Your Business ▪ Your Agenda ▪ Our Passion
Maximizing the value of IT together

REXX for CL Programmers!
TUG Mar 20, 2013

Mike Warkentin
Managing Director R&D
mwarkentin@rocketsoftware.com
(781) 577-4344

AGENDA

- What is REXX
- How Does it Differ from CL
  - When to Use it
- Basic Constructs
- Creating and Running REXX Programs
- Variables
- REXX Expressions
- REXX Instructions
- Some Examples

What is REXX?

REstructured eXtended eXecutor language

- Designed by Michael Cowlishaw of IBM UK
- "Own time project" – Mar 20 1979 – Mid 1982
- "REXX is a procedural language that allows programs and algorithms to be written in a clear and structured way"
- Built to replace EXEC and EXEC2
- First Public Exposure – SHARE 56, Texas 1981

Where can I run REXX?

- VM/CMS, VM/CGS, MVS TSO/E, AS/400, OS/2, VSE/ESA, AIX, CICS/ESA, PC DOS
- IBM has also provided versions for Novell Netware, Windows, JAVA & LINUX

What is REXX?

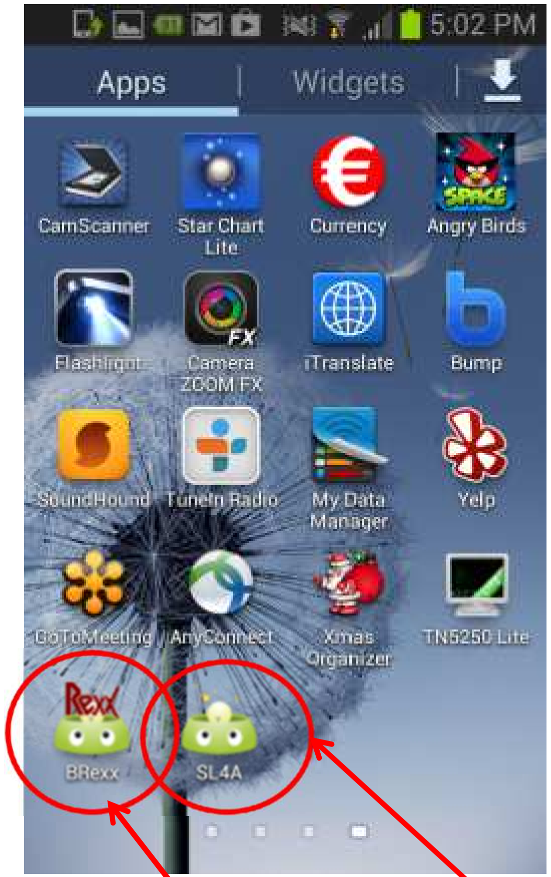
Where else can I run REXX (non IBM)

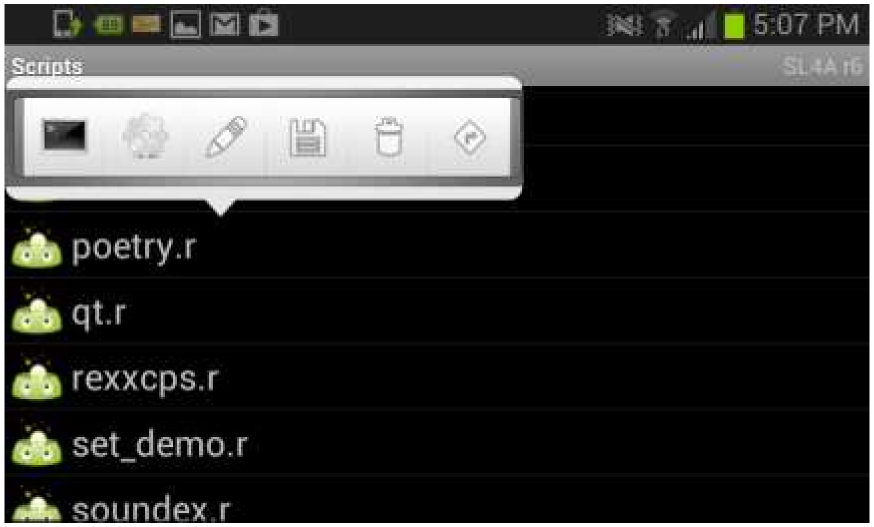
- PC/DOS by Charles Daney in 1984/85
- Atari, Amiga, Unix, Solaris, DEC, Windows, WinCE, Pocket PC, MS-DOS, Palm OS, QNX, OS/2, Linux, BeOS, EPOC32, AtheOS, OpenVMS, OpenEdition, Macintosh, MacOS/X, ANDROID, iOS (jail brake)

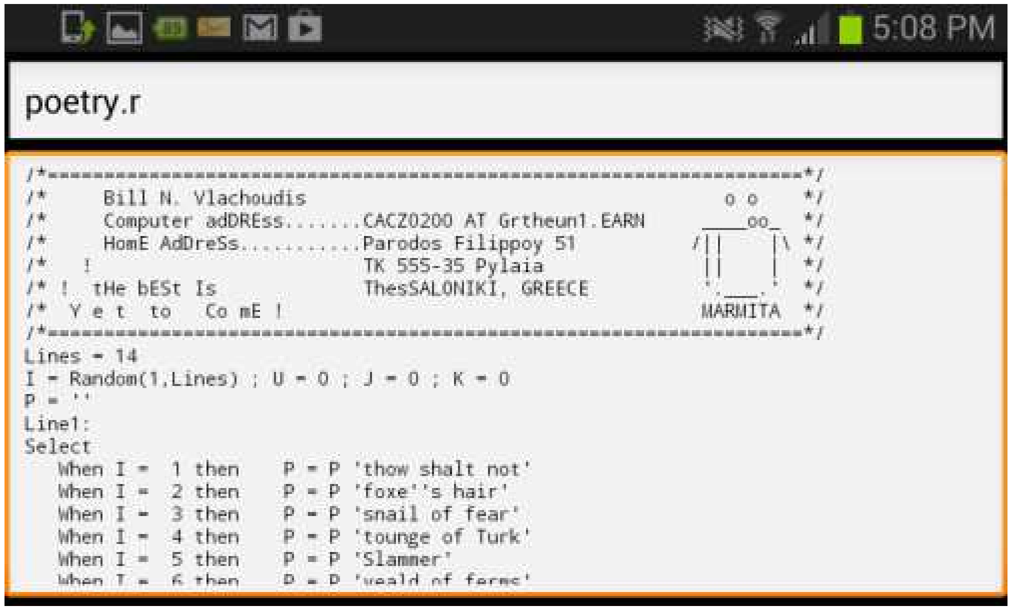
But there is more…

- Windows and Linux opensource ports – Regina & REXX/imc
- NetRexx (compiles to JAVA byte code)
- ObjectRexx – OO version

Where can REXX Run?







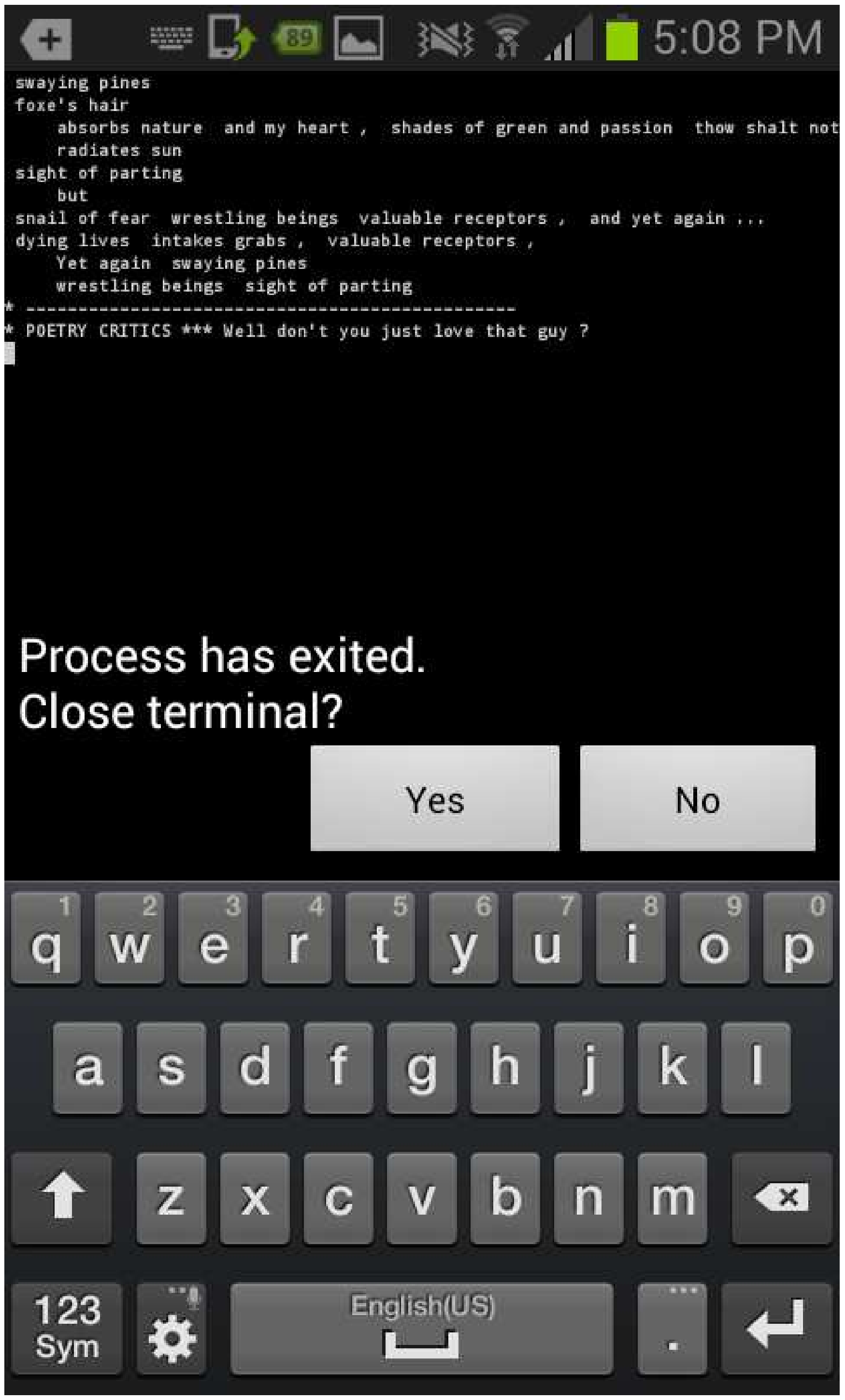
```
/*=========================================================================*/
/*     Bill N. Vlachoudis                                          o o     */
/*     Computer adDREss.......CACZ0200 AT Grtheun1.EARN          ____oo_   */
/*     HomE AdDreSs...........Parodos Filippoy 51               /||     |\ */
/*   !                        TK 555-35 Pylaia                   ||     |  */
/* !  tHe bESt Is             ThesSALONIKI, GREECE               '.___.'   */
/*  Y e t  to   Co mE !                                         MARMITA    */
/*=========================================================================*/
Lines = 14
I = Random(1,Lines) ; U = 0 ; J = 0 ; K = 0
P = ''
Line1:
Select
   When I =  1 then    P = P 'thow shalt not'
   When I =  2 then    P = P 'foxe''s hair'
   When I =  3 then    P = P 'snail of fear'
   When I =  4 then    P = P 'tounge of Turk'
   When I =  5 then    P = P 'Slammer'
   When I =  6 then    P = P 'veald of ferns'
```

Android Smartphones

- Download Scripting Layer for Android at: http://code.google.com/p/android-scripting/
- Download Brexx.apk at: http://pceet075.cern.ch/bnv/brexx/

Where can REXX Run?

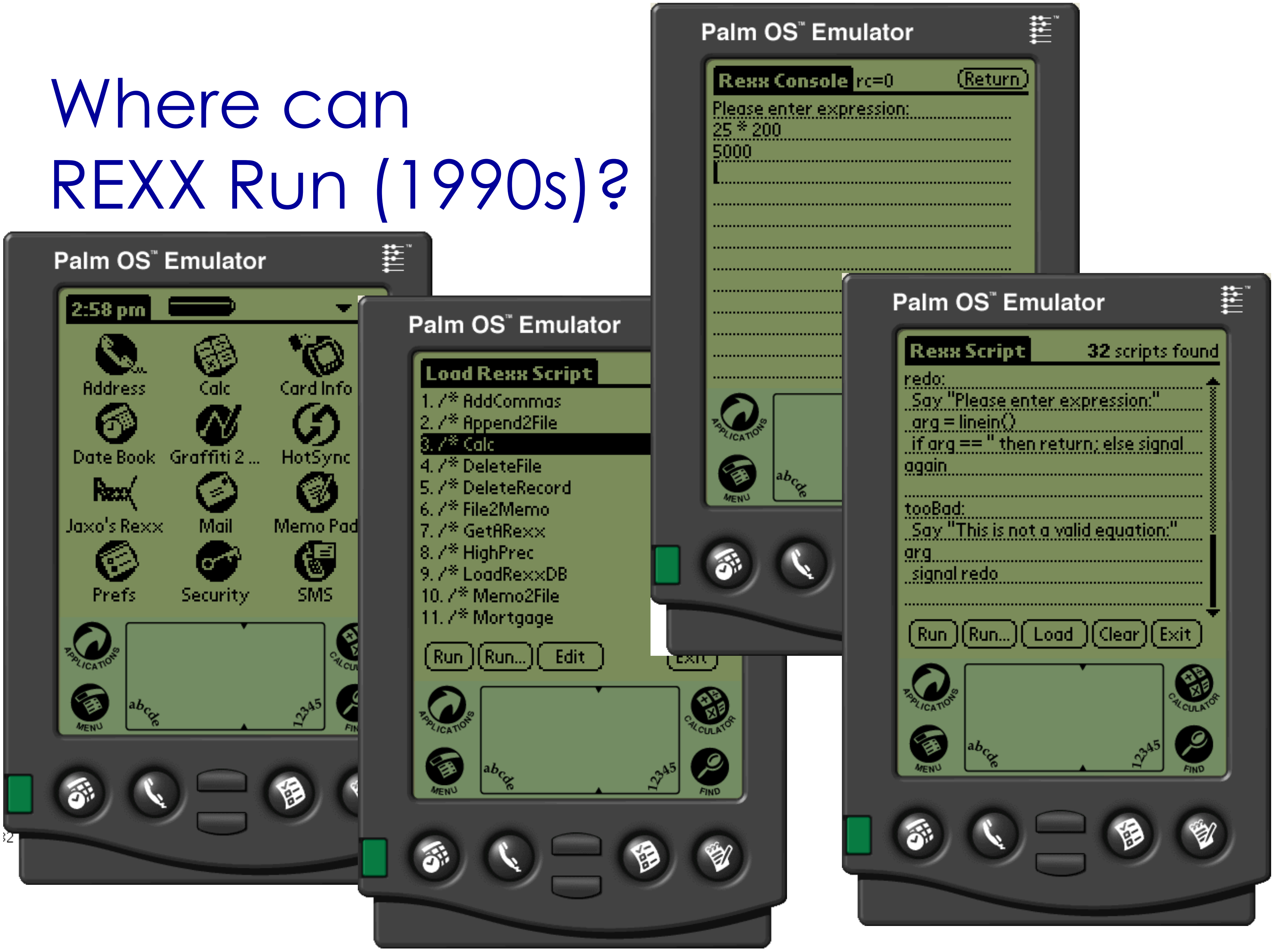
NOTE: No way to run on iOS (Ipad or iPhone) without a jailbreak!



```
swaying pines
foxe's hair
    absorbs nature  and my heart ,  shades of green and passion  thow shalt not
    radiates sun
sight of parting
    but
snail of fear  wrestling beings  valuable receptors ,  and yet again ...
dying lives  intakes grabs ,  valuable receptors ,
    Yet again  swaying pines
    wrestling beings  sight of parting
* ----------------------------------------------
* POETRY CRITICS *** Well don't you just love that guy ?
```

Process has exited.
Close terminal?

Where can REXX Run (1990s)?



Load Rexx Script:

```
1. /* AddCommas
2. /* Append2File
3. /* Calc
4. /* DeleteFile
5. /* DeleteRecord
6. /* File2Memo
7. /* GetARexx
8. /* HighPrec
9. /* LoadRexxDB
10. /* Memo2File
11. /* Mortgage
```

Rexx Console:

```
Please enter expression:
25 * 200
5000
```

Rexx Script:

```
redo:
 Say "Please enter expression:"
 arg = linein()
 if arg == '' then return; else signal
again

tooBad:
 Say "This is not a valid equation:"
arg
 signal redo
```

What is REXX continued…

REXX is: an interpreted language

- It's not compiled like CL or RPG
- When REXX pgm runs, language processor directly interprets each statement
- Can be more resource intensive then compiled programs

REXX is: free format

- No line numbers required
- Instructions can span multiple lines or many instructions on one line
- Begin in any column
- Skip lines
- Type in uppercase, lowercase, mixed, REXX doesn't care!
- Could be messy!

What is REXX continued…

REXX is: string based

- All data is a character string
- No need to declare the variable type
- Strong parsing functions for assigning variables to/from different input/output sources

REXX:

- Is ANSI compliant, SAA, portable across platforms
- Simple to use and traceable
- Contains built in functions for processing, searching & comparison ops for text and numbers, formatting and arithmetic operations

Variants of REXX

Classic Rexx

- The original procedural language developed by IBM
- Six free classic interpreters available
- Bundled with many operating systems like VM/SP, MVS, OS/2, PC DOS, Windows NT, IBM i, System z etc.
- Used as a “glue language” or macro language and primary scripting language in many OSes
- http://www.rexxla.org

Variants of REXX

NetRexx

- Open source variant that that runs on a JAVA Virtual Machine
- Both compiled and interpretive
- Additional constructs to support Object Oriented Programing
- Develop applets, applications, servlets, classes and beans
- Originally IBM owned – now owned by Rexx Language Association
- http://www.netrexx.org/

Object REXX (or Open Object Rexx)

- Object oriented scripting language initially built by IBM for OS/2
- Includes classes, messaging, single and multiple inheritance, encapsulation, data hiding, polymorphism etc.
- Large class library
- Available for multiple platforms like Linux, Solaris, Windows
- Open source and upwardly compatible with Classic Rexx
- http://sourceforge.net/projects/oorexx/

Many interpreters available too…

- Regina Rexx
  - Most widely used
  - http://regina-rexx.sourceforge.net/
- Reginald
  - Enhanced and extended for Windows
  - http://home.roadrunner.com/~jgglatt/rexx/win32/rxusrw32.htm
- R4 and Roo
  - Also extended for Windows
  - http://www.kilowattsoftware.com/
- Brexx
  - Very fast lightweight Rexx for PDAs, smartphones, embedded apps etc
  - http://sourceforge.net/projects/brexx/
- Rexx/imc
  - For Linux, Unix and BSD platforms
  - http://www.cs.ox.ac.uk/people/ian.collier/Rexx/rexximc.html

What is REXX continued...

REXX is:

- Popular …over 416,000 Google Hits on REXX!
- Long running…34 years old on March 20th 2004

REXX is this:

```
/* Count to ten and add the numbers up */
  sum = 0
  do count = 1 to 10
    say count
    sum = sum + count
  end
  say "The sum of these numbers is" sum"."
```

What is REXX continued…

REXX is:

- According to John Dvorak in his article for ZDNET Get REXX – It Pays

  "…it's apparent that REXX is something of a Swiss Army Knife among programming languages"

How does it differ from CL?

