THE\extras'
dir.7='C:\Documents and Settings\Les'
dir.8='C:\Documents and Settings\Les\My Documents\HOA\Financials'
dir.0=8
known?.=0
Do k=1 To dir.0
  ix=Translate(dir.k)
  known?.ix=1
End
ignore?.=0
ignore.1='c:\System Volume Information'
ignore.2='c:\WINNT'
ignore.3='c:\RECYCLER'
ignore.4='c:\MyWorks'
ignore.5='c:\BJPrinter'
ignore.6='c:\cabs'
ignore.0=6
--trace r
Do i=1 To ignore.0
  ix=Translate(ignore.i)
  ignore?.ix=1
```

```
End
--Trace o
Call buildlist
--Trace o
Call displaylist
Exit 0

/* Construct the sorted list as a compound variable (array). Create a temporary */
/* file to work in and do our sorting. */
BUILDLIST:
  /* Remember the current file so we can return to it */
thisfile = efileid.1()

  /* Preserve the cursor position within the file */
Call saveposition
'extract /ring/'
found?=0
Do r=1 To ring.0 While \found? /* Is there a DIR in the ring? */
  Parse Var ring.r fid ' Size=' .
  found?= fid='C:\DIR.DIR'
End
If found? Then Do /* Yes, go to it */
  Do r=1 To ring.0 While \dir()
    'nextwindow'
  End
  If dir() Then Do /* Double-check that it's there */
    do d=dir.0 to 1 by -1 /* Make room for DIR.DIR */
      dd=d+1
      dir.dd=dir.d
    end
    dir.1='C:\DIR.DIR'    /* Insert it */
    dir.0=dir.0+1
    'extract /zone/stay/line/' /* Save our settings. Can't use PRESERVE */
    'top'
    'set stay on'
    'zone 1 *'
    'nomsg all /(dir)/'
    dirs?=rc=0
    If dirs? Then Do
      'zone 39 40'
      'nomsg macro less /./'
      'top'
      'extract /nbscope/'
--      eof?=0
--      Do s=1 To nbscope.1 Until eof?
      Do s=1 To nbscope.1
        'next'
   --     eof?=rc=1
```

```
   --     If \eof? Then Do
          'extract /dirfileid/'
          If Left(dirfileid.2,1)\='.' Then Do
            dirfileid=dirfileid.1||dirfileid.2
            k=Translate(dirfileid)
            If \known?.k & \ignore?.k Then Do     /*Add to dir. array */
              Parse Value dir.0+1 dirfileid ,
               With z dir.z 1 dir.0 .
              known?.k=1
            End
          End
    --    End
      End
    End
    'set stay' stay.1        /* Restore our settings */
    'set zone' zone.1 zone.2
    'all'
    ':'line.1
    'edit' thisfile          /* Go back to where we were */
  End
End
Return

/* Given an array of sorted fileids, construct and display a popup menu */
DISPLAYLIST:
  menu="/"
     /* Build the POPUP command */
  Do i = 1 To dir.0
    menu = menu || dir.i"/"
  End

  menu ='/Cancel/-'|| Delstr(menu,Length(menu))
  'redraw'
  "popup" menu    /* Display the menu */
  select
    when popup.1 = "" | Translate(popup.1) = "CANCEL" ,
     | efileid.1() = popup.1 Then Do
      Call restoreposition
    End
       /* Go to the selected file */
    when found? & popup.1='C:\DIR.DIR' then do /* Switch to current DIR */
      Do r=1 To ring.0 While \dir()
        'nextwindow'
      End
    end
    otherwise 'dir' popup.1             /* Go to new DIR */
  end
Return
```

```
/* Preserve the current line and cursor positions */
SAVEPOSITION:
  savedlinenumber = line.1()
  "EDITV PUT SAVEDLINENUMBER"
Return

/* Restore the original current line and cursor positions */
RESTOREPOSITION:
  "EDITV GET SAVEDLINENUMBER"
  ":"savedlinenumber
Return
```

SHOWPF Source

