



REXX for CICS



Agenda

- Development of REXX
- Setting up REXX for CICS
- REXX File System and Editor
- Writing REXX programs for CICS
- Other topics
 - Import/Export
 - Invoking REXX by transaction

Development of REXX

- A language was developed that combined the semantics and style of the classic programming languages
 - Algol, PL/I, Pascal
- But, included the command and string handling facilities of EXEC2
- REX was born

Development of REXX

- REXX borrowed the features of the other languages
- But, altered to make those features easy to use
- A code fragment might look like this:

```
'COPYFILE' fname ftype fmode '= BACKUP ='  
if rc>0 then say 'Copy failed with return code' rc
```

- Literals are quoted, variables do not use ampersands
- Ease of use was a necessity

Development of REXX

- First specification written March 20, 1979
- Circulated for comment
 - Included sample programs
 - The basis for most of REXX development
- The first implementation made available within IBM in late 1979
- The vast network made it easy to provide suggestions and make improvements
 - Including one that required changing hundreds of thousands of programs

Development of REXX

- Feedback made the language grow and develop
- Second 'x' added to avoid confusion
- In 1983 included in VM/SP Release 3
- First non-IBM implementation as Personal REXX (1985)
- Announced as the Procedure Language for SAA and versions available for MVS/TSO, OS/400 (1987) and OS/2 EE (1989)

Development of REXX

- 1990's
 - REXX for OS/2 2.1 and 2.2
 - Performance improvements, Multimedia enhancements and language spec 4.0
 - REXX for AIX, Netware, PC-DOS 7 and CICS
 - Lot's of REXX implementations from non-IBM sources
 - Object REXX for OS/2, Win32, AIX and Linux
 - ANSI Standard for REXX (X3.274)
 - NetRexx

REXX for CICS

- Runs on CICS Transaction Server for VSE and z/OS
- Provides for easy application development and prototyping
- Source level debug (trace)
- Full access to the complete CICS API
 - Except for Handle Condition, Handle Aid, Handle Abend, Ignore Condition, Push, and Pop
- Interface to CEDDA and CEMT

REXX for CICS

- EXECs can be shared by using EXECLOAD
- DB2 access support
- CICS MAP support is available for full screen applications
 - A full screen panel definition facility is also provided
- VSAM I/O can be done with standard API features
- REXX File System (RFS) is used to store REXX programs and data

REXX for CICS

- RFS is implemented as VSAM files
- VSE Librarian can also be used for program storage
 - Files in RFS can be moved to/from a VSE sublibrary
- CICS programs (written in any language) can be invoked via START, XCTL or LINK
- EXECIO can be used to do I/O to temporary storage and transient data queues
- Used REXX to prototype the code for Web Services articles and presentations, before being re-written in COBOL
- Implementation information is in
'CICS Transaction Server for VSE/ESA 1.0 - REXX Guide', Appendix K

Setting up REXX for CICS

- REXX programs, full screen panels and data are stored in the RFS
- The RFS is implemented as two or more VSAM files
- As distributed with VSE
 - POOL1: RFSDIR1 and RFSPOL1
 - POOL2: RFSDIR2 and RFSPOL2
- Sample JCL in PRD1.BASE as CICVSAM.J
 - Can be used to define your own

Setting up REXX for CICS

- CICS definitions are in group CICREXX
 - Including files
 - Make sure LSR pools include keylength =>252 and CI Sizes of at least 18K
 - Add any pools that you define
- Pre-defined REXX procedures need to be copied/renamed from PRD1.BASE
 - Copy into a sublibrary in the PROC search chain for CICS
 - COPY CIC*.Z:=.PROC

Setting up REXX for CICS

- Filepool(s) need to be formatted
- User authorization required
 - SYSA set up by default
- If another user needs to be able to issue authorized commands
 - In CICSTART.PROC
 - Add user to
 - 'AUTHUSER RCUSER SYSA'