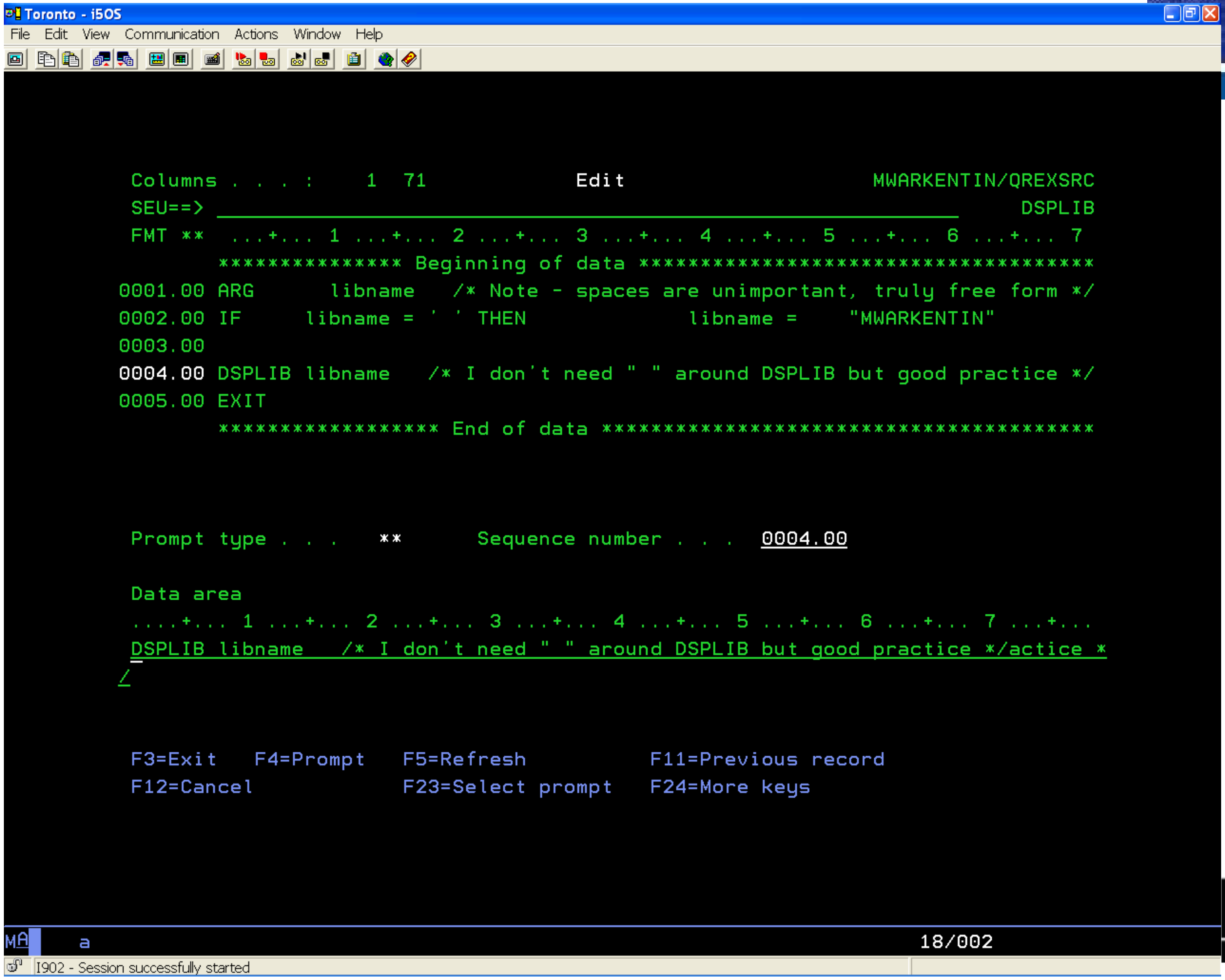
CL

- Compiled
- Handle up to 5 files at i5OS
- Lousy at string manipulations
- IBM i only
- Variables must be declared and have a type
- GOTO
- Command prompting, syntax checking standard on i

REXX

- Interpreted
- No file handling capabilities at all!
- The string expert – it's all strings!
- Runs almost anywhere
- Variables are all strings – no type and no declaration
- NO GOTO
- No command prompting, syntax checking on i BUT…

Prompting looks like this in REXX



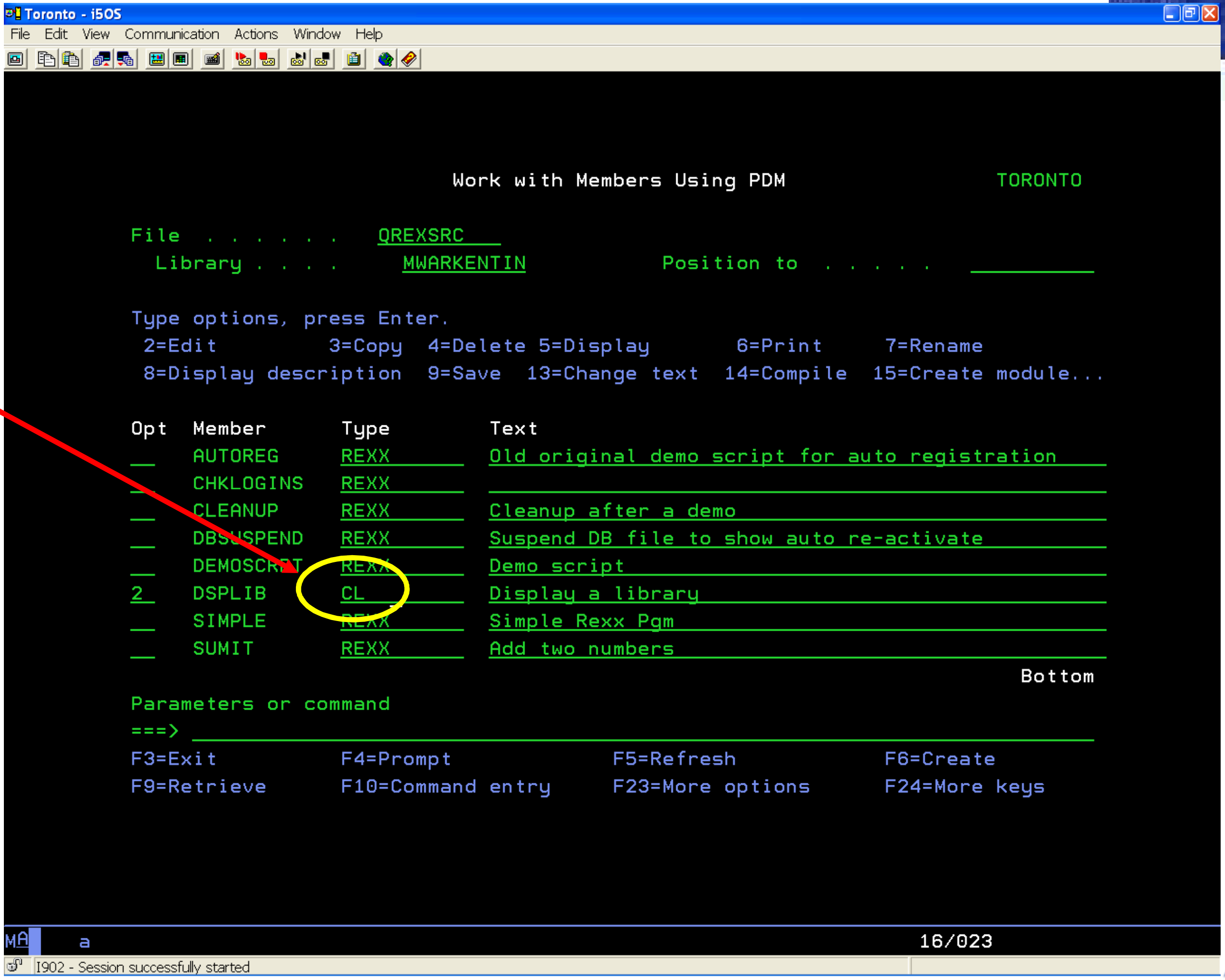
```
 Columns . . . :    1  71            Edit                       MWARKENTIN/QREXSRC
 SEU==> ____________________________________________________________      DSPLIB
 FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
        *************** Beginning of data *************************************
0001.00 ARG      libname   /* Note - spaces are unimportant, truly free form */
0002.00 IF     libname = ' ' THEN             libname =    "MWARKENTIN"
0003.00
0004.00 DSPLIB libname   /* I don't need " " around DSPLIB but good practice */
0005.00 EXIT
        ****************** End of data ****************************************


 Prompt type . . .   **      Sequence number . . .  0004.00

 Data area
 ....+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+...
 DSPLIB libname   /* I don't need " " around DSPLIB but good practice */actice *
/


 F3=Exit   F4=Prompt   F5=Refresh           F11=Previous record
 F12=Cancel            F23=Select prompt    F24=More keys

MA     a                                                       18/002
I902 - Session successfully started
```

But you can do this…

- If you have lots of CL commands in the REXX source member…

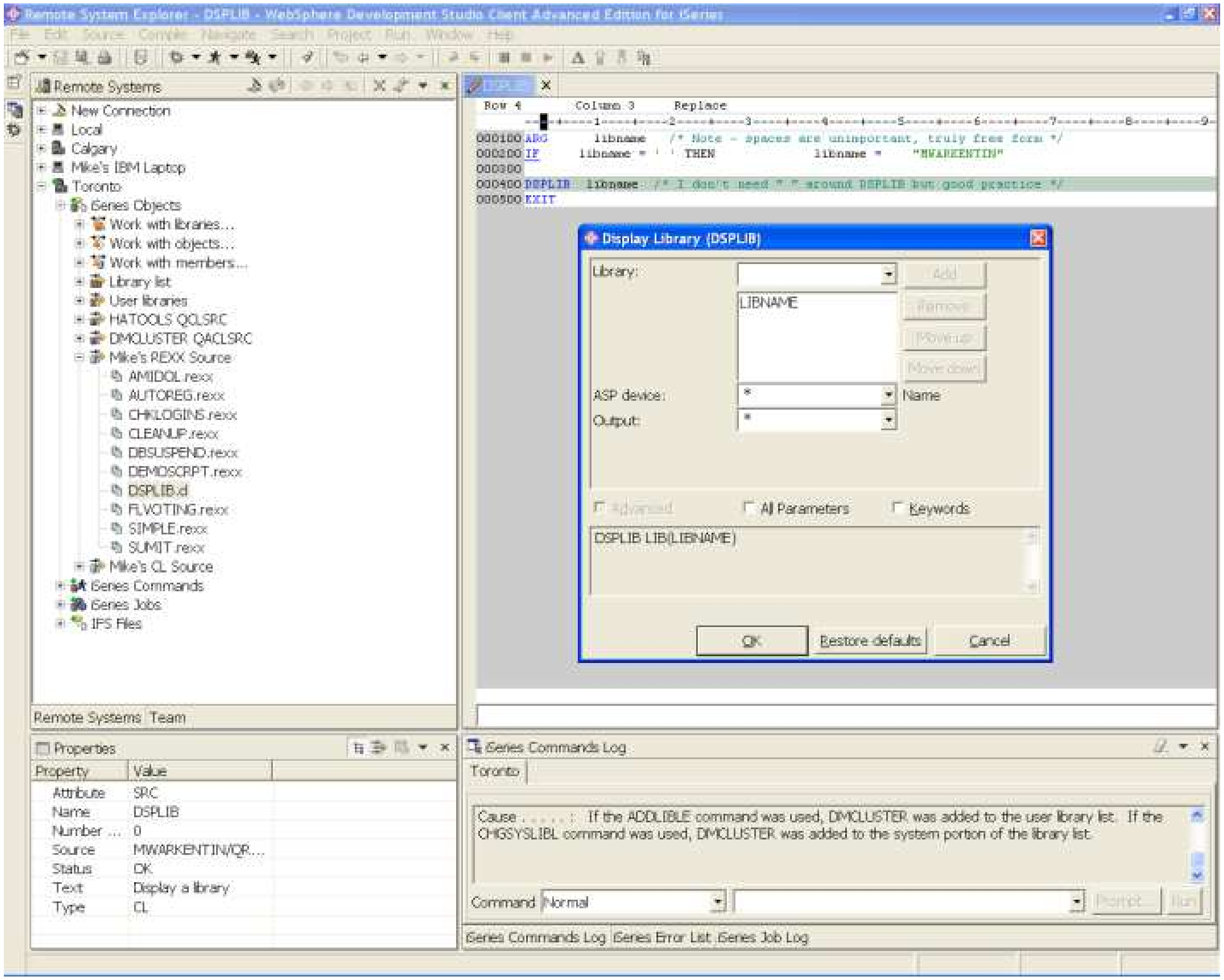
Now I can prompt all the CL commands I like!!

Just remember to change it back



```
                         Work with Members Using PDM                 TORONTO

 File  . . . . . .   QREXSRC
   Library . . . .     MWARKENTIN          Position to  . . . . .

 Type options, press Enter.
  2=Edit         3=Copy  4=Delete 5=Display       6=Print     7=Rename
  8=Display description  9=Save  13=Change text  14=Compile  15=Create module...

 Opt  Member      Type        Text
 __   AUTOREG     REXX        Old original demo script for auto registration
 __   CHKLOGINS   REXX
 __   CLEANUP     REXX        Cleanup after a demo
 __   DBSUSPEND   REXX        Suspend DB file to show auto re-activate
 __   DEMOSCRPT   REXX        Demo script
 2_   DSPLIB      CL          Display a library
 __   SIMPLE      REXX        Simple Rexx Pgm
 __   SUMIT       REXX        Add two numbers
                                                                      Bottom
 Parameters or command
 ===>
 F3=Exit          F4=Prompt             F5=Refresh            F6=Create
 F9=Retrieve      F10=Command entry     F23=More options      F24=More keys

MA      a                                                        16/023
I902 - Session successfully started
```

In WDSc I can PF4 on DSPLIB



```
Row 4          Column 3     Replace
000100 ARG        libname   /* Note - spaces are unimportant, truly free form */
000200 IF      libname = ' '  THEN              libname =     "MWARKENTIN"
000300
000400 DSPLIB  libname  /* I don't need " " around DSPLIB but good practice */
000500 EXIT
```

Display Library (DSPLIB)

Library:	LIBNAME
ASP device:	*
Output:	*

```
DSPLIB LIB(LIBNAME)
```

Property	Value
Attribute	SRC
Name	DSPLIB
Number ...	0
Source	MWARKENTIN/QR...
Status	OK
Text	Display a library
Type	CL

```
Cause . . . . . :   If the ADDLIBLE command was used, DMCLUSTER was added to the user library list.  If the
CHGSYSLIBL command was used, DMCLUSTER was added to the system portion of the library list.
```

So when should I use CL vs. REXX vs. RPG…

- If you need to manipulate strings or want an interactive response with the end user

  …use REXX

- If you need to manipulate files, write reports or do pretty screens

  …use RPG

- If you need to do a little of everything

  …use CL

For example…

- A program to find the first sentence (delimited by a period) in a 50 char variable &INPUT and place the remaining text in a second variable &REMAINDER looks like:

```
        DCL &INPUT *CHAR LEN(50)
        DCL &REMAINDER *CHAR LEN(50)
        DCL &X *DEC LEN(2 0) VALUE(1)
        DCL &L *DEC LEN(2 0)                  */remaining length */

SCAN:   IF ((%SUBSTRING(&INPUT &X 1) *NE '.') *AND +
           (&X *LT 50)) THEN(DO)
          CHGVAR &X (&X + 1)
           GOTO SCAN
         ENDDO
        CHGVAR VAR(&L) VALUE(50  - &X)
        CHGVAR VAR(&X) VALUE(&X + 1)
        CHGVAR VAR(&REMAINDER) VALUE(%SUBSTRING(&INPUT &X &L))
```

For example…

- Or in REXX…

```
parse var input . ‘.’ remainder
```

Samples provided by
REXX/400 Programmers Guide V4R1

Or even...

- A program to extract three words, with leading and trailing blanks removed from a 30 char field and assign them to variables &LIB, &FILE and &MBR:

```
            DCL &INPUT *CHAR LEN(30)
            DCL &LIB *CHAR LEN(30)
            DCL &FILE *CHAR LEN(10)
            DCL &MBR *CHAR LEN(10)
            DCL &S *DEC LEN(2 0)              /* Starting position     */
            DCL &E *DEC LEN(2 0)              /* Ending position       */
            DCL &L *DEC LEN(2 0)              /* Length of parameter */

            CHGVAR &S 1  */Remove leading blanks for &LIB */

LIBSTR:     IF (%SST(&LIB &S 1) *EQ ‘ ’) THEN(DO)
                 CHGVAR &S (&S + 1)
              GOTO LIBSTR
            ENDDO
            CHGVAR &E (&S + 1)                /* Find end of &LIB */
LIBEND:     IF (%SST(&LIB &E 1) *NE ‘ ’) THEN(DO)
                 CHGVAR &E (&E + 1)
              GOTO LIBEND
            ENDDO
            CHGVAR &L (&E - &S)
            CHGVAR &LIB (%SST(&LIB &S &L))
```

Continuing…

```
             CHGVAR &S (&E + 1)              */ Remove leading blanks for &FILE */
FILSTR:      IF (%SST(&FILE &S 1) *EQ ‘ ’) THEN(DO)
                 CHGVAR &S (&S + 1)
               GOTO FILSTR
             ENDDO
             CHGVAR &E (&S + 1)              /* Find end of &FILE */
FILEND:      IF (%SST(&FILE &E 1) *NE ‘ ’) THEN(DO)
                 CHGVAR &E (&E + 1)
               GOTO FILEND
             ENDDO
             CHGVAR &L (&E - &S)
             CHGVAR &FILE (%SST(&FILE &S &L))

             CHGVAR &S (&E + 1)              */ Remove leading blanks for &MBR */
MBRSTR:      IF (%SST(&MBR &S 1) *EQ ‘ ’) THEN(DO)
                 CHGVAR &S (&S + 1)
               GOTO MBRSTR
             ENDDO
             CHGVAR &E (&S + 1)              /* Find end of &MBR */
MBREND:      IF (%SST(&MBR &E 1) *NE ‘ ’) THEN(DO)
                 CHGVAR &E (&E + 1)
               GOTO MBREND
             ENDDO
             CHGVAR &L (&E - &S)
             CHGVAR &MBR (%SST(&MBR &S &L))
```

Compared to…

- Or in REXX…

```
parse var input lib file mbr
```

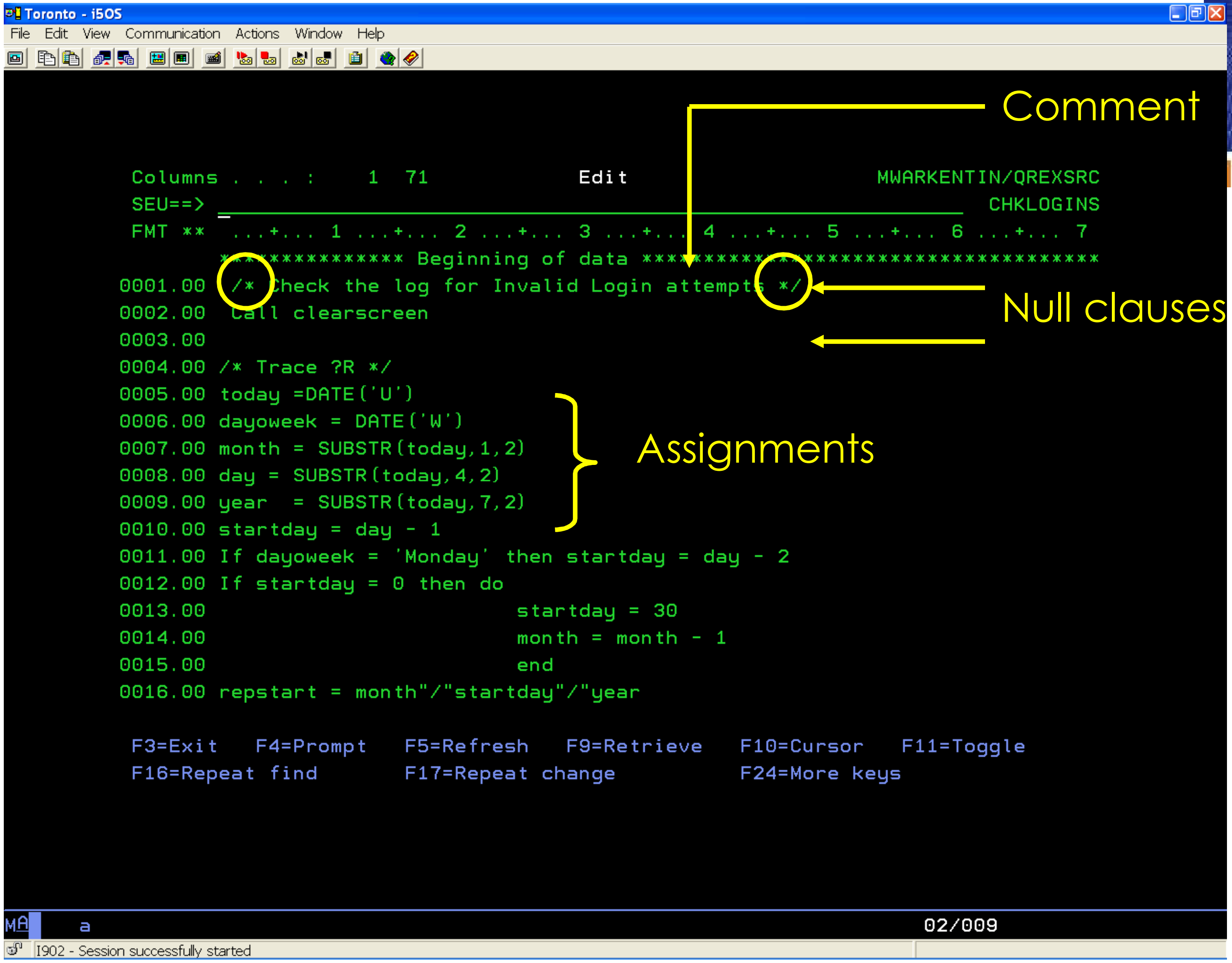
WHAT WOULD YOU RATHER CODE?