```
/* Display the F keys in a popup window
   Author: Les Koehler
   Date: 8Dec06
   Audience: New THE users, especially those with
             a VM Xedit background
   Freeware!
*/
pre=''
pres='C- A- S- .'       --Prefixes for a key
ln.=0
used=''
alphanum='ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890'
blank=copies(' ',256)

do p=1 to words(pres)
  pre=word(pres,p)
  if pre='.' then pre=''     --Get rid of place holder
  do k=1 to 12
    cmd=''
    'extract /showkey' pre || 'F'k'/'
    if showkey.0\=0 then do  --Accumulate the command
      do c=1 to showkey.0
        cmd=cmd showkey.1
      end
      ln.pre=max(ln.pre,length(cmd)) --Save biggest length
    end
    key.pre.k=cmd
    if cmd\='' then do               --Save special chars
      test=space(translate(translate(cmd),blank,alphanum),0)
      if test\='' then do
        do t=1 to length(test)
          char=substr(test,t,1)
```

```
          if pos(char,used)=0 then used=used||char
        end
      end
    end
  end
  key.pre.0=12
end

delims='/\+!@#$%^&*()_-+{}|[]:";<>?,.~`'
do u=1 to length(used) --Remove special chars used
  parse var used char+1 used
  delims=translate(delims,blank,char)
end
delim=left(space(delims,0),1) --Choose delimiter for POPUP
line=delim
pres='. C- A- S-'
save=''
do p=1 to words(pres)
  pre=word(pres,p)
  if pre='.' then pre=''
  if ln.pre>0 then do   --Allow for length of col header
    ln.pre=max(ln.pre,length(pre))
    save=save pre       --Accum the ones actually used
  end
end
pres='.' save
wds=words(pres)
do p=1 to wds            --Construct the menu title
  pre=word(pres,p)
  if pre='.' then do
    pre=''
    title=' f'copies(' ',ln.pre+1)'|'
  end
  else do
    title=title || center(pre'f',ln.pre)
    if p<wds then title=title'|'
  end
end
line=delim||title||delim
do l=1 to 12                    --Construct rest of menu
  line=line || right(l':',3)
  do p=1 to words(pres)
    pre=word(pres,p)
    if pre='.' then pre=''
    line=line || left(key.pre.l,ln.pre)
    if p<wds then line=line'|'
  end
  line=line||delim
```

```
end
'popup 'line'-'delim 'Press ENTER to continue' used delim
```

SHOWALPH Source

```
/* -- Display the Alpha keys in a popup window --
   Author: Les Koehler
   Date: 8Dec06
   Audience: New THE users, especially those with
             a VM Xedit background
   Freeware!
*/
pre=''
pres='A- C- .'       /* Prefixes for a key */
ln.=0
used=''
alphanum='ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890'
blank=copies(' ',256)
'extract /lscreen/'
width=lscreen.2-8

do p=1 to words(pres)
  pre=word(pres,p)
  if pre='.' then pre=''     /* Get rid of place holder */
  do k=1 to 26
    cmd=''
    'extract /showkey' pre || substr(alphanum,k,1)'/'
    if showkey.0\=0 then do  /* Accumulate the command */
      do c=1 to showkey.0
        cmd=cmd showkey.c
--        if showkey.0>1 then cmd=cmd||'#'
      end
      lncmd=length(cmd)
      ln.pre=max(ln.pre,lncmd) /* Save biggest pre length */
      if lncmd>ln.pre.big then do /* Save biggest key length */
        ln.pre.big=lncmd
        ln.pre.ix=k
      end
    end
    key.pre.k=cmd
    key.pre.ln.k=length(cmd)
    if cmd\='' then do               /* Save special chars */
      test=space(translate(translate(cmd),blank,alphanum),0)
      if test\='' then do
        do t=1 to length(test)
          char=substr(test,t,1)
          if pos(char,used)=0 then used=used||char
```

```
        end
      end
    end
  end
  key.pre.0=26
end

delims='/\+!@#$%^&*()_-+{}|[]:";<>?,.~`'
do u=1 to length(used) /* Remove special chars used */
  parse var used char+1 used
  delims=translate(delims,blank,char)
end
delim=left(space(delims,0),1) /* Choose delimiter for POPUP */
line=delim
pres='. A- C-'
save=''
tot=0
biggest=0
ptr=0