Setting up REXX for CICS

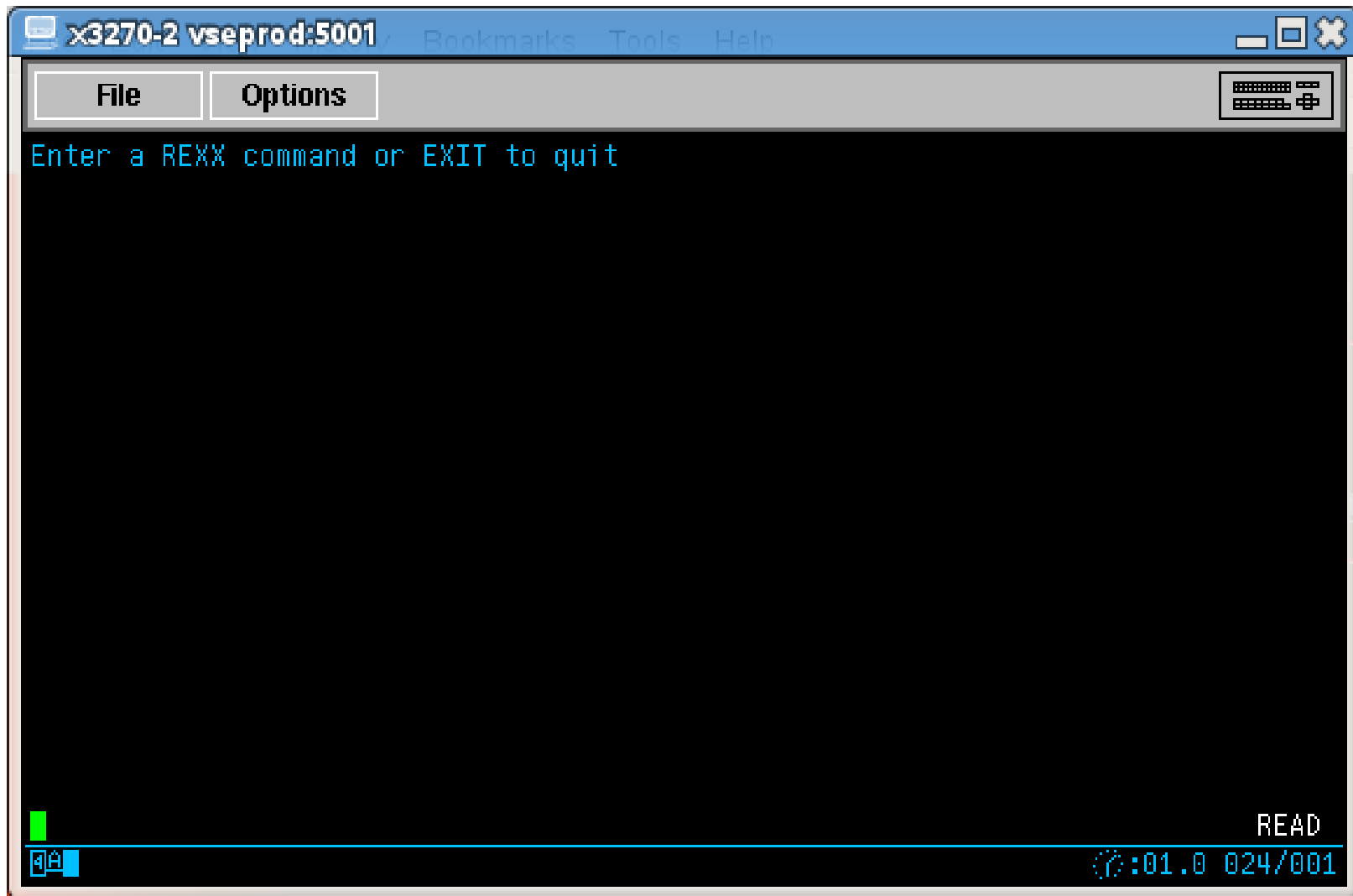
- Filepools defined by

```
'FILEPOOL DEFINE POOL1 RFSDIR1 RFSPOL1 (USER'
```

- Pool name POOL1
- Clusters RFSDIR1 and RFSPOL1
- DIR is the directory
- POL is the data storage
- One or more POL files can be defined