Samples provided by
REXX/400 Programmers Guide V4R1

BASIC REXX Constructs

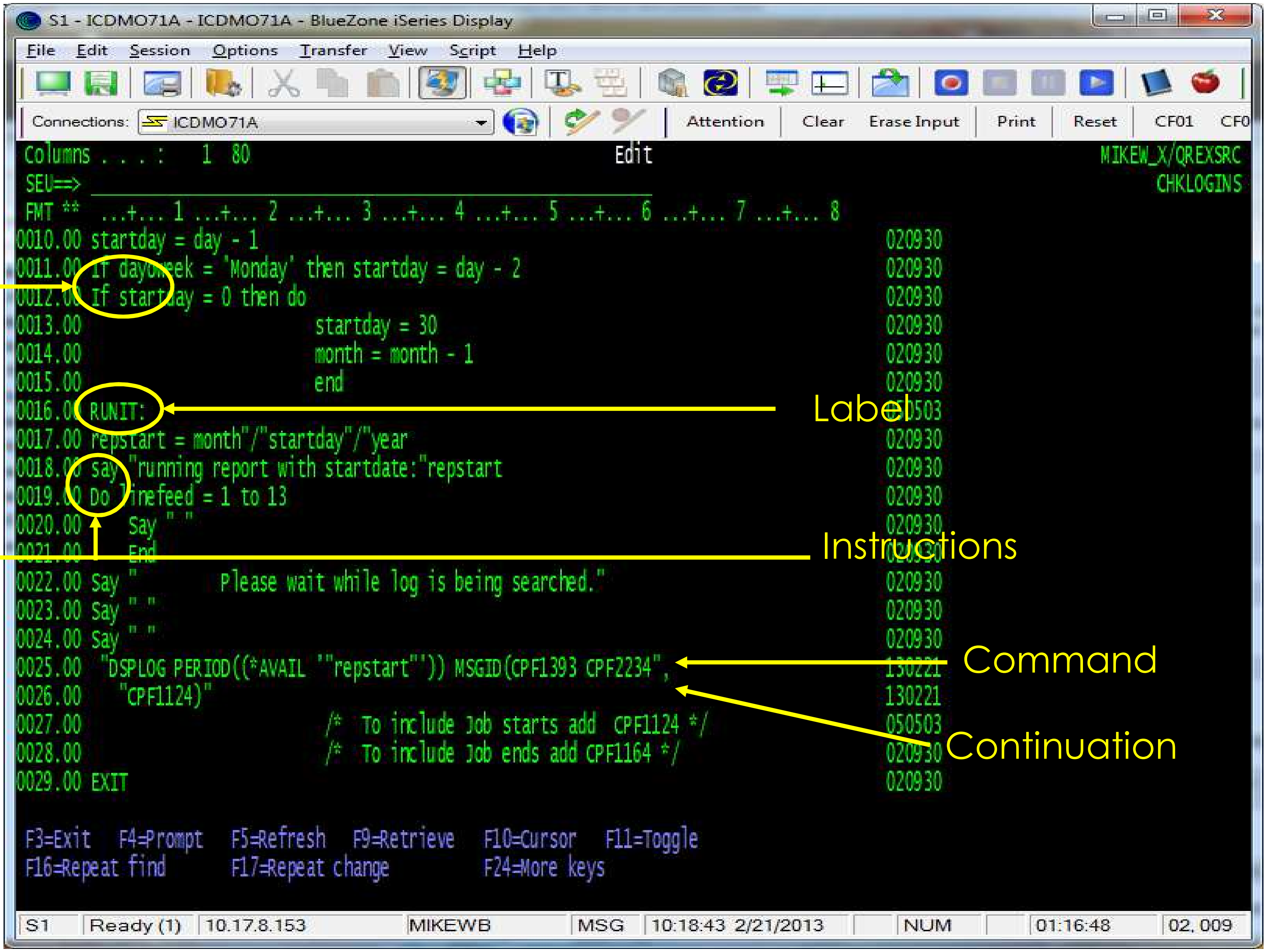
- Source statements are called “clauses” and consist of:
  - Null clauses
  - Assignments
  - Instructions
  - Labels
  - Commands
- Clauses are made up of “tokens”
  - Character strings delimited by blanks
  - Scanned left to right
    - Instructions recognized
    - Comments removed
    - Multiple blanks converted to single blanks

Examples of constructs



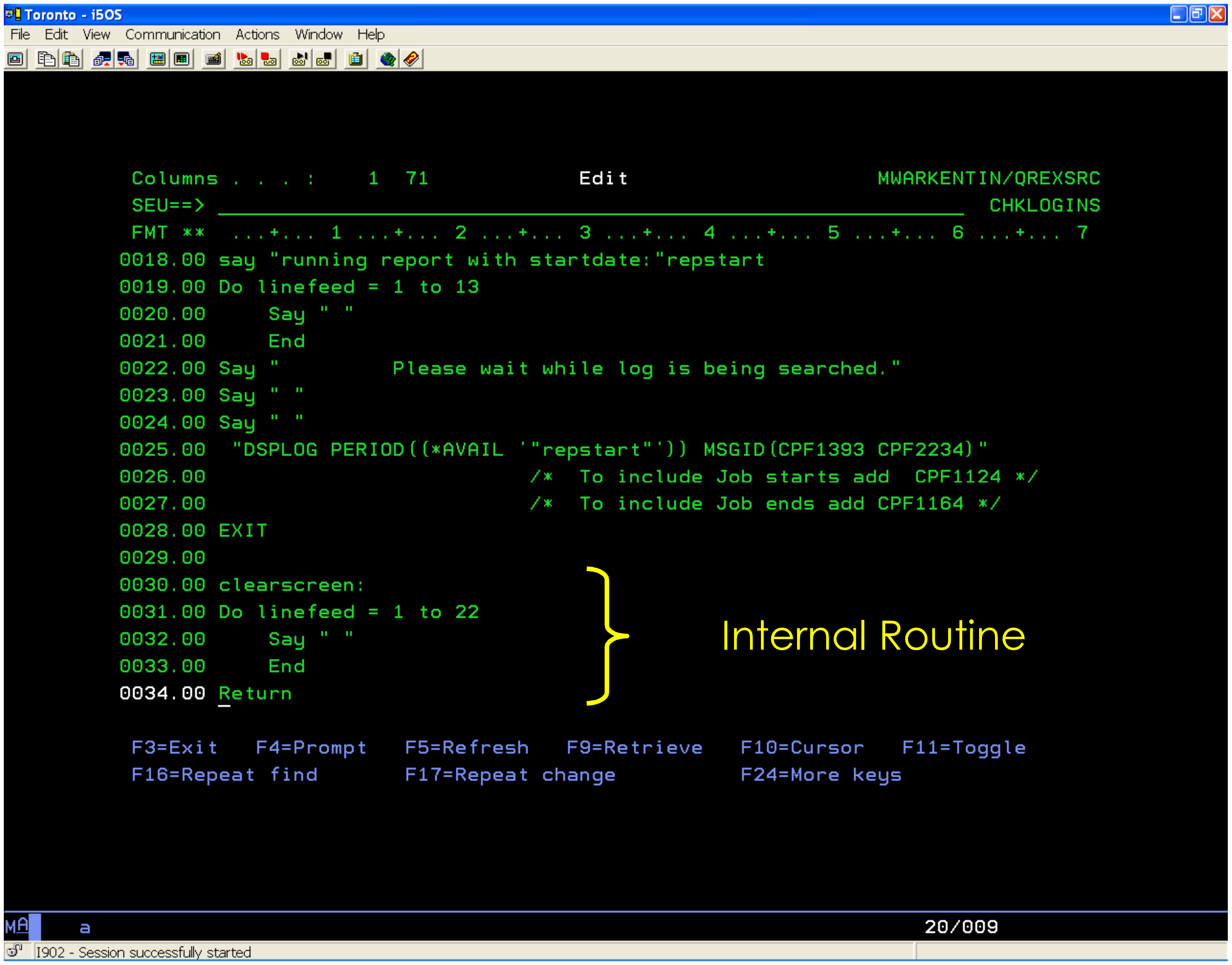
```
Columns . . . :    1  71           Edit                  MWARKENTIN/QREXSRC
SEU==> _______________________________________________________ CHKLOGINS
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
       ************** Beginning of data *************************************
0001.00  /* Check the log for Invalid Login attempts */
0002.00  call clearscreen
0003.00
0004.00 /* Trace ?R */
0005.00 today =DATE('U')
0006.00 dayoweek = DATE('W')
0007.00 month = SUBSTR(today,1,2)
0008.00 day = SUBSTR(today,4,2)
0009.00 year  = SUBSTR(today,7,2)
0010.00 startday = day - 1
0011.00 If dayoweek = 'Monday' then startday = day - 2
0012.00 If startday = 0 then do
0013.00                          startday = 30
0014.00                          month = month - 1
0015.00                          end
0016.00 repstart = month"/"startday"/"year

 F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
 F16=Repeat find       F17=Repeat change          F24=More keys

MA      a                                                       02/009
I902 - Session successfully started
```

Examples of constructs


S1 - ICDMO71A - ICDMO71A - BlueZone iSeries Display
File Edit Session Options Transfer View Script Help
Connections: ICDMO71A
Attention Clear Erase Input Print Reset CF01 CF0
Label
Instructions
Command
Continuation
S1 Ready (1) 10.17.8.153 MIKEWB MSG 10:18:43 2/21/2013 NUM 01:16:48 02, 009

```
Columns . . . :    1  80                    Edit                    MIKEW_X/QREXSRC
SEU==> ____________________________________________________                CHKLOGINS
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
startday = day - 1                                                     020930
if dayofweek = 'Monday' then startday = day - 2                        020930
If startday = 0 then do                                                020930
                        startday = 30                                  020930
                        month = month - 1                              020930
                        end                                            020930
RUNIT:                                                                 050503
repstart = month"/"startday"/"year                                     020930
say "running report with startdate:"repstart                           020930
Do linefeed = 1 to 13                                                  020930
    Say " "                                                            020930
    End                                                                020930
Say "       Please wait while log is being searched."                  020930
Say " "                                                                020930
Say " "                                                                020930
 "DSPLOG PERIOD((*AVAIL '"repstart"')) MSGID(CPF1393 CPF2234",         130221
   "CPF1124)"                                                          130221
                        /*  To include Job starts add  CPF1124 */      050503
                        /*  To include Job ends add CPF1164 */         020930
EXIT                                                                   020930

F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
F16=Repeat find       F17=Repeat change          F24=More keys
```

Examples of constructs



```
 Columns . . . :     1  71            Edit                   MWARKENTIN/QREXSRC
 SEU==> ____________________________________________________________ CHKLOGINS
 FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0018.00 say "running report with startdate:"repstart
0019.00 Do linefeed = 1 to 13
0020.00     Say " "
0021.00     End
0022.00 Say "          Please wait while log is being searched."
0023.00 Say " "
0024.00 Say " "
0025.00  "DSPLOG PERIOD((*AVAIL '"repstart"')) MSGID(CPF1393 CPF2234)"
0026.00                         /*  To include Job starts add  CPF1124 */
0027.00                         /*  To include Job ends add CPF1164 */
0028.00 EXIT
0029.00
0030.00 clearscreen:
0031.00 Do linefeed = 1 to 22
0032.00     Say " "
0033.00     End
0034.00 Return

 F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
 F16=Repeat find       F17=Repeat change          F24=More keys
```

Internal Routine

```
MA     a                                                       20/009
I902 - Session successfully started
```

Chklogins pgm

```
/* Check the log for Invalid Login attempts */
Call clearscreen          ←         Subroutine

/* Trace ?R */            ←         Trace function
today =DATE('U')
dayoweek = DATE('W')
month = SUBSTR(today,1,2)
day = SUBSTR(today,4,2)
year  = SUBSTR(today,7,2)
startday = day - 1
If dayoweek = 'Monday' then startday = day - 2
If startday = 0 then do
              startday = 30
              month = month - 1
              end
```

Chklogins pgm

```
RUNIT:
repstart = month"/"startday"/"year
say "running report with startdate:"repstart
Do linefeed = 1 to 13
    Say " "
    End
Say "        Please wait while log is being searched."
Say " "
Say " "
 "DSPLOG PERIOD((*AVAIL "repstart")) MSGID(CPF1393 CPF2234)"
                    /*  To include Job starts add  CPF1124 */
                    /*  To include Job ends add CPF1164 */
EXIT
```

Special Function to force CR

Chklogins pgm

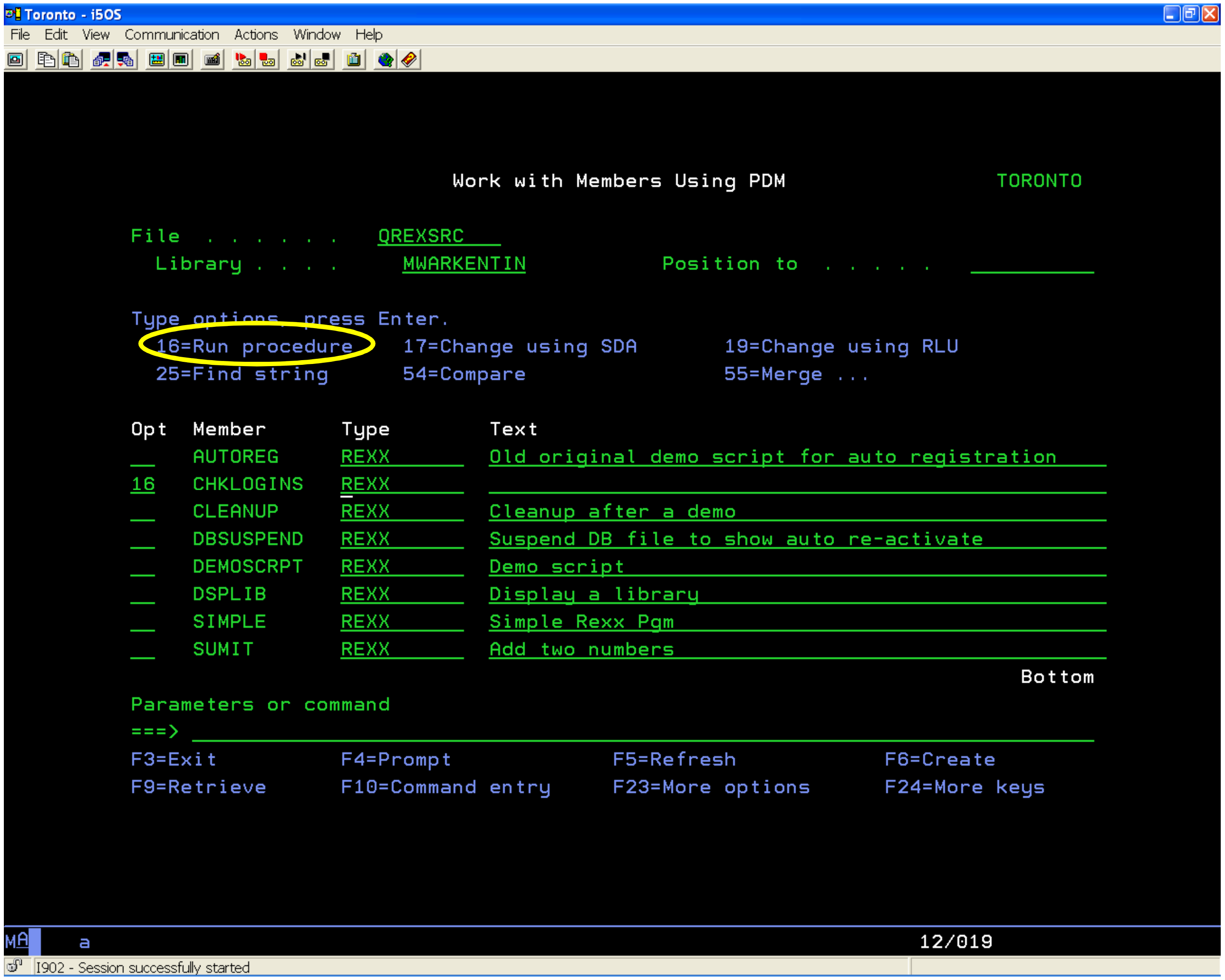
```
clearscreen:
Do linefeed = 1 to 22
    Say " "
    End
Return
```

$\}$ Subroutine defined

Creating and Running REXX

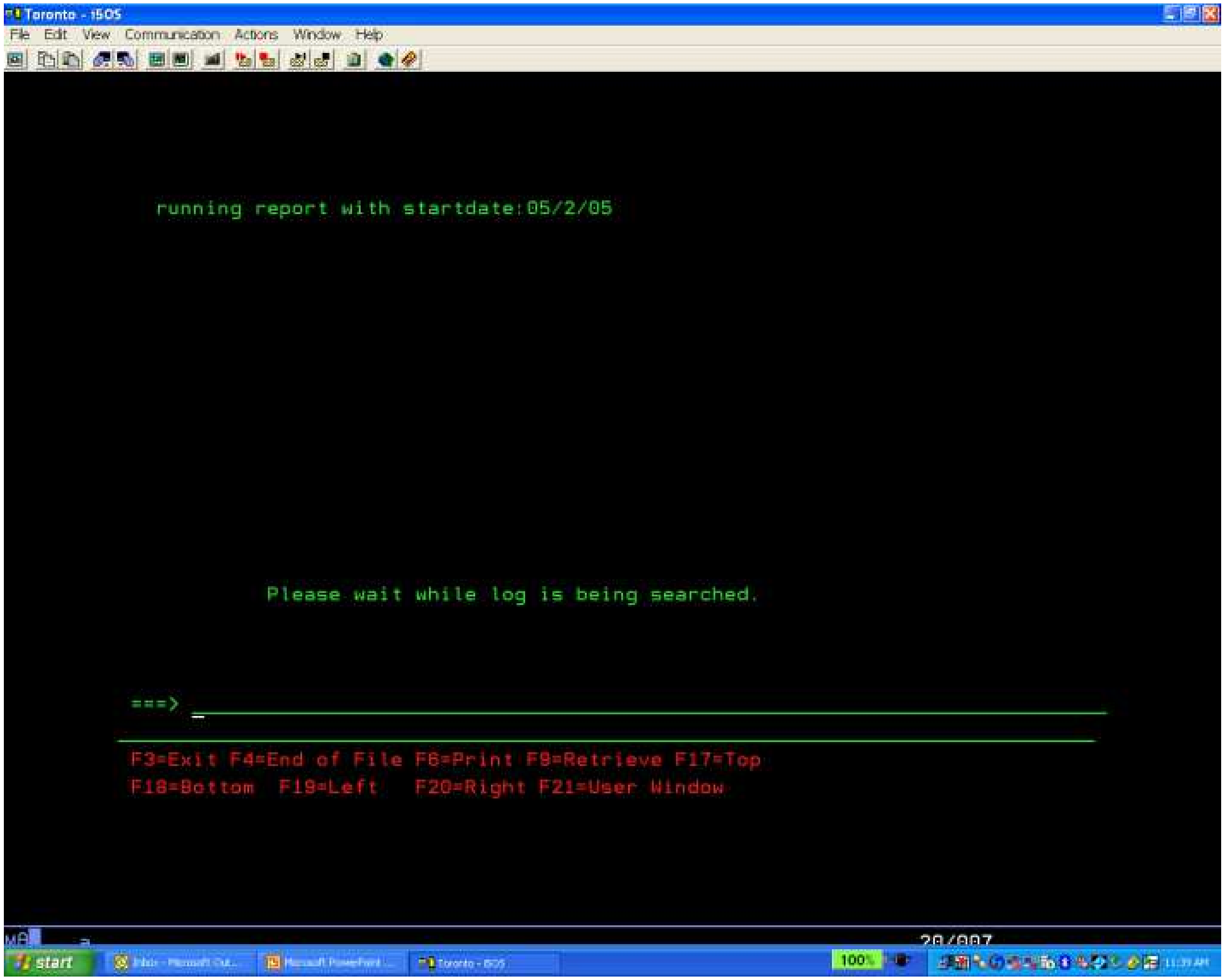
- Create as source physical files (QREXSRC in QGPL)
- Use SEU or WDSc
- Source type can be REXX but not required
- Not program objects (no compiles)
- To run use…
  - STRREXPRC command – pass parms
  - Option 16 in WRKMBRPDM
  - Call QREXX API