do p=2 to words(pres)         /* Check that everything will fit */
  pre=word(pres,p)
  tot=tot+ln.pre
  if ln.pre>biggest then do /* Save widest prefix column */
    biggest=ln.pre
    ptr=p
  end
end
size=0
If tot>width then do /* We have a line that is to long */
  pre=word(pres,ptr) /* Get the section it is in */
  if ptr=2 then other=word(pres,3)
  else other=word(pres,2)
  length=0
  do k=1 to 26
    if k\=ln.pre.ix then do /* Find the Next-longest max key */
      size=max(size,key.pre.ln.k)
    end
  end
  leftover=width-ln.other /* How much room is left? */
  if leftover>size then size=leftover-2  /* allow some some room for truncating */
  ln.pre=size                       /* Fix the width */
  k=ln.pre.ix                       /* Get index to long key descrip */
  key.pre.k=left(key.pre.k,size-3)'...' /* Truncate it */
end
do p=1 to words(pres)
  pre=word(pres,p)
  if pre='.' then pre=''
```

```
  if ln.pre>0 then do   /*Allow for length of col header */
    ln.pre=max(ln.pre,length(pre))
    save=save pre       /*Accum the ones actually used */
  end
end

pres='.' save
wds=words(pres)
do p=1 to wds            /*Construct the menu title */
  pre=word(pres,p)
  if pre='.' then do
    pre=''
    title='ltr'copies(' ',ln.pre+0)'|'
  end
  else do
    title=title || center(pre'ltr',ln.pre)
    if p<wds then title=title'|'
  end
end
line=delim||title||delim
do l=1 to 26                    /*Construct rest of menu */
  line=line || right(substr(alphanum,l,1)':',3)
  do p=1 to words(pres)
    pre=word(pres,p)
    if pre='.' then pre=''
    line=line || left(key.pre.l,ln.pre)
    if p<wds then line=line'|'
  end
  line=line||delim
end
'popup 'line'-'delim 'Press ENTER to continue' used delim
```

SHOWOTHER Source

```
/* -- Display the 'Other' keys in a popup window --
   Author: Les Koehler
   Date: 8Dec06
   Audience: New THE users, especially those with
             a VM Xedit background
   Freeware!
*/
pre=''
ln.=0
used=''
alphanum='ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890'
keys=     ,
'BKSP    ',
```

```
'CURD    ',
'CURL    ',
'CURR    ',
'CURU    ',
'DEL     ',
'END     ',
'ENTER   ',
'ESC     ',
'HOME    ',
'INS     ',
'NUMENTER',
'PGDN    ',
'PGUP    ',
'PLUS    ',
'SLASH   ',
'STAR    ',
'MINUS   ',
'NUM7    ',
'NUM8    ',
'NUM9    ',
'NUM4    ',
'NUM5    ',
'NUM6    ',
'NUM1    ',
'NUM2    ',
'NUM3    ',
'NUM0    ',
'NUMSTOP ',
'TAB     '
pres='A- C- S- .'       /* Prefixes for a key */
blank=copies(' ',256)
'extract /lscreen/'
width=lscreen.2-15

do p=1 to words(pres)
  pre=word(pres,p)
  if pre='.' then pre=''     /* Get rid of place holder */
  do k=1 to words(keys)
    cmd=''
    'extract /showkey' pre || word(keys,k)'/'
    if showkey.0\=0 then do  /* Accumulate the command */
      do c=1 to showkey.0
        cmd=cmd strip(showkey.c)
--        if showkey.0>1 then cmd=cmd||'#'
      end
      lncmd=length(cmd)
      ln.pre=max(ln.pre,lncmd) /* Save biggest pre length */
      if lncmd>ln.pre.big then do /* Save biggest key length */
```

```
        ln.pre.big=lncmd
        ln.pre.ix=k
      end
    end
    key.pre.k=cmd
    key.pre.ln.k=length(cmd)
    if cmd\='' then do               /* Save special chars */
      test=space(translate(translate(cmd),blank,alphanum),0)
      if test\='' then do
        do t=1 to length(test)
          char=substr(test,t,1)
          if pos(char,used)=0 then used=used||char
        end
      end
    end
  end
  key.pre.0=26
end

delims='/\+!@#$%^&*()_-+{}|[]:";<>?,.~`'
do u=1 to length(used) /* Remove special chars used */
  parse var used char+1 used
  delims=translate(delims,blank,char)
end
delim=left(space(delims,0),1) /* Choose delimiter for POPUP */
line=delim
pres='. A- C- S-'
save=''
tot=0
biggest=0
ptr=0

trace or
do p=1 to words(pres)         /* Check that everything will fit */
  pre=word(pres,p)
  if pre='.' then pre=''     /* Get rid of place holder */
  tot=tot+ln.pre
  if ln.pre>biggest then do /* Save widest prefix column */
    biggest=ln.pre
    ptr=p
  end
end
size=0
If tot>width then do /* We have a line that is to long */
  pre=word(pres,ptr) /* Get the section it is in */
  if pre='.' then pre=''     /* Get rid of place holder */
  tot_others=0
  do p=1 to words(pres) /* Get next biggest */
```