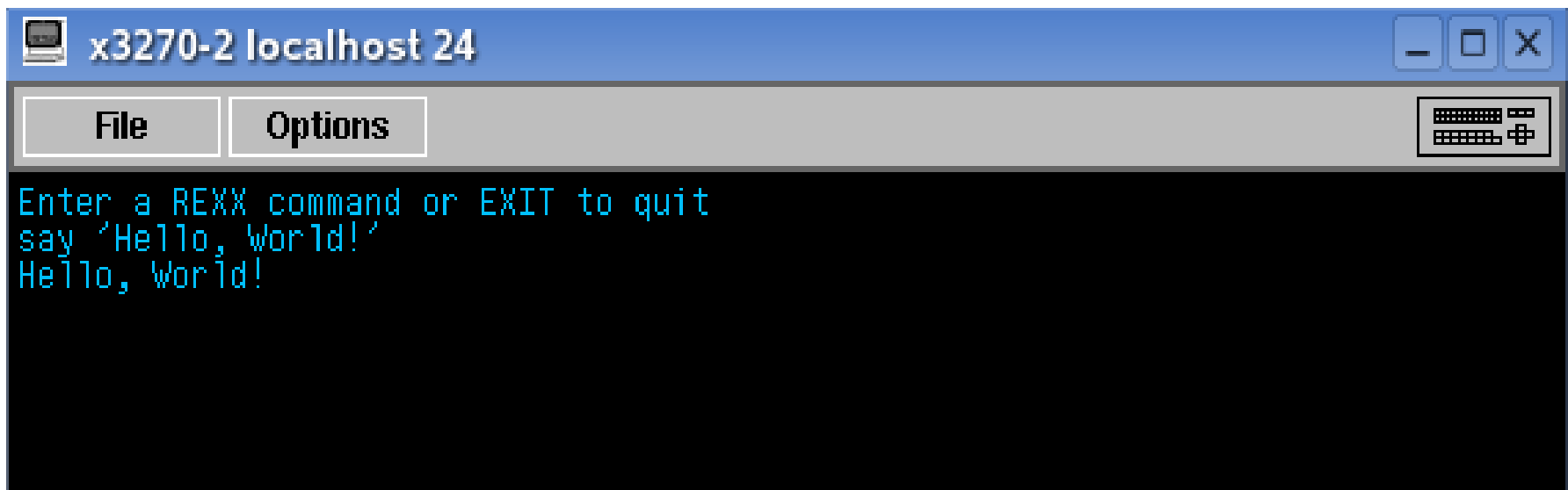
Setting up REXX for CICS

- Type REXX on a clear screen



Setting up REXX for CICS

- REXX clauses can be entered here and immediately executed
- Works like REXXTRY on other platforms
 - Called CICRTRY
- **Type** say 'Hello, World!'



The screenshot shows a terminal window titled "x3270-2 localhost 24". The window has a menu bar with "File" and "Options" buttons. The main area displays the following text in cyan on a black background:

```
Enter a REXX command or EXIT to quit
say 'Hello, World!'
Hello, World!
```

Setting up REXX for CICS

- Each pool listed in CICSTART.PROC will need to be formatted

```
FILEPOOL FORMAT POOL1
```

- No response should be returned when the command is successful
- A -4 return code may mean that the user does not have authority to use the commands
- When MORE appears on the bottom right, press CLEAR or ENTER to continue
- PF12 can be used to retrieve previous commands