Let's Run It



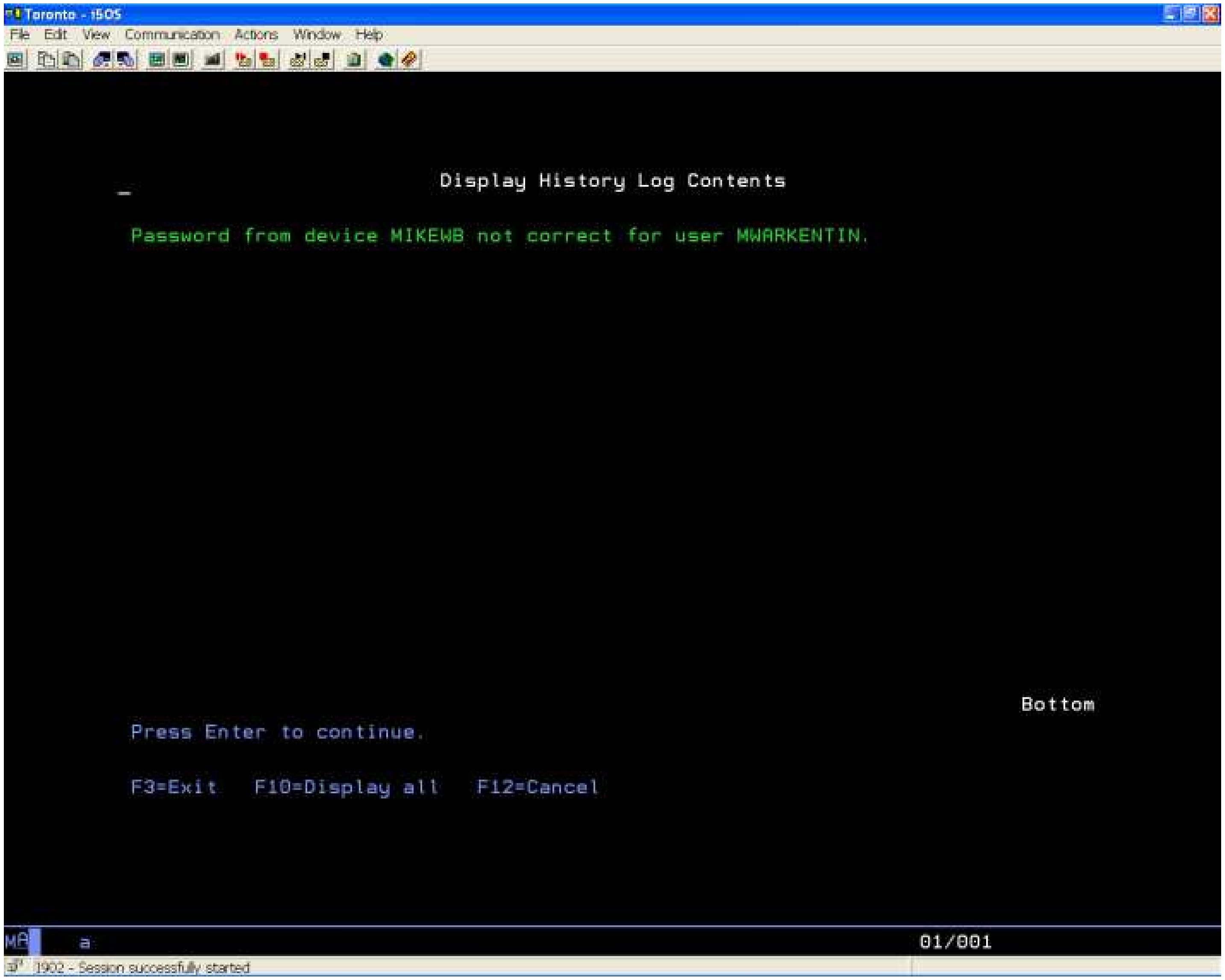
```
                              Work with Members Using PDM                    TORONTO

 File  . . . . . .   QREXSRC
   Library . . . .     MWARKENTIN            Position to  . . . . .

 Type options, press Enter.
   16=Run procedure        17=Change using SDA         19=Change using RLU
   25=Find string          54=Compare                  55=Merge ...

 Opt  Member      Type        Text
 __   AUTOREG     REXX        Old original demo script for auto registration
 16   CHKLOGINS   REXX
 __   CLEANUP     REXX        Cleanup after a demo
 __   DBSUSPEND   REXX        Suspend DB file to show auto re-activate
 __   DEMOSCRPT   REXX        Demo script
 __   DSPLIB      REXX        Display a library
 __   SIMPLE      REXX        Simple Rexx Pgm
 __   SUMIT       REXX        Add two numbers
                                                                          Bottom
 Parameters or command
 ===>
 F3=Exit          F4=Prompt              F5=Refresh          F6=Create
 F9=Retrieve      F10=Command entry      F23=More options    F24=More keys

MA     a                                                  12/019
I902 - Session successfully started
```

The result…



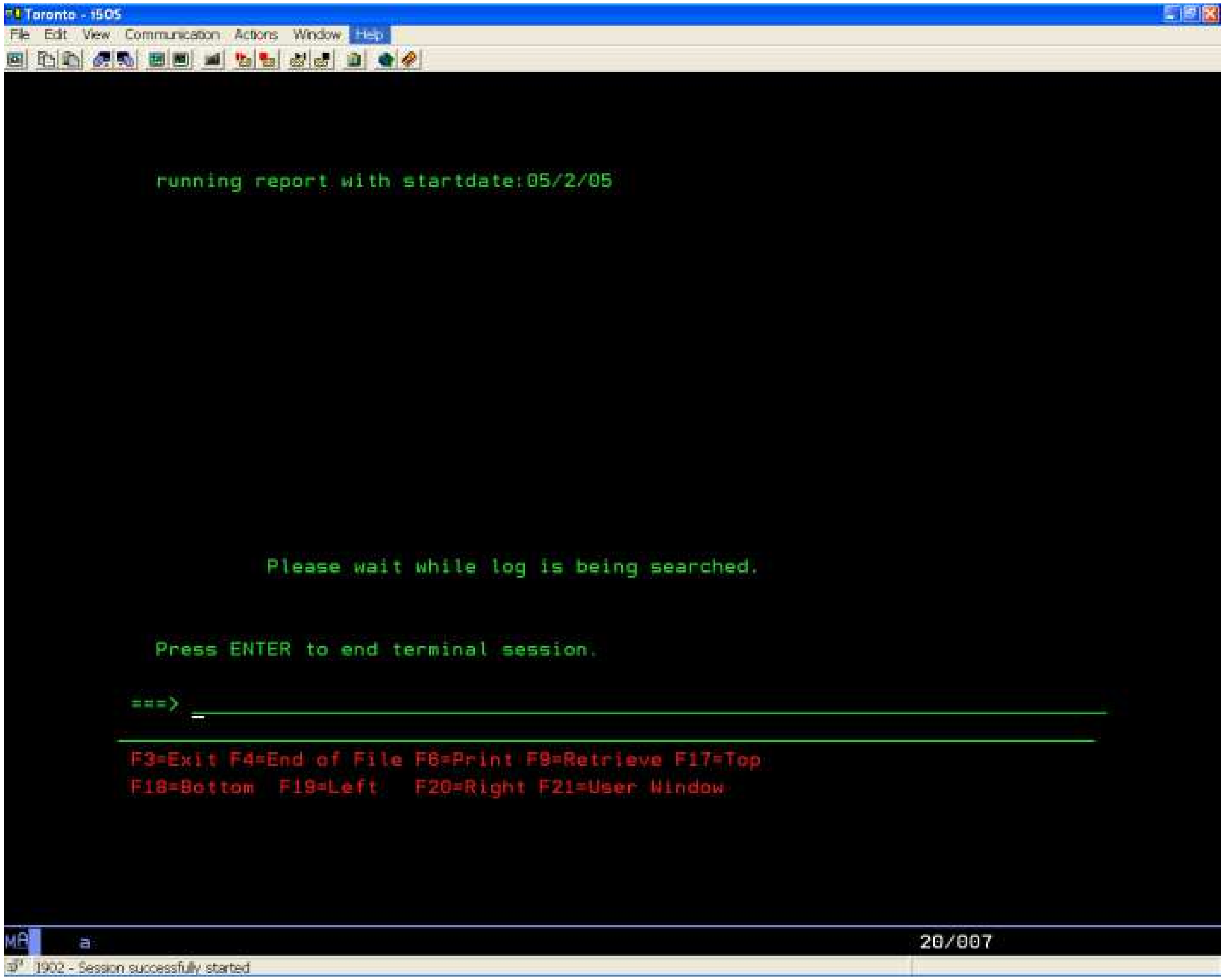
```
     running report with startdate:05/2/05




              Please wait while log is being searched.



  ===> _
 ______________________________________________________________
  F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
  F18=Bottom  F19=Left   F20=Right F21=User Window
```

The result…



```
                    Display History Log Contents

Password from device MIKEWB not correct for user MWARKENTIN.


                                                          Bottom
Press Enter to continue.

F3=Exit   F10=Display all   F12=Cancel

MA     a                                          01/001
1902 - Session successfully started
```

The result…



```
running report with startdate:05/2/05




        Please wait while log is being searched.


Press ENTER to end terminal session.

===> _______________________________________________________________

F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window

MA    a                                                      20/007
1902 - Session successfully started
```

REXX Input & Output

- Three files for input and output:

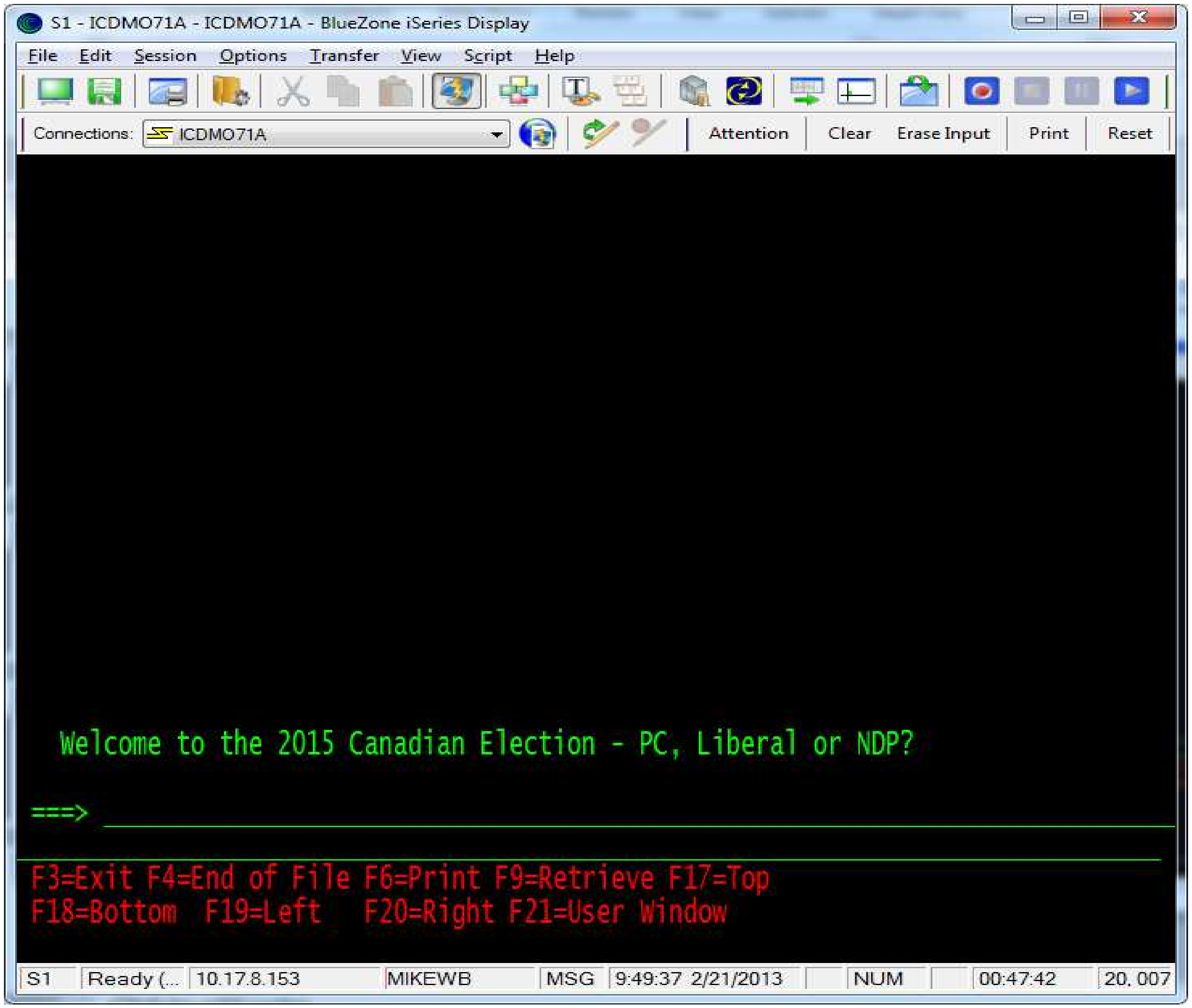
File	Used By	Default in Interactive	Default in Batch
STDIN	PULL	Keyboard	QINLINE
STDOUT	SAY	Display	QPRINT
STDERR	TRACE	Display	QPRINT

A simple example using PULL & SAY…

```
/* Canadian Election 2015*/
call clearscreen

SAY "Welcome to the 2015 Canadian Election – PC, Liberal or NDP?"
PULL who
IF who \= "PC" THEN
      DO UNTIL who = "PC"
      SAY "Thank you for voting Conservative!"
      PULL who
      END
SAY "You have successfully voted for Harper!"
EXIT

clearscreen:
Do linefeed = 1 to 22
    Say " "
 End
Return
```

← Must be UC

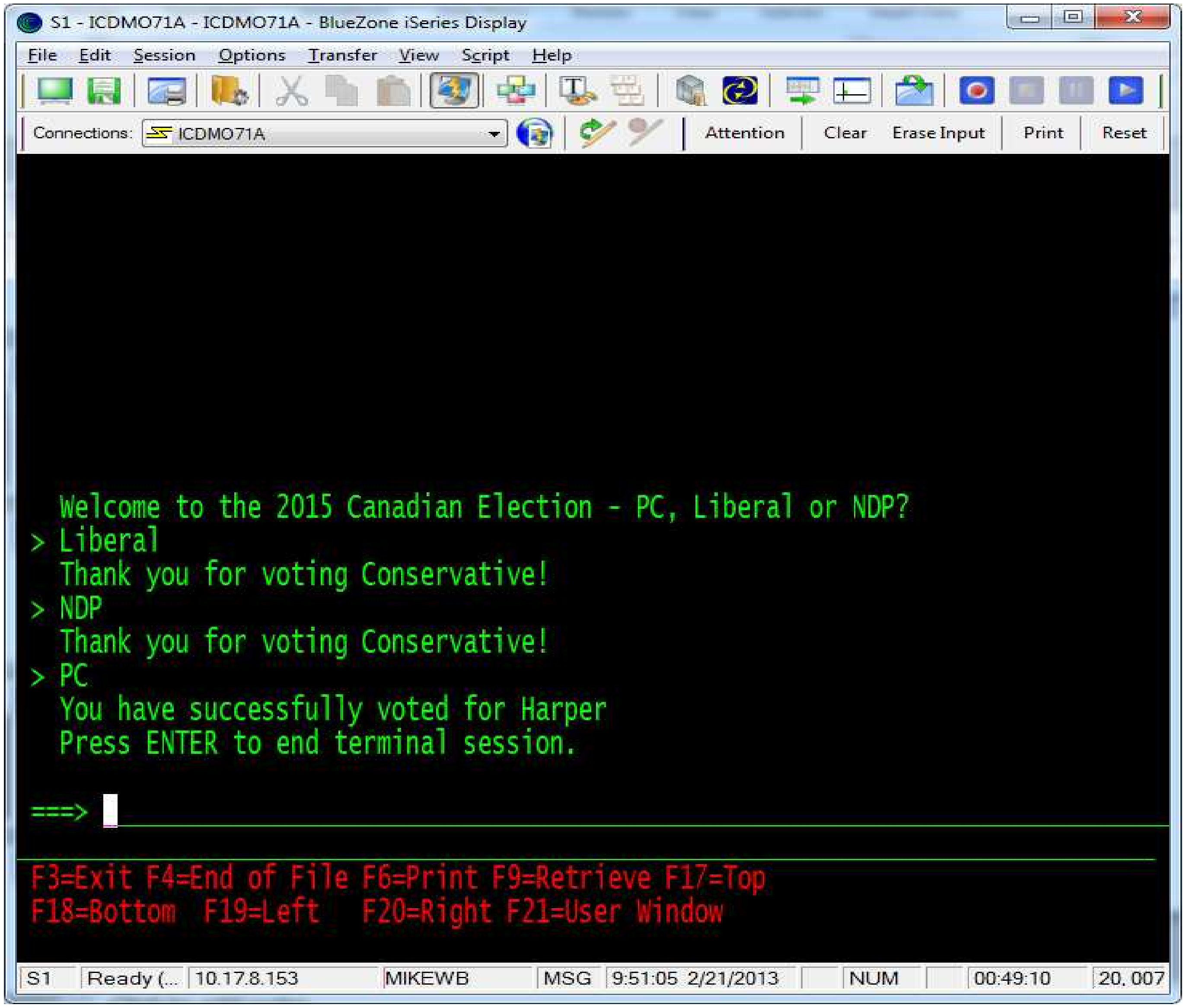
When it is run… PULL waits for a response from STDIN


S1 - ICDMO71A - ICDMO71A - BlueZone iSeries Display
File Edit Session Options Transfer View Script Help
Connections: ICDMO71A
Attention Clear Erase Input Print Reset

```
  Welcome to the 2015 Canadian Election - PC, Liberal or NDP?

===> ________________________________________________________

F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window
```

S1 Ready (... 10.17.8.153 MIKEWB MSG 9:49:37 2/21/2013 NUM 00:47:42 20, 007

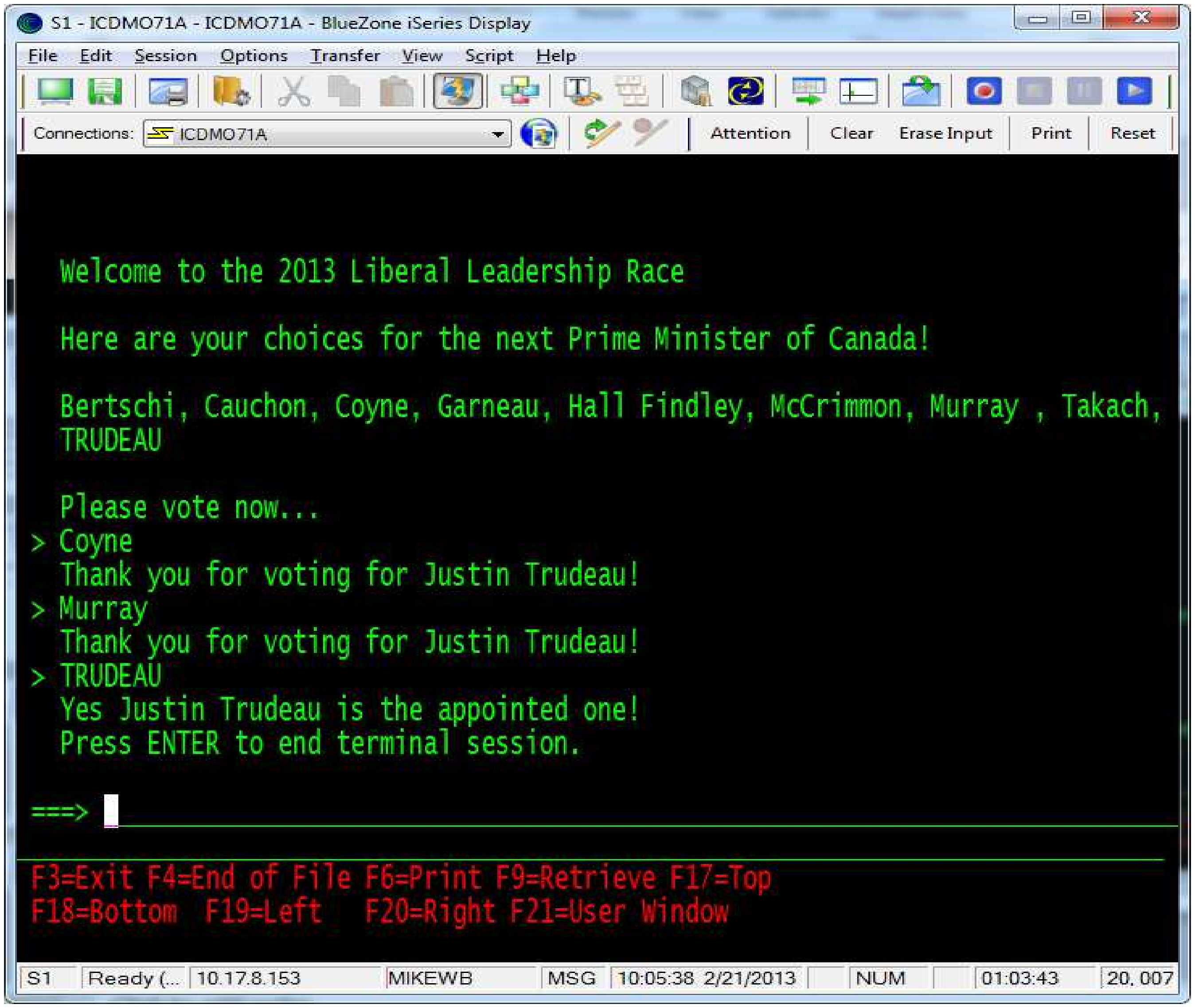
When the correct response is given – the program ends


S1 - ICDMO71A - ICDMO71A - BlueZone iSeries Display
File Edit Session Options Transfer View Script Help
Connections: ICDMO71A
Attention Clear Erase Input Print Reset

```
  Welcome to the 2015 Canadian Election - PC, Liberal or NDP?
> Liberal
  Thank you for voting Conservative!
> NDP
  Thank you for voting Conservative!
> PC
  You have successfully voted for Harper
  Press ENTER to end terminal session.

===>

F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window
```

```
S1 | Ready (... | 10.17.8.153 | MIKEWB | MSG | 9:51:05 2/21/2013 | NUM | 00:49:10 | 20, 007
```

There are some who say this is how the Liberal Leadership race will go…


S1 - ICDMO71A - ICDMO71A - BlueZone iSeries Display
File Edit Session Options Transfer View Script Help
Connections: ICDMO71A
Attention Clear Erase Input Print Reset

```
  Welcome to the 2013 Liberal Leadership Race

  Here are your choices for the next Prime Minister of Canada!

  Bertschi, Cauchon, Coyne, Garneau, Hall Findley, McCrimmon, Murray , Takach,
  TRUDEAU

  Please vote now...
> Coyne
  Thank you for voting for Justin Trudeau!
> Murray
  Thank you for voting for Justin Trudeau!
> TRUDEAU
  Yes Justin Trudeau is the appointed one!
  Press ENTER to end terminal session.

===>

F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window
```

S1 | Ready (... | 10.17.8.153 | MIKEWB | MSG | 10:05:38 2/21/2013 | NUM | 01:03:43 | 20, 007

REXX Variables & Constants

- Actually called “symbols” in REXX
- Up to 250 characters in length
- Beginning with digit (0-9) or . => constant
  - Cannot be used as variables
  - Examples :

    57

    .0095

    3.1e7 (or 31,000,000)
- Beginning with A-Z, a-z, !, ?, _ => variable
  - All treated as uppercase so mikey, MiKeY or MIKEY are all MIKEY.