```
    if p\=ptr then do
      npre=word(pres,p)
      if npre='.' then pre=''     /* Get rid of place holder */
      tot_others=tot_others+ln.npre
    end
  end
  length=0
  do k=1 to words(keys)
    if k\=ln.pre.ix then do /* Find the Next-longest max key */
      size=max(size,key.pre.ln.k)
    end
  end
  leftover=width-tot_others /* How much room is left? */
  if leftover>size then size=leftover-2  /* allow some some room for truncating */
  ln.pre=size                        /* Fix the width */
  k=ln.pre.ix                        /* Get index to long key descrip */
  key.pre.k=left(key.pre.k,size-3)'...' /* Truncate it */
end
do p=1 to words(pres)
  pre=word(pres,p)
  if pre='.' then pre=''
  if ln.pre>0 then do   /*Allow for length of col header */
    ln.pre=max(ln.pre,length(pre))
    save=save pre       /*Accum the ones actually used */
  end
end

pres='.' save
wds=words(pres)
do p=1 to wds             /*Construct the menu title */
  pre=word(pres,p)
  if pre='.' then do
    pre=''
    title='     key'copies(' ',ln.pre+2)'|'
  end
  else do
    title=title || center(pre'key',ln.pre)
    if p<wds then title=title'|'
  end
end
line=delim||title||delim
do l=1 to words(keys)          /*Construct rest of menu */
  defined?=0
  do p=1 to words(keys) until defined?
    pre=word(pres,p)
    if pre='.' then pre=''
    defined?=key.pre.l\=''
  end
```

```
  if defined? then do
    line=line || right(word(keys,l)':',10)
    do p=1 to words(pres)
      pre=word(pres,p)
      if pre='.' then pre=''
      line=line || left(key.pre.l,ln.pre)
      if p<wds then line=line'|'
    end
    line=line||delim
  end
end
used='(Undefined keys not shown)'
'popup 'line'-'delim 'Press ENTER to continue' used delim
```

TOSUB Source

```
/* Position to the subroutine the cursor is on, or return to orig pos
*/
trace or
parse upper arg ret
return?=ret='RETURN'
emsg=''
'preserve'
'set case = ignore = ='
'extract /line/cursor/field/efileid/curline/'
syn='@tosub'efileid.1
lnsyn=length(syn)
'extract /synonym' syn'/'
parse var synonym.3 . on nest .
if on\='ON' then do /* First time */
  if return? then do
    emsg='Nothing saved for' syn 'to RETURN to.'
  end
  else do
    'set synonym' syn lnsyn 'nop ON 0' /* Set nesting for this file */
    nest=0
  end
end
else do
  if \return? then do
    'set synonym' syn||nest lnsyn+length(nest) 'nop',
     line.1 cursor.3 cursor.4 /* Save where we are */
  end
end
if \return? then do
  ucline=translate(field.1)
  parse value ucline with 'CALL' subrtn .
```

```
  if subrtn='' then do
    emsg="Sorry, but TOSUB can't figure out what the cursor is on.",
     "Only CALL is supported at this time."
  end
  else do
    'locate /'subrtn':/'
    if rc=0 then do
      nest=nest+1              /* Point to next save area */
      'set synonym' syn lnsyn 'nop ON' nest /* Set nesting for this file */
      'cursor escreen' curline.2 length(subrtn)+2
    end
    else emsg=subrtn': not found'
  end
end
else call goback
'restore'
if emsg\='' then 'emsg' emsg
exit

goback:
  nest=nest-1
  'extract /synonym' syn||nest'/'
  parse var synonym.3 . line.1 cursor.3 cursor.4
  if datatype(line.1,'W') then do
    ':'line.1 'cursor file' cursor.3 cursor.4
    'set synonym' syn lnsyn 'nop ON' nest /* Set nesting for this file */
    'set synonym' syn||nest lnsyn+length(nest) 'nop' /* Clear settings */
  end
  else emsg='Nothing saved for' syn||nest
return
```

OOFIX Source