REXX File System and Editor

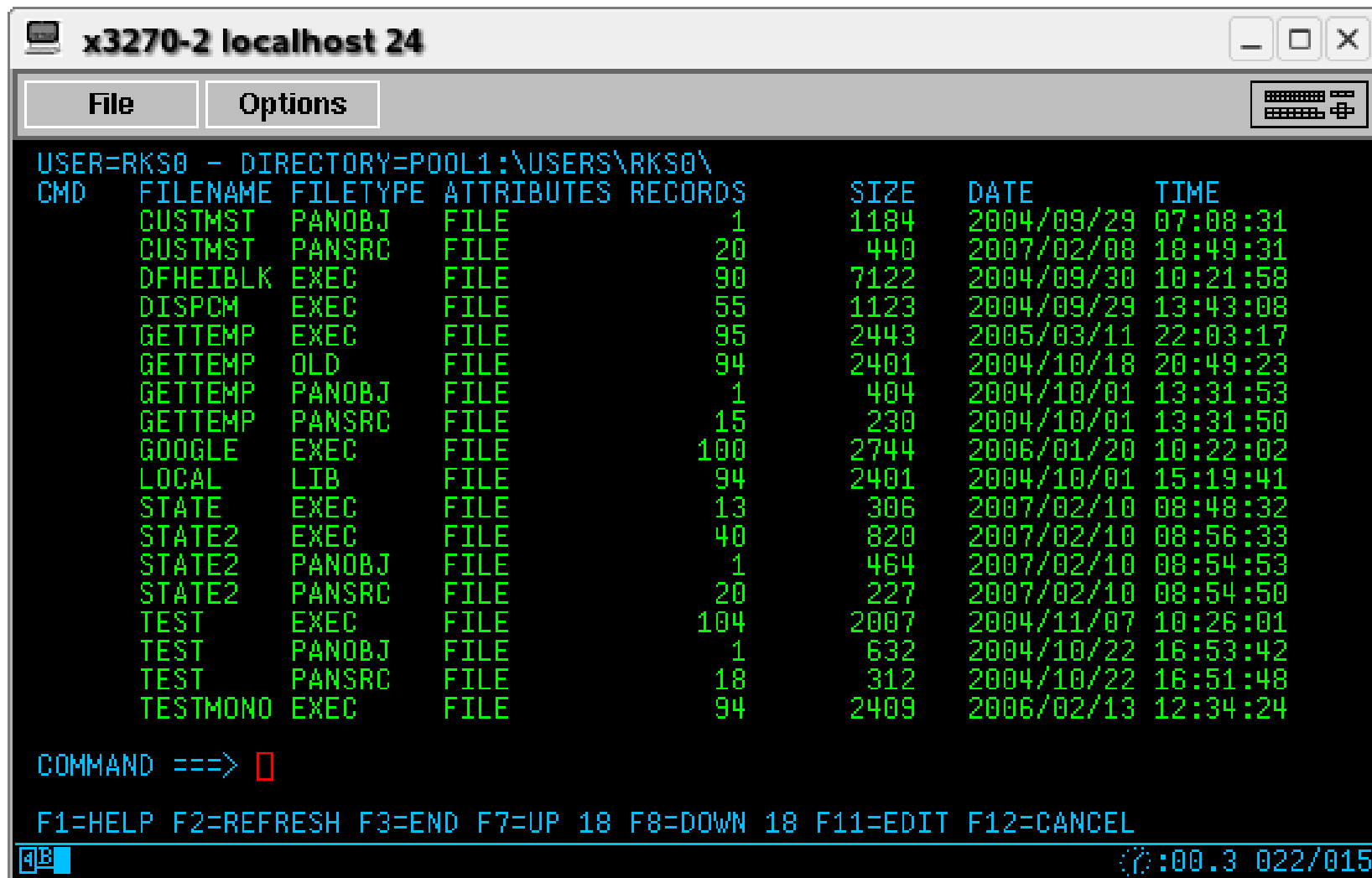
- Provides a hierarchical file system
- Can be used for program and data storage
- Must be used to store panel definitions
- A filelist utility is provided to display files in the pool
- The editor looks feels and acts a lot like Xedit
- Commands to manipulate files can be used from REXX
 - Read, Write, Check Dir, Check File, Delete, Copy...