REXX Variables & Constants

- You don’t declare variables – just assign them
  - Symbol = expression
  - Expression can be a number, string or calculation
  - Examples:
    - total = price + tax
    - total = 0
    - data = “I love my cat”
    - data = substr(“I love my cat”,2,4)
  - NOTE: If not assigned – value is symbol in uppercase!

REXX Variables & Constants

- Can also be pulled from user input (interactively)
  - Say “Give me two names separated by a space, then hit Enter”
  - Pull firstname secondname
- Entered as arguments
  - ARG first second
  - SAY “The total is” first + second
  - To run STRREXPRC SRCFILE(QGPL/QREXSRC) SRCMBR(SUM) PARM(‘1 2’)

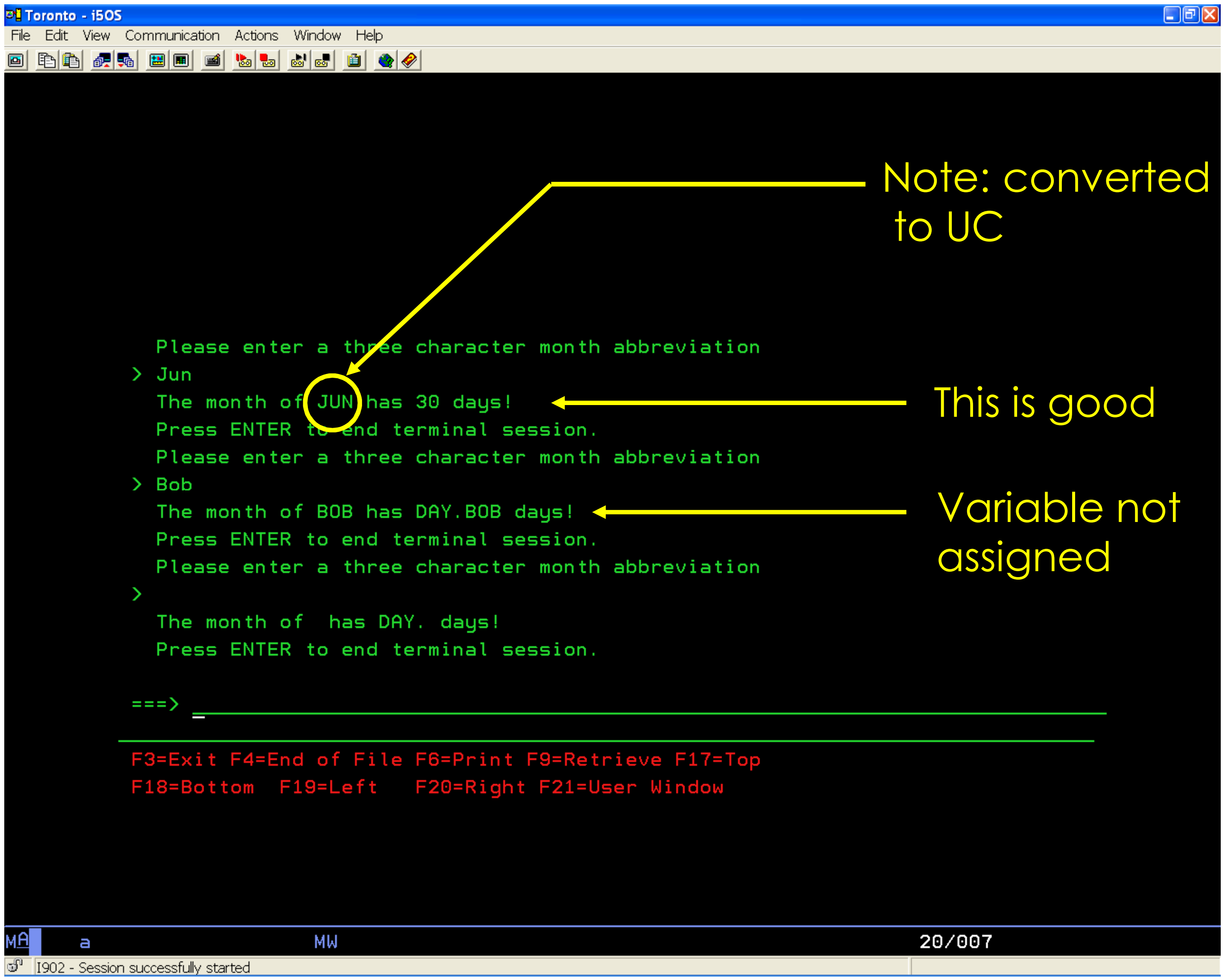
Compound Variables

- A variable containing at least one ‘.’ and one other character following the period
  - Cannot begin with a digit or ‘.’
  - If only one ‘.’ it cannot be the last character
- “Stem” = everything up to first ‘.’
- “Tail” = everything else
- Examples:
  - day.1 = “Sunday”   /* day is the stem, 1 is the tail */
  - Region.branch.office /* region is stem, branch.office is tail */
- Also called compound symbol

Using compound variables for arrays – DAYSINMON pgm

```
/* Get the number of days in the month */

day.jan = 31
day.feb = 28
day.mar = 31
day.apr = 30
day.may = 31
day.jun = 30
day.jul = 31
day.aug = 31
day.sep = 30
day.oct = 31
day.nov = 30
day.dec = 31

Say "Please enter a three character abbreviation for the month"
Pull month
Say "The month of " month "has " day.month " days!"
```

Using compound variables for arrays – running it…



```
  Please enter a three character month abbreviation
> Jun
  The month of JUN has 30 days!
  Press ENTER to end terminal session.
  Please enter a three character month abbreviation
> Bob
  The month of BOB has DAY.BOB days!
  Press ENTER to end terminal session.
  Please enter a three character month abbreviation
>
  The month of  has DAY. days!
  Press ENTER to end terminal session.

===> _

F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window

MA     a              MW                              20/007
I902 - Session successfully started
```

Note: converted to UC

This is good

Variable not assigned

Using compound variables for arrays – Let’s fix it…

```
/* Get the number of days in the month */

Call clearscreen

day.jan = 31
day.feb = 28
day.mar = 31
day.apr = 30
day.may = 31
day.jun = 30
day.jul = 31
day.aug = 31
day.sep = 30
day.oct = 31
day.nov = 30
day.dec = 31

/* Set valid months list */
valid_month_list = 'JAN FEB MAR APR MAY JUN JUL AUG SEP',
                   'OCT NOV DEC'

Say "Please enter a three character month abbreviation"
```

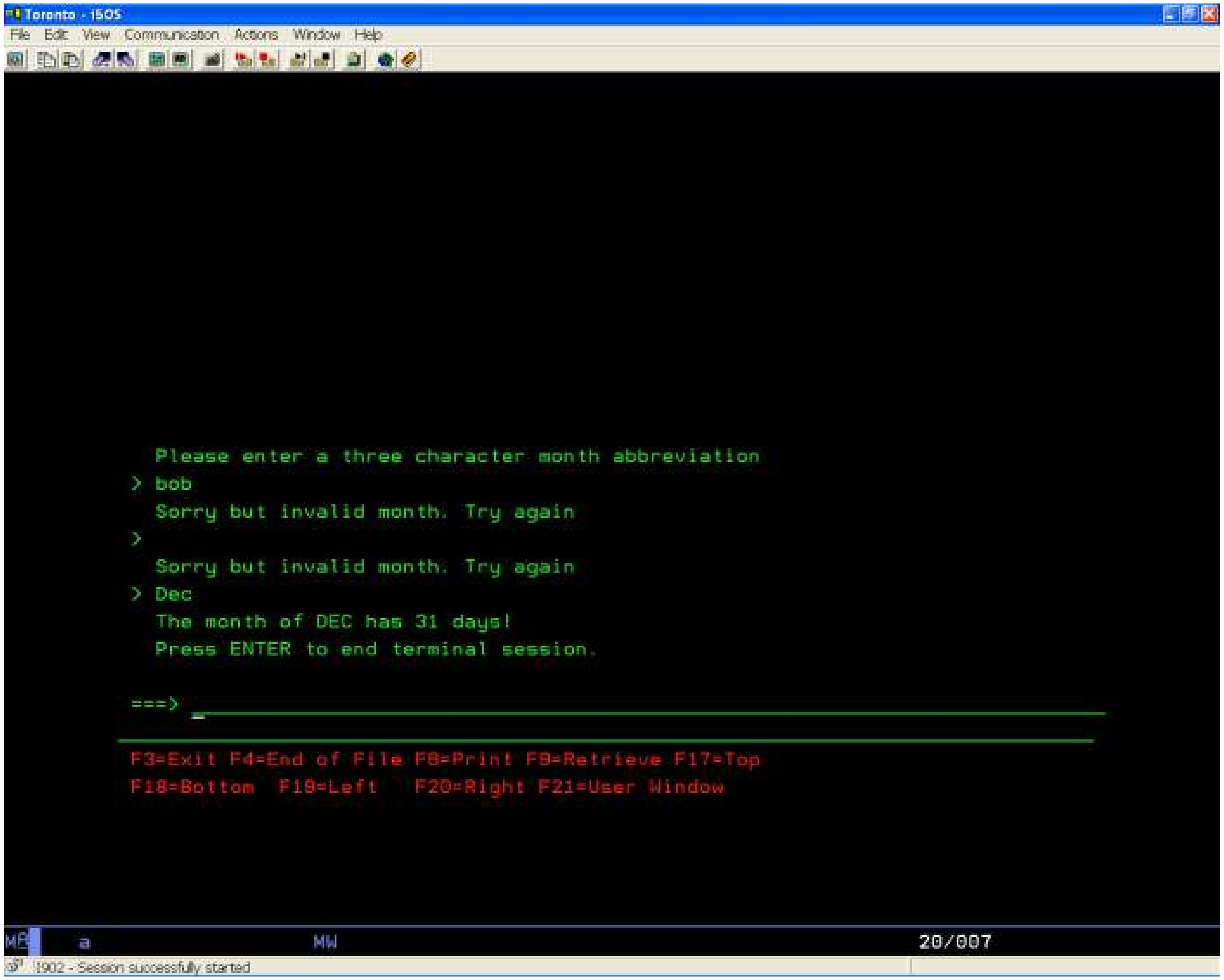
$\}$ Compound Variables

Using compound variables for arrays – Let’s fix it…

```
Parse Upper Pull month
valid = POS(month,valid_month_list)
        do while valid =0
        Say "Sorry but invalid month. Try again"
        Parse Upper Pull month
        valid = POS(month,valid_month_list)
       END
Say "The month of" month "has" day.month "days!"

EXIT
clearscreen:
Do linefeed = 1 to 22
   Say " "
   End
Return
```

Function to check starting position of one string in another

Now let's run it again…



```
  Please enter a three character month abbreviation
> bob
  Sorry but invalid month. Try again
>
  Sorry but invalid month. Try again
> Dec
  The month of DEC has 31 days!
  Press ENTER to end terminal session.

===> _

F3=Exit F4=End of File F8=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window
```

```
MA    a           MW                               20/007
1902 - Session successfully started
```

Functions and Subroutines

- Indicated by clauses called labels (just like CL)
- Can be internal, built-in or external routine
- Returns a single result string
- Subroutines
  - Run when named on a CALL instruction
  - No () required to pass parameters
  - Up to 20 parameters allowed
  - Return value is assigned to variable called "result" and may or may not be passed
  - Uses ARG (special function) to access parms passed to routine or to main REXX pgm

Functions and Subroutines

- Functions
  - No call required
  - Use () to pass parameters
  - Up to 20 parameters allowed
  - Do not touch the “result” variable

Examples of Functions and Subroutines

```
/* Here is a function */

numone = 5

numtwo = 10


Say 'The sum of' numone 'and' numtwo 'is'  Sum(numone, numtwo)

EXIT


Sum:

Total = ARG(1) + ARG(2)

RETURN total
```

The function (returns a single result)

This is the internal routine
ARG is a special built-in function to
Access the parms passed

Examples of Functions and Subroutines

```
/* Here is a subroutine */

numone = 5

numtwo = 10


CALL Sum numone, numtwo

Say 'The sum of' numone 'and' numtwo 'is' result

EXIT


Sum:

Total = ARG(1) + ARG(2)

RETURN total
```

The subroutine (returns a single result assigned to variable "result")

This is the internal routine
ARG is a special built-in function to
Access the parms passed

REXX Built-in Functions

- ABBREV (Abbreviation)
- ABS (Absolute Value)
- ADDRESS
- ARG
- BITAND (Bit by Bit AND)
- BITOR (Bit by Bit OR)
- BITXOR (Bit by Bit Exclusive OR)
- B2X (Binary to Hexadecimal)
- CENTER/CENTRE
- COMPARE
- CONDITION
- COPIES
- C2D (Character to Decimal)
- C2X (Character to Hexidecimal)
- DATATYPE
- DATE
- DBCS
- DELSTR (Delete String)
- DELWORD (Delete Word)
- DIGITS
- D2C (Decimal to Character)
- D2H (Decimal to Hexadecimal)
- ERRORTEXT
- FORM
- FORMAT
- FUZZ
- INSERT
- LASTPOS (Last Position)
- LEFT
- LENGTH

REXX Built-in Functions

- MAX (Maximum)
- MIN (Minimum)
- OVERLAY
- POS (Position)
- QUEUED
- RANDOM
- REVERSE
- RIGHT
- SETMSGRC (Set Message Return Code)
- SIGN
- SOURCELINE
- SPACE
- STRIP
- SUBSTR (Substring)
- SUBWORD
- SYMBOL
- TIME
- TRACE
- TRANSLATE
- TRUNC (Truncate)
- VALUE
- VERIFY
- WORD
- WORDINDEX
- WORDLENGTH
- WORDPOS (Word Position)
- WORDS
- XRANGE
- X2B (Hexidecimal to Binary)
- X2C (Hexidecimal to Character)
- X2D (Hexidecimal to Decimal)

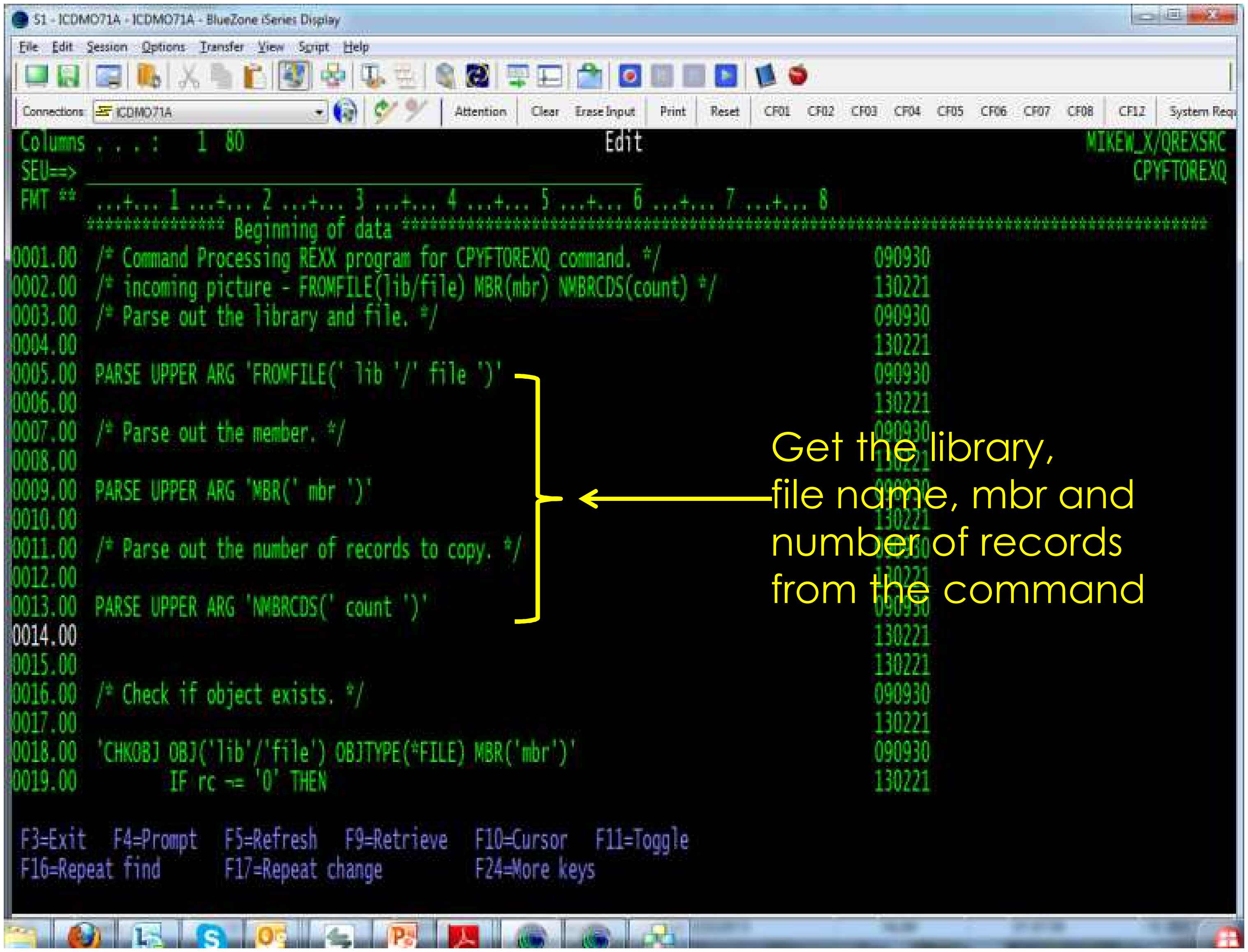
File Handling – The External Data Queue

- Used to temporarily hold data
- Data available as:
  - Lines (character string of variable length up to 32,767 chars)
  - Buffers (subgrouping of lines is a queue)
- Exists when job is started and persists until job ends
- All programs running under same job have access to the queue

What can you do with a queue?