```
/* Fix a file for disallowed ooRexx chars and functions
   Specifically: ^$#@ and cent-sign
   Author: Les Koehler
*/
'preserve'
'set stay on'
bad_chars='^ $ # @'
good_chars='\ !dlr! !lb! !at!'
do i=1 to words(bad_chars)
  'top'
  old=word(bad_chars,i)
  new=word(good_chars,i)
  'locate /'old'/'
```

```
  if rc=0 then 'schange /'old'/'new'/ * *'
end
'restore'
'MACRO FIXFIND *'
return
```

FIXDO Source

```
/* Fix standalone DO and THEN in Rexx style code
 * Author: Les Koehler
 * Syntax: FIXDO <n or *>
 *
 * If no parameters are passed, FIXDO will leave you positioned
 * at the last potential line to be fixed.
 *
 * If it finds a comment on the two lines to be joined, it stops
 * to let you decide how to do the join.
 *
 * It is suggested that you modify the whole file to insert two leading
 * blanks before running FIXDO.  You can then readily remove them.
 *
 */
 trace or
parse source . . exec_name exec_type .
parse value reverse(exec_name) with '.' me '\' .
me=reverse(me)
  Signal On NOVALUE
  Signal On SYNTAX
  'set impos off'
  c='COMMAND'
  m='MACRO'
  eof=0
  soc='/' || '*'
  eoc='*' || '/'

  'COMMAND EXTRACT /LINE /TOF /EOF /SIZE'
  Select
    When tof.1 = 'ON'
      Then start_line = line.1 + 1
    When eof.1 = 'ON'
      Then start_line = line.1 - 1
    Otherwise start_line = line.1              /* Line we started on */
  End
  first_line = start_line                 /* Line to begin processing */
  c 'PRESERVE'
  c 'SET STAY OFF'
--c 'SET SPAN OFF'
```

```
--c 'SET VARBLANK OFF'
  c 'SET WRAP     OFF'
  arg target .
  If target='*' Then lines=size.1
  If target='' Then lines=size.1
  If datatype(target,'Whole') Then lines=target
trace value 'OR'
  c 'EXTRACT /SYNONYM @FIXDO@/'
  Parse var synonym.3 testline
  if line.1\=testline then c 'UP'
trace value 'OR'
  Do l=1 to lines until eof
    call nexthit
    If \eof Then Do
      c 'CMSG' me target
--      c 'REFRESH'
      If target='' & hit\=0 Then eof=1
      Else Do
        c '+1'
        eof=(rc=1)
        c '-1'
      End
    End
  End
  c 'SET SYNONYM @FIXDO@ 6 l :'line.1
  c 'RESTORE'
  Exit

/*Ð××××××××××××¯
  ³   NEXTHIT   ³
  ¿××××××××××××¨*/
 parse source nexthit! .;drop nexthit!; /* dummy for SRCXREF */
NEXTHIT:
  Trace r
  sc=0
  cmt2=0
  hit=0
  c 'nomsg /  THEN  / | /  DO  / | /  THEN DO  / | / THEN  /' ,
   '|/  ELSE  / | / ELSE  /'
  If rc=0 Then Do
    c 'EXTRACT /CURLINE/'
    curline.3=strip(curline.3,'T')
    testline=curline.3
    test=Strip(curline.3,'L')
    Parse Upper Var test w1 w2 rest
    up=0
    then?=0
    thenif?=0
```

