Developing Applications with ISPF and Open Object Rexx

René Vincent Jansen
30th International Rexx Language Symposium 2019
Hursley Park, UK
Developing applications with ooRexx and ISPF

(that's ell-ess-pee-eff)
LSPF
Dialogue Manager by Daniel Erdos
Open source version of IBM’s Dialogue Manager, for Linux

Written in C++ with the Boost Library

Works with:
- linux 5.2.9
- gcc 9.1.0
- ncurses 6.1
- boost 1.69.0

For application Programming

Mostly compatible
Building the application

- Building is not particularly easy
- Very sensitive to OS and library versions
- After some attempts I chose what the author uses
- Which is Arch Linux and its toolchain
- Don’t worry, I made a Docker image
- This contains ISPF and Open Object Rexx 5.00
Start the docker container

docker run -it -v "$PWD":/test -w /test rvjansen/lspf:latest
You can fully customize this as in real ISPF
It does a nice 3.4
EDIT /test/qtime.rex

Command ==> 1

000020 /*
000021 */
000022 /* Argument words:
000023 */
000024 /* Two argument strings may be supplied. If "?" is given
000025 /* as the first argument then the program displays a
000026 /* description of itself. If a second argument is supplied
000027 /* it is used as a test value to check the operation of the
000028 /* program. This second value must be a time in the format
000029 /* HH:MM:SS, and does not have its syntax checked.
000030 */
000031 */
000032="/********** First process the argument strings **********/
000033 parse arg parm, testtime. /* get the argument strings */
000034 select
000035 when parm='?' then call tell /* say what we do */
000036 when parm=' ' then nop /* OK (no first argument) */
000037 otherwise
000038 say 'The only valid parameter to QT is "?". The argument'
000039 say 'that you supplied ("parm") has been ignored.'
000040 tell /* usually helpful to describe the program */
000041 end
000042 if testtime=' ' then now=time() /* default - use time now */
000043 else now=testtime /* caller gave test value */
000044
000045="/********** Now start processing in earnest **********/
000046 /* Nearness phrases - using a compound variable as example */
000047 near.0='' /* exact */
000048 near.1=' just gone'; near.2=' just after'
000049 near.3=' nearly'; near.4=' almost'
000050 /* after */
000051 /* before */
000052 /* Extract the hours, minutes, and seconds from the time. */
000053 parse var now hour:'min':'sec
000054
000055 if sec>59 then min=min+1 /* round up minutes */
000056 mod=min/5 /* where we are in 5 minute bracket */
000057 out="It's"near.mod /* start building the result */
000058 if min>32 then hour=hour+1 /* we are TO the hour... */
000059 min=min+2 /* shift minutes to straddle a 5-minute point */