- Place line at front of current buffer (QUEUE)
- Place a line at end of current buffer (PUSH)
- Retrieve a line from front of queue (PULL)
- Determine number of lines in a queue (QUEUED)
- Create, remove a queue buffer (ADDREXBUF or RMVREXBUF)

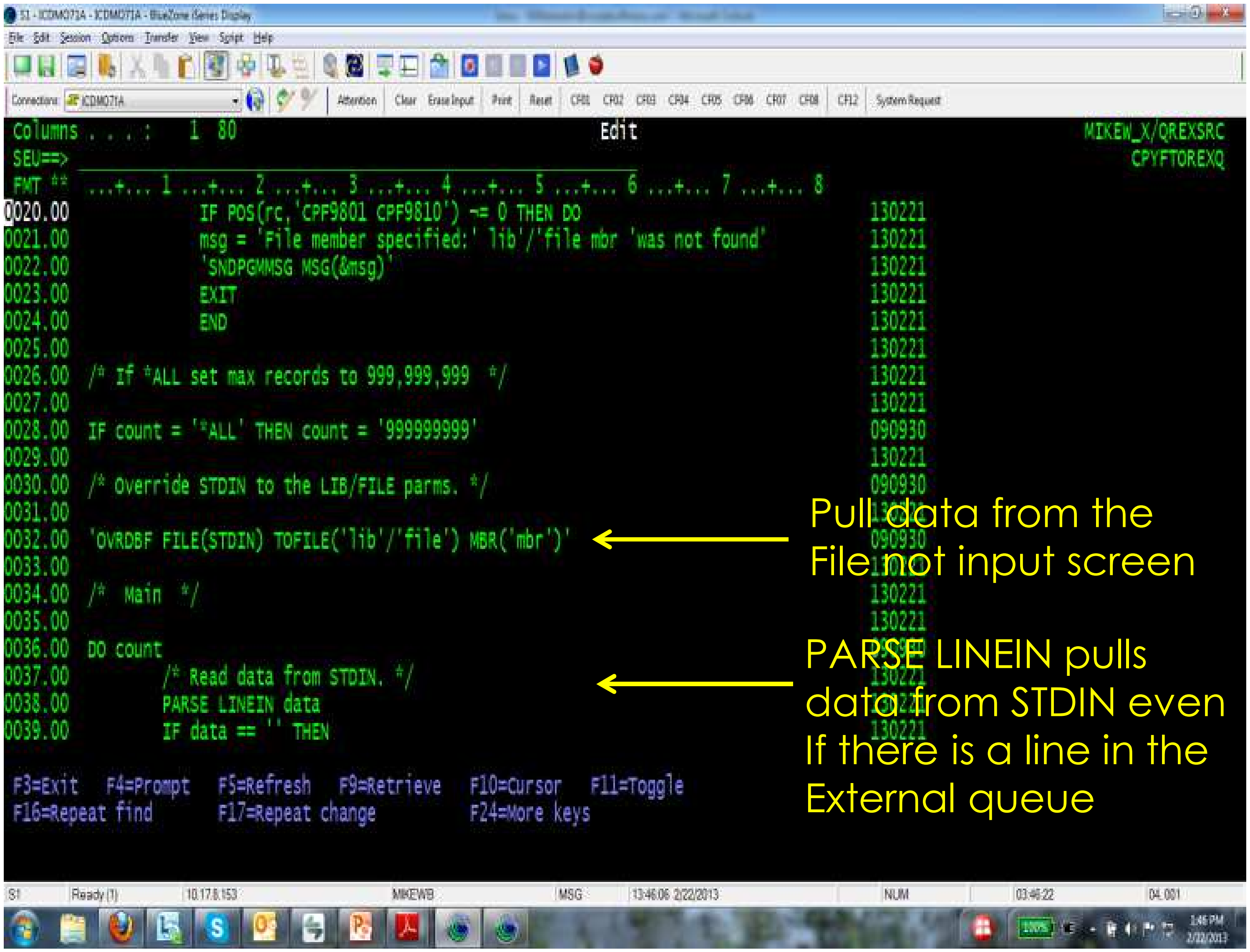
Example


S1 - ICDMO71A - ICDMO71A - BlueZone iSeries Display
File Edit Session Options Transfer View Script Help
Connections: ICDMO71A
Attention Clear Erase Input Print Reset CF01 CF02 CF03 CF04 CF05 CF06 CF07 CF08 CF12 System Req

```
Columns . . . :    1  80                     Edit                       MIKEW_X/QREXSRC
SEU==>                                                                         CPYFTOREXQ
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
       *************** Beginning of data *************************************
0001.00 /* Command Processing REXX program for CPYFTOREXQ command. */          090930
0002.00 /* incoming picture - FROMFILE(lib/file) MBR(mbr) NMBRCDS(count) */    130221
0003.00 /* Parse out the library and file. */                                  090930
0004.00                                                                        130221
0005.00 PARSE UPPER ARG 'FROMFILE(' lib '/' file ')'                           090930
0006.00                                                                        130221
0007.00 /* Parse out the member. */                                            090930
0008.00                                                                        130221
0009.00 PARSE UPPER ARG 'MBR(' mbr ')'                                         090930
0010.00                                                                        130221
0011.00 /* Parse out the number of records to copy. */                         090930
0012.00                                                                        130221
0013.00 PARSE UPPER ARG 'NMBRCDS(' count ')'                                   090930
0014.00                                                                        130221
0015.00                                                                        130221
0016.00 /* Check if object exists. */                                          090930
0017.00                                                                        130221
0018.00 'CHKOBJ OBJ('lib'/'file') OBJTYPE(*FILE) MBR('mbr')'                   090930
0019.00          IF rc ¬= '0' THEN                                             130221

 F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
 F16=Repeat find       F17=Repeat change          F24=More keys
```

Get the library, file name, mbr and number of records from the command

Example

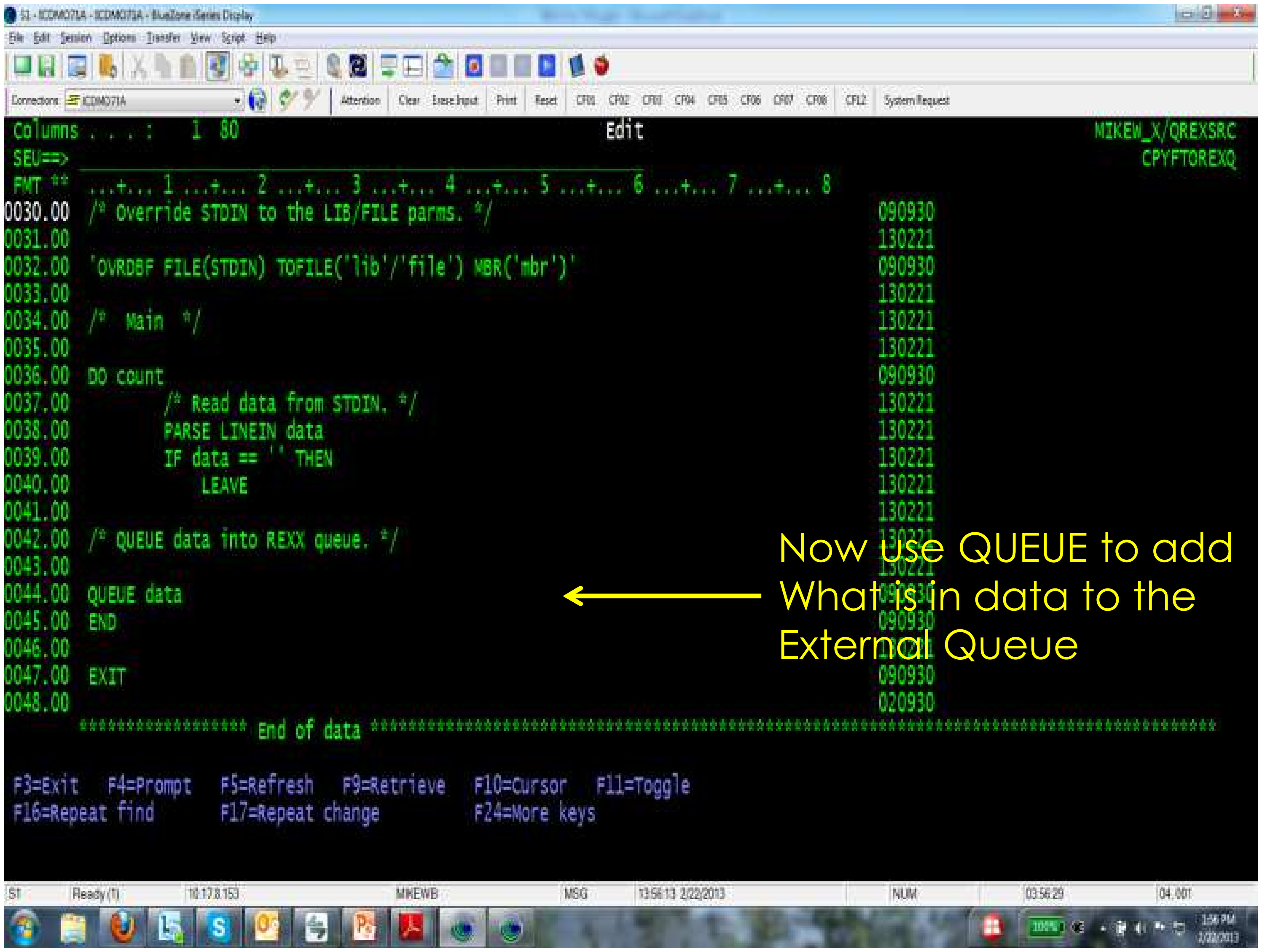


```
Columns . . . :    1  80                         Edit                    MIKEW_X/QREXSRC
SEU==>                                                                         CPYFTOREXQ
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
0020.00           IF POS(rc,'CPF9801 CPF9810') ¬= 0 THEN DO                      130221
0021.00           msg = 'File member specified:' lib'/'file mbr 'was not found'  130221
0022.00           'SNDPGMMSG MSG(&msg)'                                          130221
0023.00           EXIT                                                           130221
0024.00           END                                                            130221
0025.00                                                                          130221
0026.00  /* If *ALL set max records to 999,999,999  */                           130221
0027.00                                                                          130221
0028.00  IF count = '*ALL' THEN count = '999999999'                               090930
0029.00                                                                          130221
0030.00  /* Override STDIN to the LIB/FILE parms. */                             090930
0031.00                                                                          130221
0032.00  'OVRDBF FILE(STDIN) TOFILE('lib'/'file') MBR('mbr')'                     090930
0033.00                                                                          130221
0034.00  /*  Main  */                                                            130221
0035.00                                                                          130221
0036.00  DO count                                                                130221
0037.00         /* Read data from STDIN. */                                      130221
0038.00         PARSE LINEIN data                                                130221
0039.00         IF data == '' THEN                                               130221

F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
F16=Repeat find       F17=Repeat change          F24=More keys
```

Pull data from the File not input screen

PARSE LINEIN pulls data from STDIN even If there is a line in the External queue

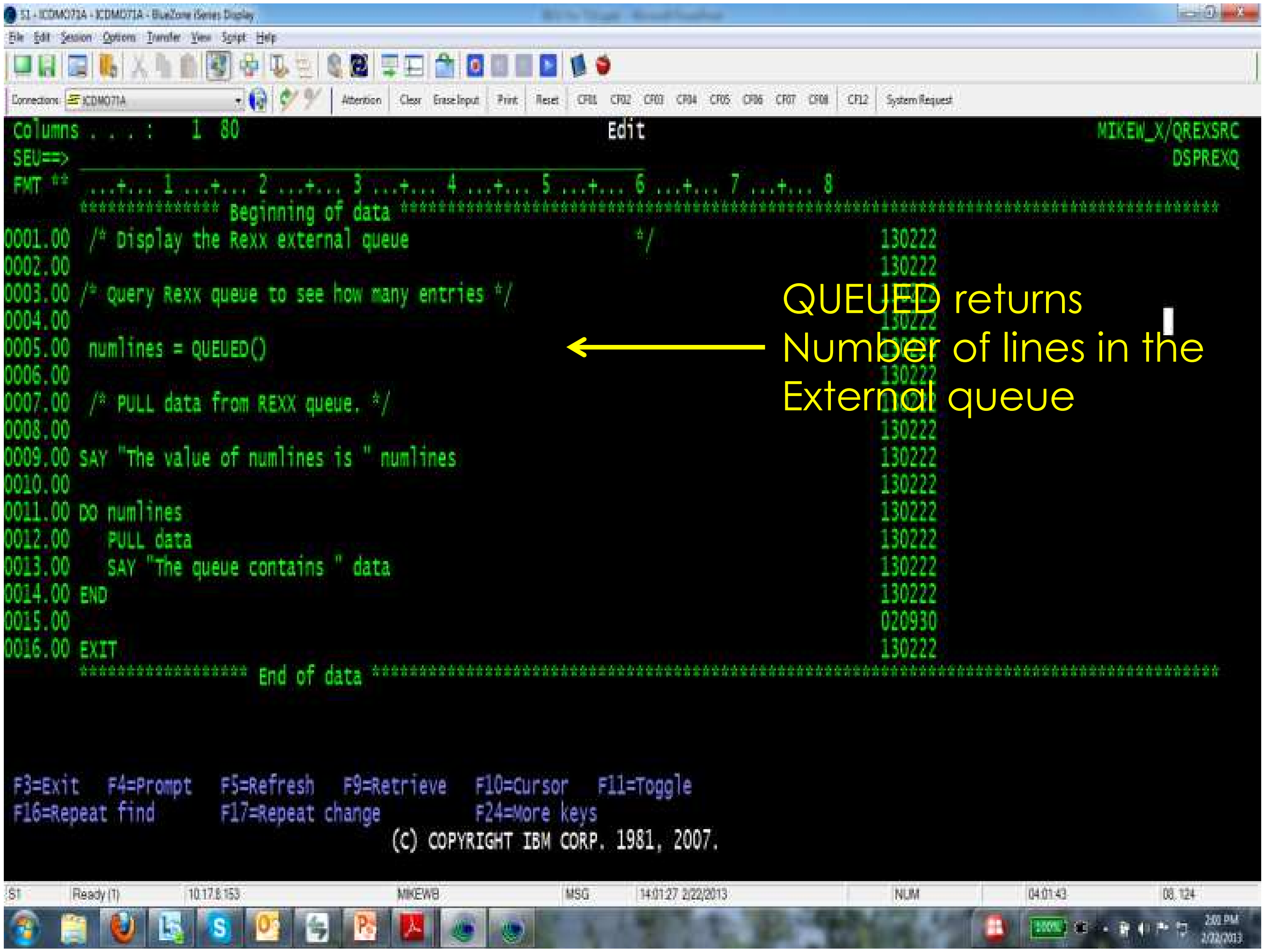
Example



```
Columns . . . :    1  80                          Edit                    MIKEW_X/QREXSRC
SEU==>                                                                          CPYFTOREXQ
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
0030.00 /* Override STDIN to the LIB/FILE parms. */                      090930
0031.00                                                                  130221
0032.00 'OVRDBF FILE(STDIN) TOFILE('lib'/'file') MBR('mbr')'             090930
0033.00                                                                  130221
0034.00 /*  Main  */                                                     130221
0035.00                                                                  130221
0036.00 DO count                                                         090930
0037.00         /* Read data from STDIN. */                              130221
0038.00         PARSE LINEIN data                                        130221
0039.00         IF data == '' THEN                                       130221
0040.00             LEAVE                                                130221
0041.00                                                                  130221
0042.00 /* QUEUE data into REXX queue. */                                130221
0043.00                                                                  130221
0044.00 QUEUE data                                                       090930
0045.00 END                                                              090930
0046.00                                                                  090930
0047.00 EXIT                                                             090930
0048.00                                                                  020930
        ****************** End of data ******************************************

F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
F16=Repeat find       F17=Repeat change          F24=More keys
```

Now use QUEUE to add What is in data to the External Queue

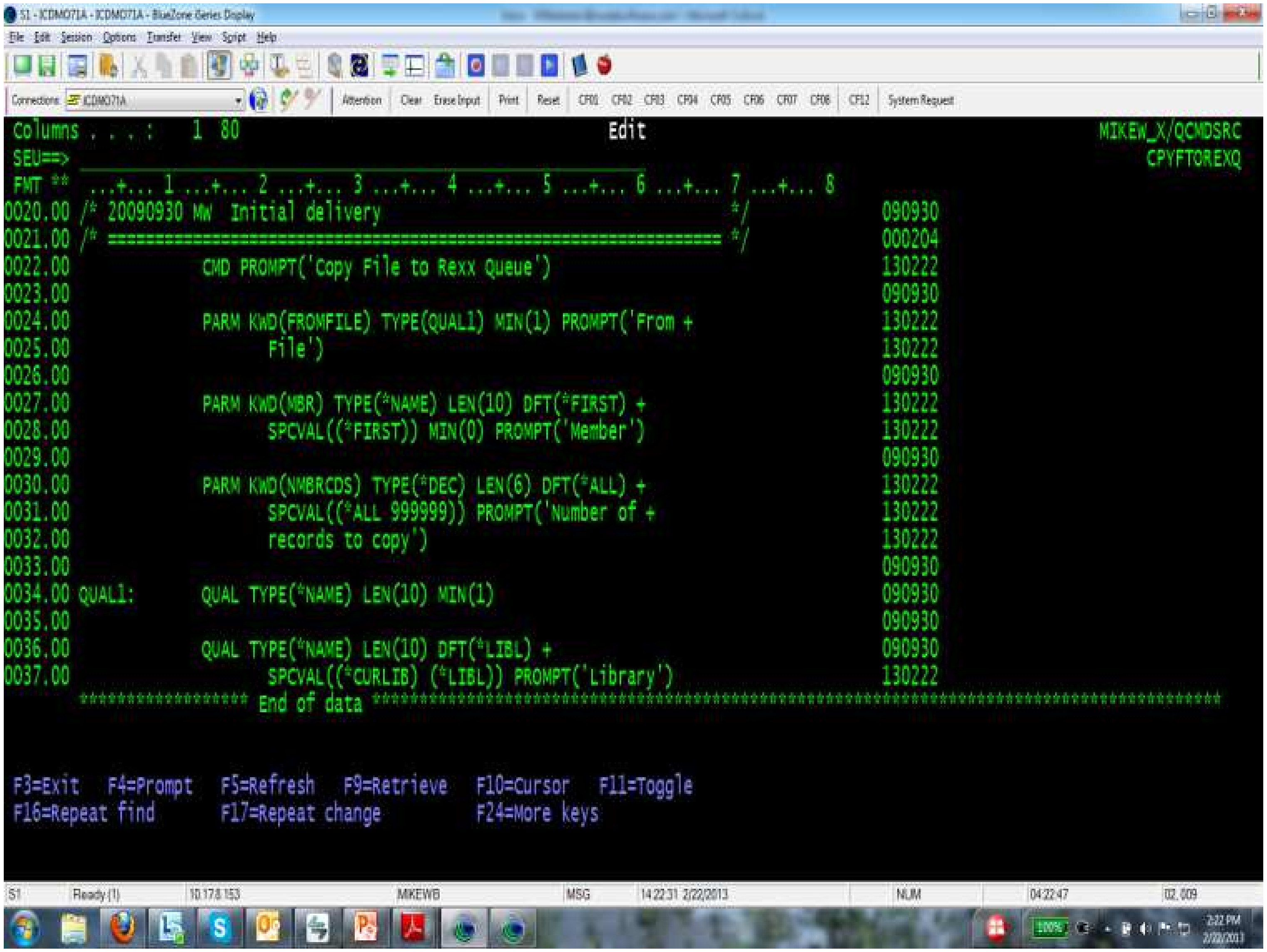
Example



```
Columns . . . :    1  80                                Edit                    MIKEW_X/QREXSRC
SEU==>                                                                                  DSPREXQ
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
        *************** Beginning of data *************************************************
0001.00  /* Display the Rexx external queue                        */              130222
0002.00                                                                            130222
0003.00 /* Query Rexx queue to see how many entries */                             130222
0004.00                                                                            130222
0005.00  numlines = QUEUED()                                                       130222
0006.00                                                                            130222
0007.00  /* PULL data from REXX queue. */                                          130222
0008.00                                                                            130222
0009.00 SAY "The value of numlines is " numlines                                   130222
0010.00                                                                            130222
0011.00 DO numlines                                                                130222
0012.00    PULL data                                                               130222
0013.00    SAY "The queue contains " data                                          130222
0014.00 END                                                                        130222
0015.00                                                                            020930
0016.00 EXIT                                                                       130222
        ****************** End of data ***********************************************

 F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
 F16=Repeat find       F17=Repeat change          F24=More keys
                                     (C) COPYRIGHT IBM CORP. 1981, 2007.
```