```
If w1='THEN' | w1='DO'/* | w1='ELSE' | w1 w2='ELSE DO'*/ ,
 | w1 w2='THEN DO' Then up=1
If rest='' | w2='' | w2=soc        ,
 | Left(Strip(rest,'L'),2)=soc   ,
 | Pos('  THEN  ',rest'  ')\=0 Then Do
  cmt2=Pos(soc,curline.3)\=0 | Pos(eoc,curline.3)\=0
  then?=wordPos('THEN',translate(curline.3))\=0
  c '-'up
  c 'EXTRACT /CURLINE/LINE/'
  curline.3=strip(curline.3,'T')
  sc=Pos(soc,curline.3)
  comment=''
  If then? Then Do
    Parse Upper Var curline.3 wif
    thenif?=(wif='IF')
  End
  If sc>0 Then Do
    If \cmt2 & target\='' & \thenif? Then Do
      comment=Substr(curline.3,sc)
      c 'CLOC :'sc
      c 'CREP' Copies(' ',Length(comment))
    End
    lastchar=Right(Strip(Substr(curline.3,1,sc-1)),1)
    hit=Lastpos(lastchar,curline.3,sc-1)+2
  End
  Else hit=Length(curline.3)+2
  c 'CURSOR FILE' line.1 hit
  If rc=0 Then Do
    If sc=0 | \cmt2 & target\='' & \thenif? Then Do
      'join cursor'
      !rc=rc
      blanks=pos(word(testline,1),testline)-1
      if blanks>1 then do
        dels=blanks
        'cloc :'hit-1
        'cdel' dels
      end

      If !rc=0 Then Do
        If comment\='' Then Do
          c 'EXTRACT /CURLINE/'
          curline.3=strip(curline.3,'T')
          If Substr(curline.3,sc)='' Then c 'CREPLACE' comment
          Else c 'CAPPEND  'comment
        End
      End
      Else do
        c 'CURSOR FILE' line.1 '60'
```

```
            rc=1
          End
        End
      End
      Else Do
        c 'CURSOR FILE' line.1 '60'
        rc=1
      End
    End
  End
  eof=(rc=1|rc=2) | sc\=0 & cmt2
Return
```

PUSH/POP Source

```
/* Extract the setting and save it for POP to retrieve it */
arg vname setting extra
'extract /'setting extra'/'
if rc=0 then do
  do s=0 to value(setting'.0')
    'editv setlf' vname'_'setting'.'s value(setting'.'s)
  end
end
else do
  'msg (Push) Unknown keyword:' setting
end

/* Retrieve the setting saved by PUSH */
arg vname setting extra
'editv getf' vname'_'setting'.0'
if rc=0 then do
  do s=1 to value(vname'_'setting'.0')
    'editv getf' vname'_'setting'.'s
--    msg vname'_'setting'.'s'='value(vname'_'setting'.'s)
  end
end
else do
  'msg (POP) Unknown keyword:' setting
end
```

POPCURS

```
/* Restore the cursor */
call pop 'my cursor'
if rc\=0 then return

trace or
```

```
if my_cursor.7&gt;0 & my_cursor.8&gt;0 then --In filearea
 'cursor goto' my_cursor.7 my_cursor.8
 else do                              --Must be prefix or Home
   if my_cursor.7&gt;0 & my_cursor.8<0 then do --In prefix
     'cursor screen' my_cursor.5 my_cursor.6
   end
   else do
     if my_cursor.7<0 & my_cursor.8<0 ,
      & my_cursor.3<0 & my_cursor.4<0 then 'cursor cmdline'
     else 'cursor home'
   end
 end
return
/* Retrieve the setting saved by PUSH */
pop:
 arg vname setting extra
 'editv getf' vname'_'setting'.0'
 if datatype(value(vname'_'setting'.0'))\='NUM' then exit

 if rc=0 then do
   do s=1 to value(vname'_'setting'.0')
     'editv getf' vname'_'setting'.'s
   end
 end
 else do
   'msg (POP) Unknown keyword:' setting
 end
return
```

RUN Source

```
/*
 * A macro to execute the currently edited file as a THE/Rexx macro
 * Currently only setup for DOS,Windows,OS/2 platforms
 */
'extract /fpath/filename/line/cursor/'
origline=line.1
':1'
oldfid=fpath.1||filename.1
fn = '.\__tmprun.the'
'preserve'
Call Stream fn, 'C', 'OPEN WRITE REPLACE'
Do i = 1
   'extract /curline'
   Call Lineout fn,curline.3
   'n'
   If rc \= 0 Then Leave
```

```
End
Call Stream fn, 'C', 'CLOSE'
'macro' fn arg(1)
Address CMD 'del' fn

'extract /fpath/filename/' /* We might be in REXXOUTPUT, or other */
newfid=fpath.1||filename.1
if oldfid\=newfid then 'the' oldfid /* Aha... gotcha! */
'restore'
'top'
--'msg :'origline
':'origline
if oldfid\=newfid then 'the' newfid
```