REXX File System and Editor

- The FLST transaction displays the filelist
- FLST looks, feels and acts like FILELIST on CMS
- COPY, RENAME, DELETE commands act on files
 - Even PF11 to Edit a file
- In an empty directory the EDIT command on the command line can be used to edit a file
 - EDIT STATE.EXEC

REXX File System and Editor

- Filelist (FLST transaction)



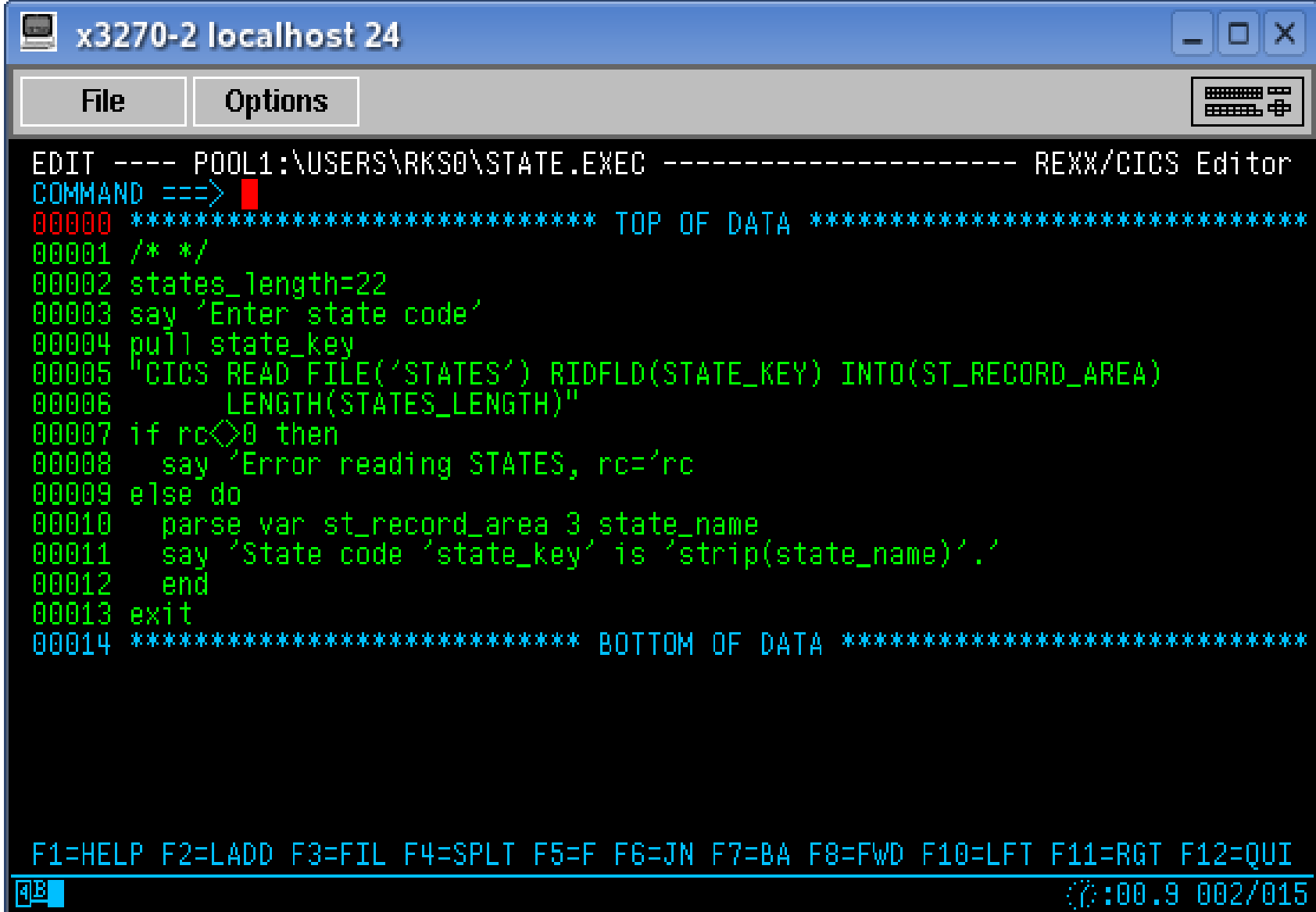
The screenshot shows a window titled "x3270-2 localhost 24" with a menu bar containing "File" and "Options". The main display area shows a filelist for the user "RKS0" in the directory "POOL1:\USERS\RKS0\". The filelist is a table with columns: CMD, FILENAME, FILETYPE, ATTRIBUTES, RECORDS, SIZE, DATE, and TIME. The files listed include CUSTMST, DFHEIBLK, DISPCM, GETTEMP, GOOGLE, LOCAL, STATE, STATE2, TEST, and TESTMONO. At the bottom, there is a command prompt "COMMAND ==>" and a footer with function key definitions: "F1=HELP F2=REFRESH F3=END F7=UP 18 F8=DOWN 18 F11=EDIT F12=CANCEL". The status bar at the very bottom shows "022/015".

```
USER=RKS0 - DIRECTORY=POOL1:\USERS\RKS0\  
CMD  FILENAME FILETYPE ATTRIBUTES RECORDS      SIZE  DATE      TIME  
CUSTMST  PANOBJ  FILE      1      1184  2004/09/29 07:08:31  
CUSTMST  PANSRC  FILE     20      440  2007/02/08 18:49:31  
DFHEIBLK EXEC    FILE     90     7122  2004/09/30 10:21:58  
DISPCM   EXEC    FILE     55     1123  2004/09/29 13:43:08  
GETTEMP  EXEC    FILE     95     2443  2005/03/11 22:03:17  
GETTEMP  OLD     FILE     94     2401  2004/10/18 20:49:23  
GETTEMP  PANOBJ  FILE      1      404  2004/10/01 13:31:53  
GETTEMP  PANSRC  FILE     15      230  2004/10/01 13:31:50  
GOOGLE   EXEC    FILE    100     2744  2006/01/20 10:22:02  
LOCAL    LIB     FILE     94     2401  2004/10/01 15:19:41  
STATE    EXEC    FILE     13      306  2007/02/10 08:48:32  
STATE2   EXEC    FILE     40      820  2007/02/10 08:56:33  
STATE2   PANOBJ  FILE      1      464  2007/02/10 08:54:53  
STATE2   PANSRC  FILE     20      227  2007/02/10 08:54:50  
TEST     EXEC    FILE    104     2007  2004/11/07 10:26:01  
TEST     PANOBJ  FILE      1      632  2004/10/22 16:53:42  
TEST     PANSRC  FILE     18      312  2004/10/22 16:51:48  
TESTMONO EXEC    FILE     94     2409  2006/02/13 12:34:24  
  
COMMAND ==>   
  
F1=HELP F2=REFRESH F3=END F7=UP 18 F8=DOWN 18 F11=EDIT F12=CANCEL  
022/015
```