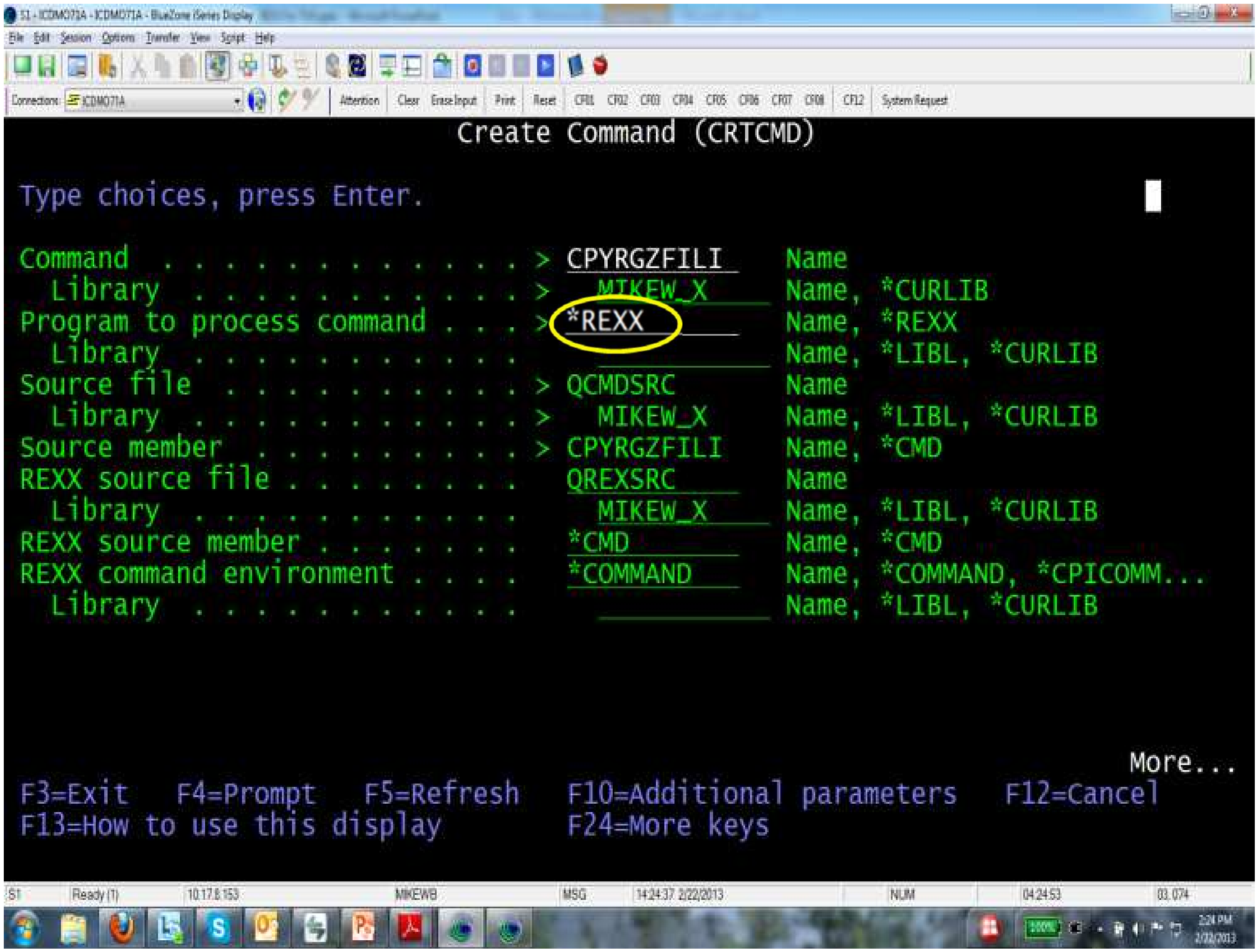
QUEUED returns Number of lines in the External queue

CPYFTOREXQ Command



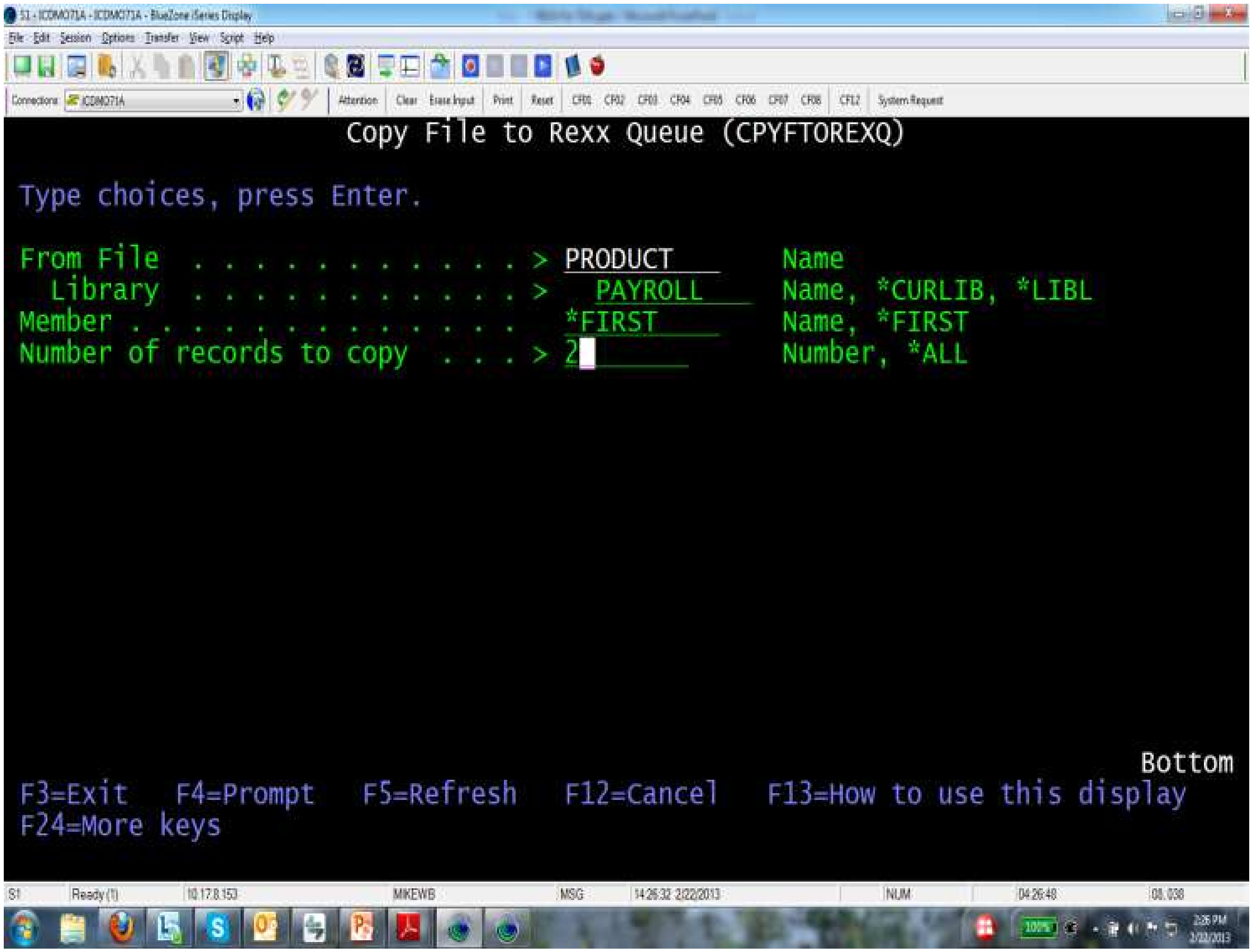
```
 Columns . . . :    1  80                          Edit                  MIKEW_X/QCMDSRC
 SEU==>                                                                       CPYFTOREXQ
 FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
        /* 20090930 MW  Initial delivery                            */      090930
        /* ================================================ */      000204
                    CMD PROMPT('Copy File to Rexx Queue')                   130222
                                                                            090930
                    PARM KWD(FROMFILE) TYPE(QUAL1) MIN(1) PROMPT('From +    130222
                           File')                                           130222
                                                                            090930
                    PARM KWD(MBR) TYPE(*NAME) LEN(10) DFT(*FIRST) +         130222
                           SPCVAL((*FIRST)) MIN(0) PROMPT('Member')         130222
                                                                            090930
                    PARM KWD(NMBRCDS) TYPE(*DEC) LEN(6) DFT(*ALL) +         130222
                           SPCVAL((*ALL 999999)) PROMPT('Number of +        130222
                           records to copy')                                130222
                                                                            090930
        QUAL1:      QUAL TYPE(*NAME) LEN(10) MIN(1)                         090930
                                                                            090930
                    QUAL TYPE(*NAME) LEN(10) DFT(*LIBL) +                   090930
                           SPCVAL((*CURLIB) (*LIBL)) PROMPT('Library')      130222
        ****************** End of data *******************************************


 F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
 F16=Repeat find       F17=Repeat change          F24=More keys
```

Compile it…



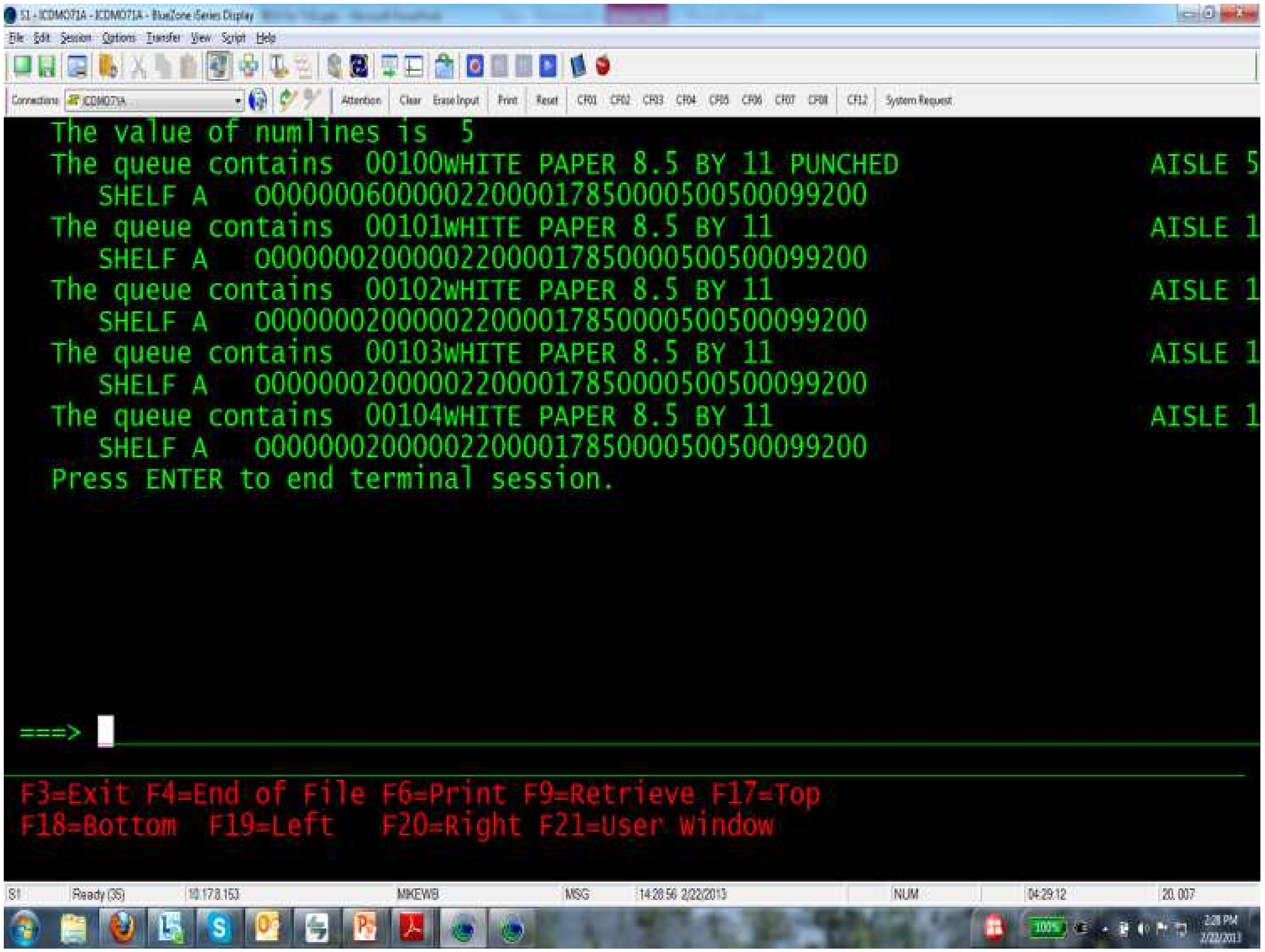
```
                         Create Command (CRTCMD)

 Type choices, press Enter.

 Command . . . . . . . . . . . . > CPYRGZFILI    Name
   Library . . . . . . . . . . . >   MIKEW_X     Name, *CURLIB
 Program to process command . . . > *REXX        Name, *REXX
   Library . . . . . . . . . . .                 Name, *LIBL, *CURLIB
 Source file . . . . . . . . . . > QCMDSRC       Name
   Library . . . . . . . . . . . >   MIKEW_X     Name, *LIBL, *CURLIB
 Source member . . . . . . . . . > CPYRGZFILI    Name, *CMD
 REXX source file . . . . . . . .  QREXSRC       Name
   Library . . . . . . . . . . . .   MIKEW_X     Name, *LIBL, *CURLIB
 REXX source member . . . . . . .  *CMD          Name, *CMD
 REXX command environment . . . .  *COMMAND      Name, *COMMAND, *CPICOMM...
   Library . . . . . . . . . . . .               Name, *LIBL, *CURLIB

                                                                  More...
 F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
 F13=How to use this display        F24=More keys
```

Call it…



```
                        Copy File to Rexx Queue (CPYFTOREXQ)

 Type choices, press Enter.

 From File . . . . . . . . . . . . . > PRODUCT        Name
   Library . . . . . . . . . . . . . >   PAYROLL      Name, *CURLIB, *LIBL
 Member . . . . . . . . . . . . . .    *FIRST         Name, *FIRST
 Number of records to copy . . . . > 2                Number, *ALL

                                                                        Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Run the REXX …



```
   The value of numlines is  5
   The queue contains  00100WHITE PAPER 8.5 BY 11 PUNCHED                AISLE 5
      SHELF A   00000006000002200001785000050050009920O
   The queue contains  00101WHITE PAPER 8.5 BY 11                        AISLE 1
      SHELF A   000000020000022000017850000500500099200
   The queue contains  00102WHITE PAPER 8.5 BY 11                        AISLE 1
      SHELF A   000000020000022000017850000500500099200
   The queue contains  00103WHITE PAPER 8.5 BY 11                        AISLE 1
      SHELF A   000000020000022000017850000500500099200
   The queue contains  00104WHITE PAPER 8.5 BY 11                        AISLE 1
      SHELF A   000000020000022000017850000500500099200
   Press ENTER to end terminal session.

 ===>

 F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
 F18=Bottom  F19=Left   F20=Right F21=User Window
```

File Handling – Using SQL

- Yes you have access to SQL in REXX!
- Need to specify the SQL command environment
  - ADDRESS EXECSQL
- Typical command:
  - ADDRESS EXECSQL 'INSERT INTO DB/TABLE VALUES(789)'
- Check whether SQL call was successful in REXX RC variable
  - Also display SQLCODE and SQLSTATE from the SQLCA (SQL Communications Area)
  - For full details see http://publib.boulder.ibm.com/infocenter/iseries/v7r1m0/index.jsp?topic=%2Frzajp%2Frzajprexx.htm

Small example: (Create the table)

```
ADDRESS '*EXECSQL'
   EXECSQL,
      'SET OPTION COMMIT = *NC'

tablespec =  lib"/INVENTORY",
          "(ITEM_NUMBER CHAR(6) NOT NULL, ",
           "ITEM_NAME VARCHAR(20) NOT NULL WITH DEFAULT '***UNKNOWN***', ",
           "UNIT_COST DECIMAL(8,2) NOT NULL WITH DEFAULT, ",
           "QUANTITY_ON_HAND SMALLINT DEFAULT NULL, ",
           "LAST_ORDER_DATE DATE, ",
           "ORDER_QUANTITY SMALLINT DEFAULT 20, ",
           "PRIMARY KEY(ITEM_NUMBER)) "

ADDRESS '*EXECSQL'
   EXECSQL,
      'CREATE TABLE' tablespec

SAY lib"/INVENTORY CREATED"
SAY “SQLCODE =“ SQLCODE
SAY “SQLSTATE =“ SQLSTATE
```

Small example: (Populate it)

```
/*  Data for Inventory Table   */
inum.1 = '153047';inam.1 = 'Pencils,red';ucost.1 = 10.00;qoh.1 = 25;
inum.2 = '229740';inam.2 = 'Lined tablets';ucost.2 = 1.50;qoh.2 = 120;
inum.3 = '544931';inam.3 = 'UNKNOWN      ';ucost.3 = 5.00;qoh.3 = 50;
inum.4 = '303476';inam.4 = 'Paper Clips  ';ucost.4 = 2.00;qoh.4 = 100;
inum.5 = '559343';inam.5 = 'Envelopes, legal';ucost.5 = 3.00;qoh.5 = 500;

ADDRESS '*EXECSQL'
Do datagroups
Do x = 1 to 5
insert_stmt = lib"/INVENTORY ",
              "(ITEM_NUMBER,",
              "ITEM_NAME,",
              "UNIT_COST,",
              "QUANTITY_ON_HAND)",
        "VALUES('"inum.x"',",
              "'"inam.x"',",
 ucost.x",",
              qoh.x
 EXECSQL,
        'INSERT INTO' insert_stmt
 END
 END
```

Some examples (*optional*)

COMPARE(string1,string2,*pad*)

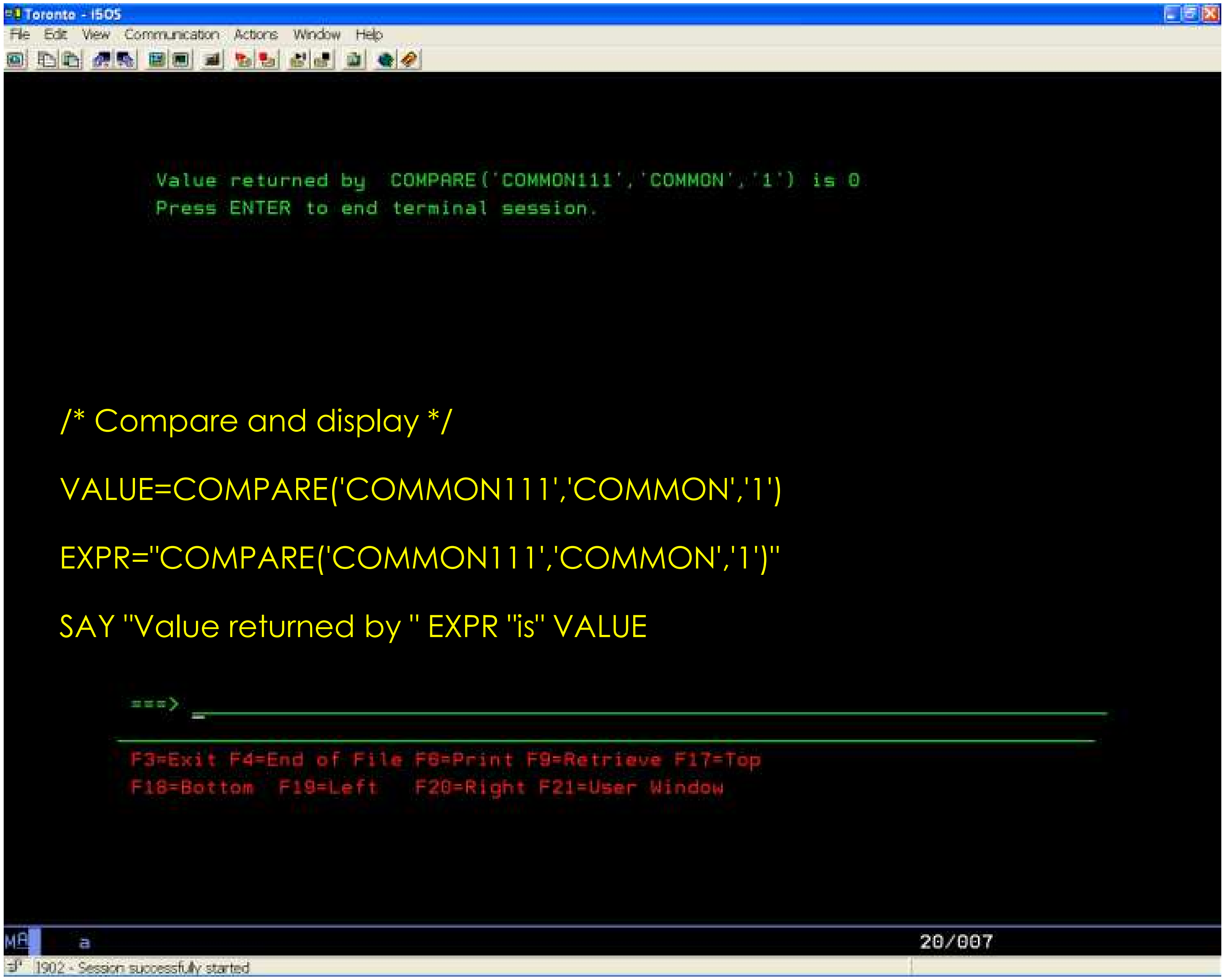
returns 0 if a match else

position of first character that does not match

pad shorter string with *pad* if necessary

COMPARE('Common','Common')	=>	0
COMPARE('Common','Code')	=>	3
COMPARE('Common ','Common',' ')	=>	0
COMPARE('mylib--- ','mylib','-')	=>	9
COMPARE('Common111,'Common','1')	=>	0

Here is a simple way to test this...