Writing REXX Programs

- A program to look up state names with the two character state code
- Asks the user for the state code
- Look up the state code in the file
- If found, display the state code and state name
- The CICS environment is used to issue CICS commands
 - The return code is the EIBRESP code

Writing REXX Programs



```
EDIT ---- POOL1:\USERS\RKSQ\STATE.EXEC ----- REXX/CICS Editor
COMMAND ==>
00000 ***** TOP OF DATA *****
00001 /* */
00002 states_length=22
00003 say 'Enter state code'
00004 pull state_key
00005 "CICS READ FILE('STATES') RIDFLD(STATE_KEY) INTO(ST_RECORD_AREA)
00006         LENGTH(STATES_LENGTH)"
00007 if rc<>0 then
00008     say 'Error reading STATES, rc='rc
00009 else do
00010     parse var st_record_area 3 state_name
00011     say 'State code 'state_key' is 'strip(state_name)'.'
00012     end
00013 exit
00014 ***** BOTTOM OF DATA *****

F1=HELP F2=LADD F3=FILE F4=SPLT F5=F F6=JN F7=BA F8=FWD F10=LFT F11=RGF F12=QUI
(?:00.9 002/015
```

Writing REXX Programs

- Program can be executed from FLST
 - EXEC line command
- Command line
 - EXEC STATE.EXEC

```
Enter state code  
State code WI is Wisconsin.  
STATE.EXEC complete; rc=0  
Press the ENTER key to continue...
```

```
Enter state code  
Error reading STATES, rc=13  
STATE.EXEC complete; rc=0  
Press the ENTER key to continue...
```

Writing REXX Programs

- REXX for CICS provides a panel definition feature
- Used for creating full screen applications
- Panels are defined in the RFS (and must be in the RFS)
- Panels are 'compiled' before use
 - Compiling is automatic at first reference
 - Also when the panel source changes
- BMS maps can be used with REXX programs

Writing REXX Programs

- Panel definition uses locally defined special characters to identify 3270 attribute characters
- The `.DEFINE` verb(s) must be first in the file
- One or more attributes can be configured per attribute character
- The `.PANEL` verb designates the beginning of the display area
- Another `.PANEL` verb designates the end of the display area
- Save source as `.PANSRC`

Writing REXX Programs

```
.DEFINE < BLUE PROTECT  
.DEFINE @ BLUE SKIP  
.DEFINE ! RED PROTECT  
.DEFINE > GREEN UNPROTECT UNDERLINE  
.DEFINE # GREEN NUMERIC RIGHT UNDERLINE  
  
.PANEL STATE2
```

```
@ State Code:>&s@
```

```
@ State Name:<&state_name @
```

```
!&errmsg
```

Writing REXX Programs

- Program to drive our panel will be an extension of the first program
- It asks for a state code
- When it is entered it will look it up on the file and display the name
- Better file access error handling

Writing REXX Programs

- PANEL SEND will write the panel section of the panel file to the terminal
- PANEL RECEIVE will read the panel from the terminal
- Special stem variable pan. contains 3270 attributes
 - pan.aid – AID key
 - pan.curs – Cursor location
 - pan.cnam – REXX variable at cursor location

```

/* */
states_length=22
s=''
state_name=''
errmsg=''
do forever
  'PANEL SEND STATE2 CURSOR(S)'
  if rc>4 then signal panel_error
  'PANEL RECEIVE STATE2'
  if pan.aid='PF3' then
    leave
  state_key=translate(s)
  "CICS READ FILE('STATES') RIDFLD(STATE_KEY) INTO(ST_RECORD_AREA)
    LENGTH(STATES_LENGTH)"
  select
    when rc=0 then do
      errmsg=''
      parse var st_record_area 3 state_name
      end
    when rc=12 then
      errmsg='STATES file is not defined'
    when rc=13 then do
      state_name=''
      errmsg=s' not found on STATES file'
      end
    when rc=19 then
      errmsg='STATES file is not open'
    when rc=84 then
      errmsg='STATES file is disabled'
    otherwise
      errmsg='Return code='rc 'on STATES read'
    end
  end
end
exit

```

```

panel_error:
say 'Panel error, rc='rc
say 'Reason code='pan.rea
say 'Location='pan.log
exit

```

```
x3270-2 localhost 24
File Options
State code: L
State name:
PF: 3=END
```

```
x3270-2 localhost 24
File Options
State code: WI
State name: Wisconsin
PF: 3=END
```

```
x3270-2 localhost 24
File Options
State code: MM
State name:
MM not found on STATES file
PF: 3=END
:00.0 002/015
```

Import/Export

- Objects in RFS can be moved to and from a VSE sublibrary
 - Transfer between systems
 - Code delivery
 - Make data available for other use
- Import and Export use full file path
 - For example:

```
EXPORT POOL1:\USERS\RKS0\STATE.EXEC LOCAL.LIB(STATE.EXEC)
```

Invoking REXX by a transaction

- REXX programs can be triggered via a tranid
- In CICSTART.PROC use the DEFTRNID command

```
'DEFTRNID STA STATE2.EXEC'
```

- Use CEDA to define transaction

```
CEDA DEF TRANS(STA) PROG(CICREXD) GR(STA) TWASIZE(32)
```

- Install the group, update transaction security

Conclusion

- REXX for CICS provides an alternative to other languages for application prototyping or fast application development
- VSAM and DB2 data resources are available to REXX for CICS
- REXX for CICS can use the full CICS API which will give your programs access to all CICS functions



Questions?

Rich Smrcina
VM Assist
414-491-6001
rsmrcina@vmassist.com

Specializing in support of z/VSE,
Linux on System z and z/VM systems