```
Value returned by  COMPARE('COMMON111','COMMON','1') is 0
Press ENTER to end terminal session.
```

```
/* Compare and display */

VALUE=COMPARE('COMMON111','COMMON','1')

EXPR="COMPARE('COMMON111','COMMON','1')"

SAY "Value returned by " EXPR "is" VALUE
```

```
===> _

F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window
```

```
MA     a                                           20/007
1902 - Session successfully started
```

Some examples (optional)

DATATYPE(string,*type*)

returns NUM if a valid number
  else returns CHAR
If *type* specified, returns 1 if string matches *type*
  else returns 0

Valid types:

Alphanumeric	**M**ixed case
Binary	**N**umber
C (Mixed SBSC/DBSC)	**S**ymbol
Dbcs	**U**ppercase
Lowercase	**W**hole number
	he**X**adecimal)

Some examples (optional)

DATATYPE(string,*type*)

returns NUM if a valid number
else returns CHAR
If *type* specified, returns 1 if string matches *type*
else returns 0

DATATYPE(‘ 15 ‘)	=>	‘NUM’
DATATYPE(‘123*’)	=>	‘CHAR’
DATATYPE(‘125.7’,’N’)	=>	1
DATATYPE(‘125.7,’W’)	=>	0
DATATYPE(‘Mikey’,’M’)	=>	1
DATATYPE(‘BC d3’,’X’)	=>	1

Some examples (*optional*)

DATE(*option*)

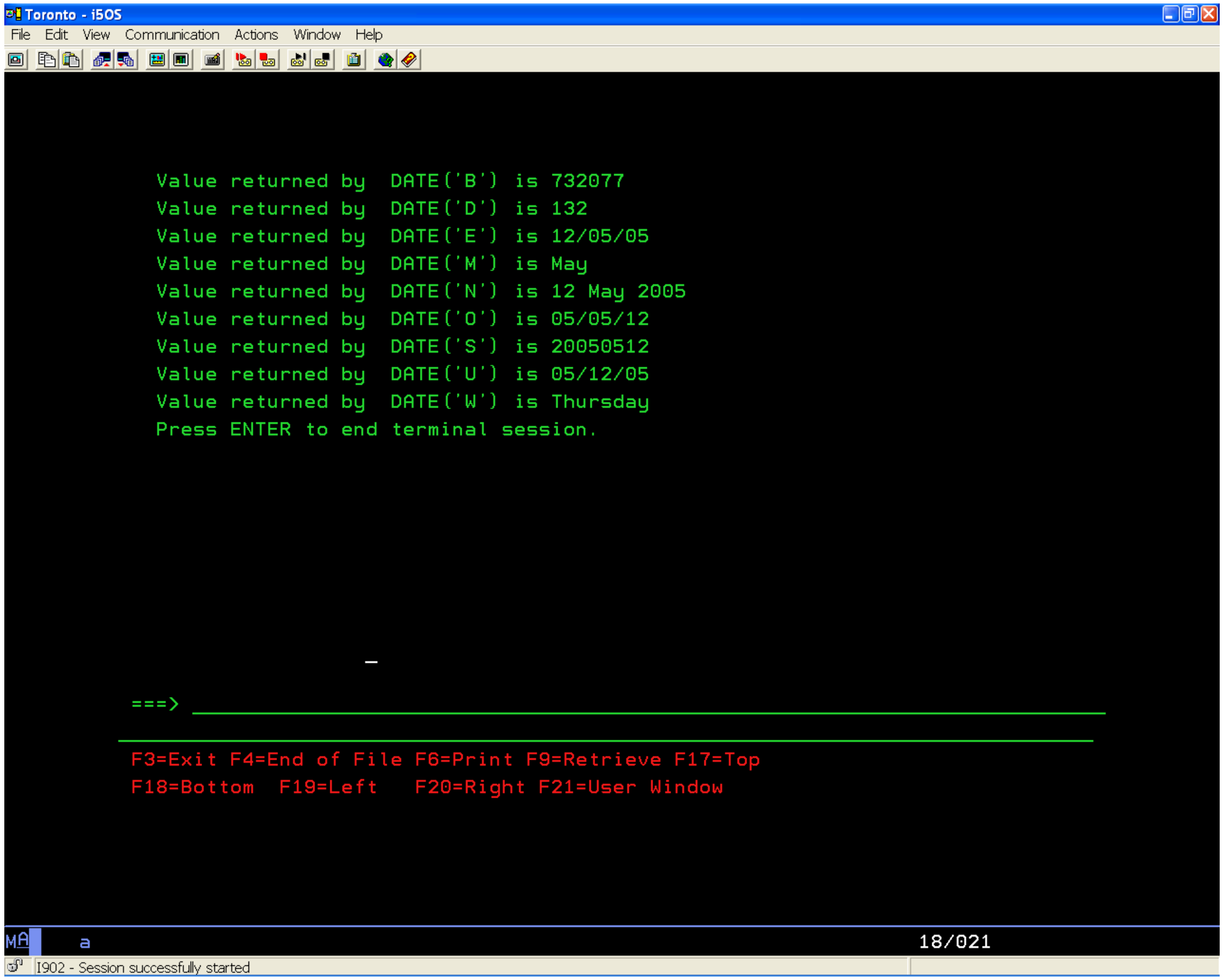
returns local date in format dd mon yyyy

If *option* specified, returns local date in format specified by *option*

Valid options:

- **B**ase (number of days minus today since 1 January 0001 in format dddddd)
- **D**ays (number of days including today so far this year in format ddd
- **E**uropean (current date in format dd/mm/yy)
- **M**onth (full English name of current month i.e. June)
- **N**ormal (the default dd mon yyyy)
- **O**rdered (format yy/mm/dd – suitable for sorting)
- **S**tandard (format yyyymmdd – suitable for sorting)
- **U**sa (format mm/dd/yy)
- **W**eekday (returns English name of day of the week i.e. Monday)

DATE Function running



```
Value returned by  DATE('B') is 732077
Value returned by  DATE('D') is 132
Value returned by  DATE('E') is 12/05/05
Value returned by  DATE('M') is May
Value returned by  DATE('N') is 12 May 2005
Value returned by  DATE('O') is 05/05/12
Value returned by  DATE('S') is 20050512
Value returned by  DATE('U') is 05/12/05
Value returned by  DATE('W') is Thursday
Press ENTER to end terminal session.

===> 

F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window

MA     a                                             18/021
I902 - Session successfully started
```

Some examples (optional)

INSERT(new,target,*n*,*length*,*pad*)

inserts new padded with *pad* or truncated - to length *length* into target after *n*th character

defaults: *n*=0, *length* = length of new, *pad* = ‘ ‘

INSERT(‘ ‘,’SimonCowell’,5)	=>	‘Simon Cowell’
INSERT(‘789’,’xyz’,5,6,’+’)	=>	‘xyz++789+++‘
INSERT(‘789’,’xyz’,5,6)	=>	‘xyz 789 ‘
INSERT(‘789’,xyz’)	=>	‘789xyz’
INSERT(‘789’,’xyz’,,5,’*’)	=>	‘789**xyz’

Some examples (optional)

LASTPOS(needle,haystack,<u>*start*</u>)

returns position of last occurrence of needle in haystack, zero if needle is null string or not found

defaults: backward scan

```
LASTPOS(‘ ‘,’this is really weird’)       =>  15
LASTPOS(‘ ’,’thisisreallywierd’)          =>  0
LASTPOS(‘45’,’12345’)                     =>  4
LASTPOS(‘ ’,’this is really weird’,7)     =>  5
```

Some examples (optional)

MAX(number)

returns maximum in a list of numbers

MIN(number)

returns minimum in a list of numbers

Maximum of 20 numbers – you can nest if you need more!

MAX(12,6,7,9)	=> 12
MAX(-7,-5,-6.3,-15)	=> -5
MIN(100,25,-6,10)	=> -6
MIN(21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,MIN(2,1))	=> 1

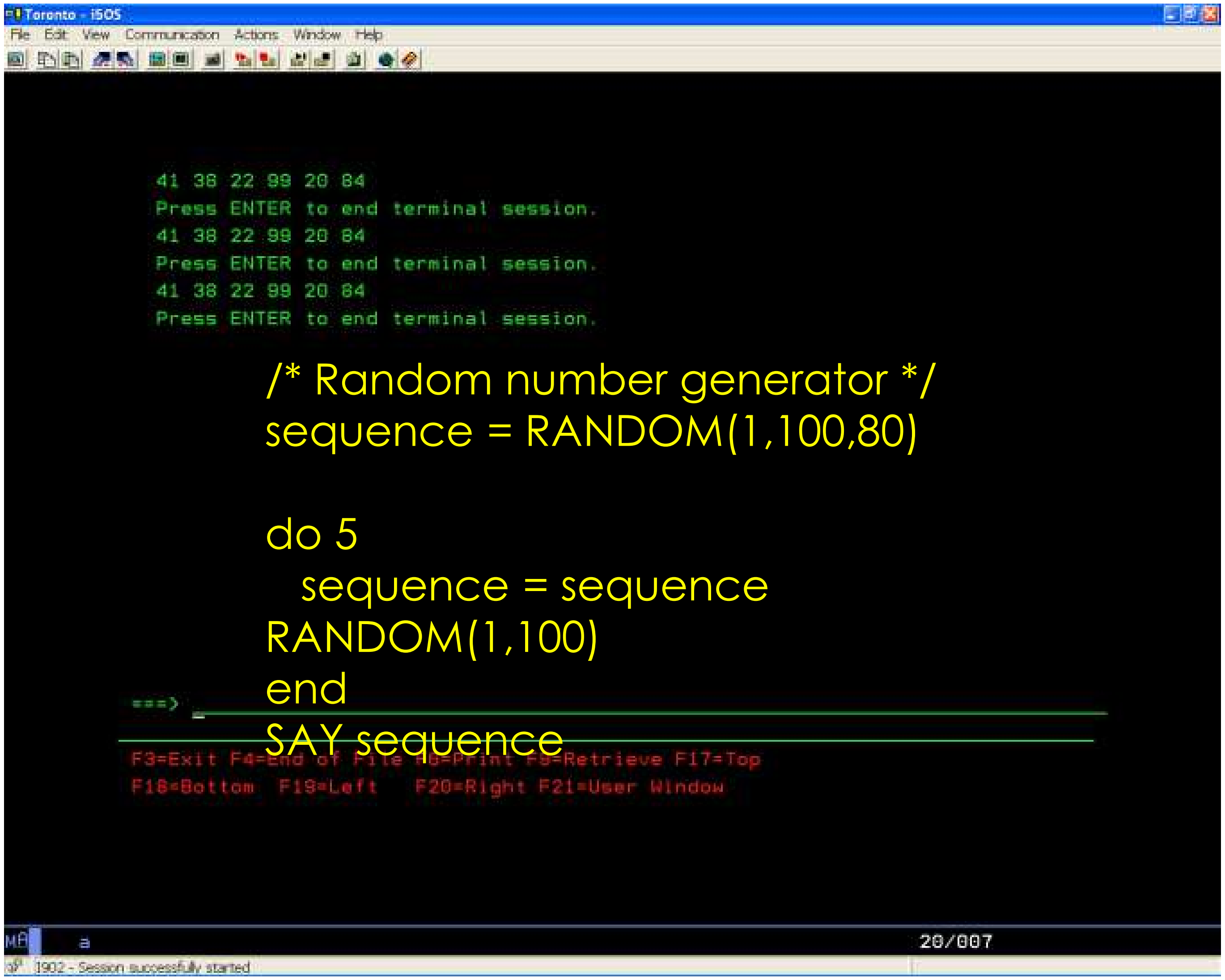
Some examples (optional)

RANDOM(<u>*min*</u>,<u>*max*</u>,<u>*seed*</u>)

- generates a random positive whole number between min and max
- seed provides for reproducible random number
- Defaults: <u>*min*</u>=0, <u>*max*</u>=999, <u>*max*</u>-<u>*min*</u><100000, <u>*seed*</u><999999999

```
RANDOM()                 =>  42?
RANDOM(16,57)            =>  33?
RANDOM(1)                =>   1?
RANDOM(,,63782)          => 567?
```

Random function running



Toronto - iSOS
File Edit View Communication Actions Window Help

```
41 38 22 99 20 84
Press ENTER to end terminal session.
41 38 22 99 20 84
Press ENTER to end terminal session.
41 38 22 99 20 84
Press ENTER to end terminal session.
```

```
/* Random number generator */
sequence = RANDOM(1,100,80)

do 5
   sequence = sequence
RANDOM(1,100)
end
SAY sequence
```

```
===> _
F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window
```

MA a 20/007
1902 - Session successfully started

Some examples (optional)

TIME(*option*)

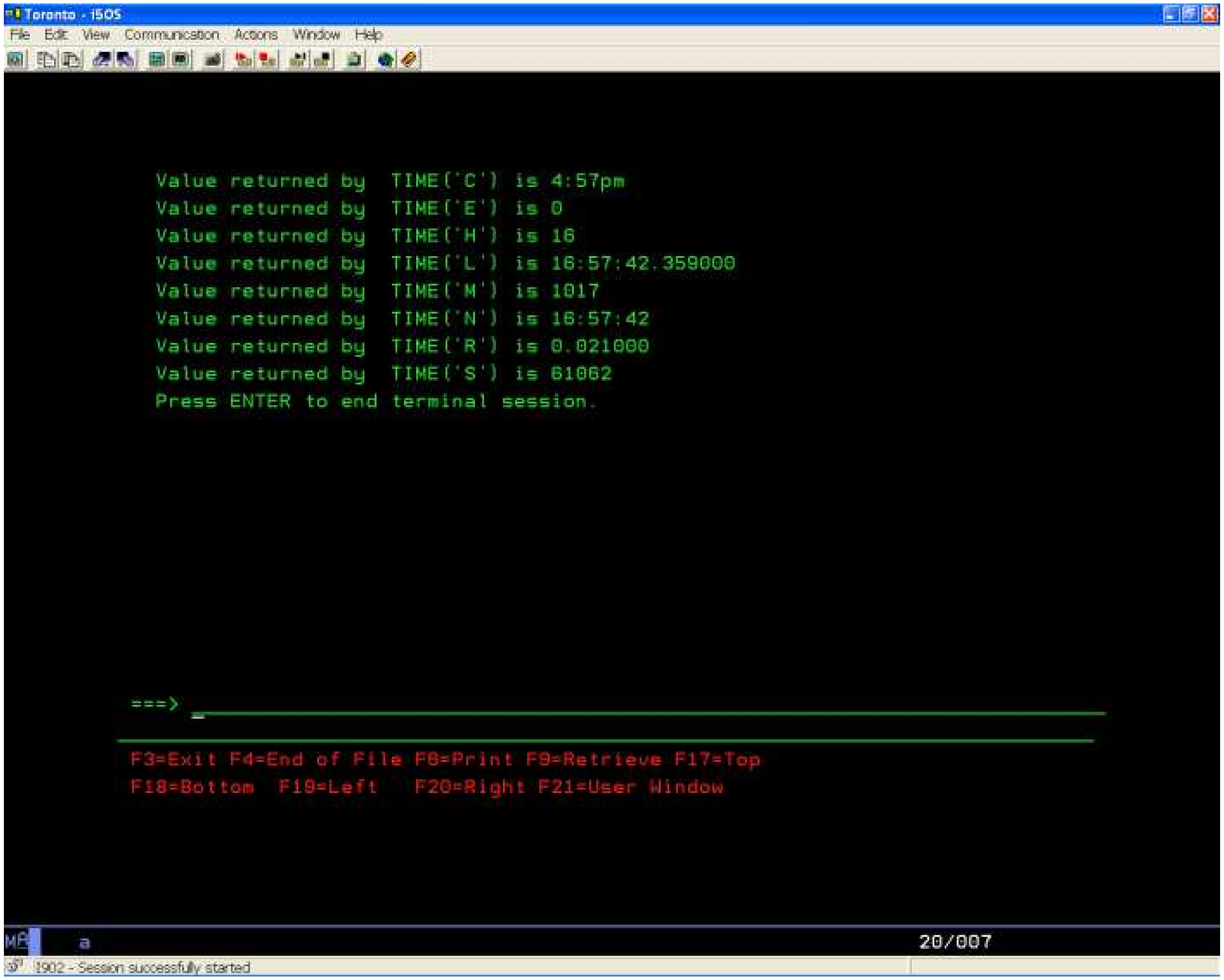
returns local time in 24 hour clock format hh:mm:ss

If *option* specified, returns local time in format specified by *option*

Valid options:

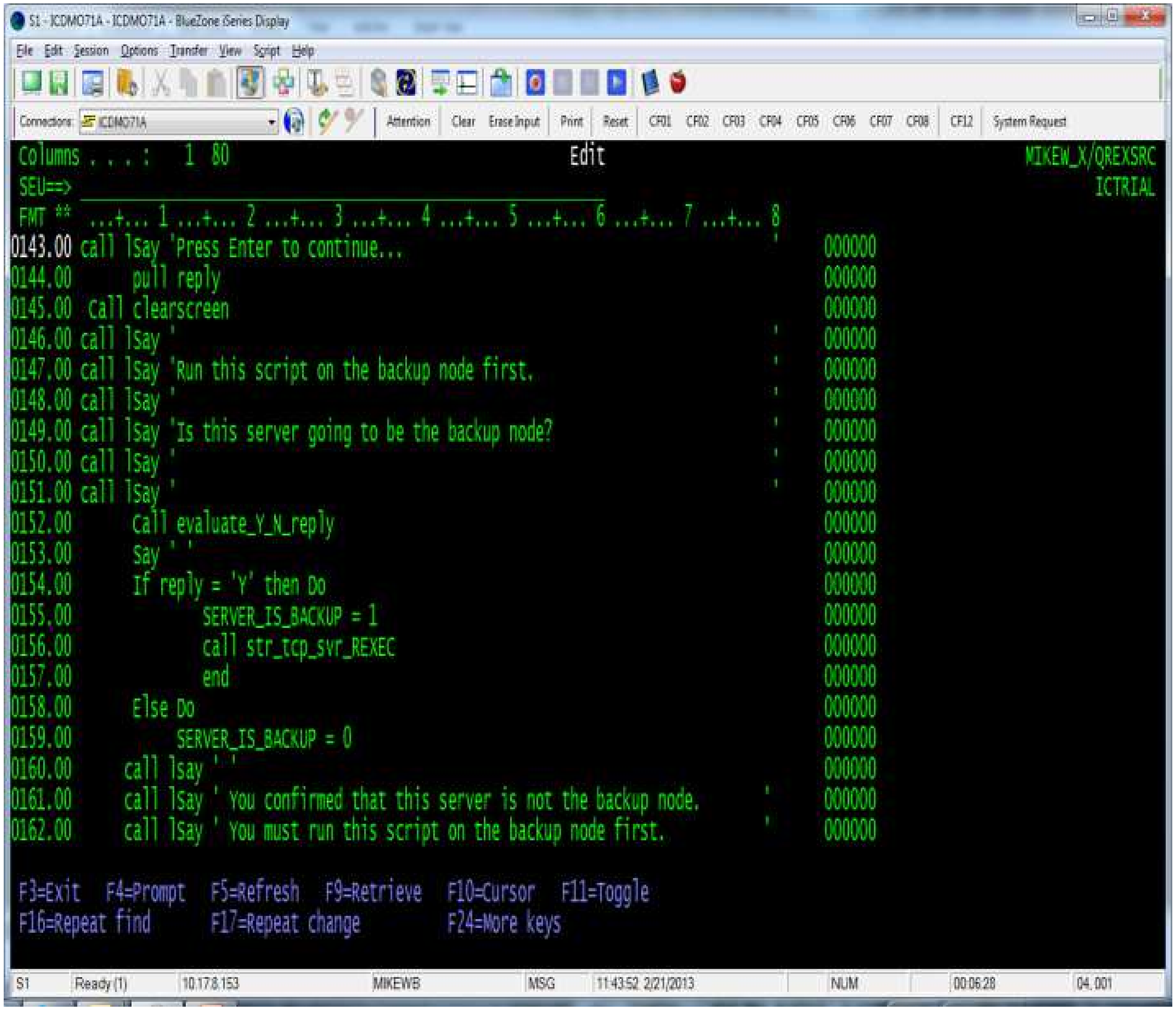
- **C**ivil (current time in format hh:mmxx)
- **E**lapsed (number of seconds.microseconds since elapsed clock reset)
- **H**ours (number of hours since midnight in format hh)
- **L**ong (current time in format hh:mm:ss.uuuuuu)
- **M**inutes (number of minutes since midnight in format mmmm)
- **N**ormal (the default hh:mm:ss)
- **R**eset (returns same as Elapsed and resets elapsed clock to zero)
- **S**econds (number of seconds since midnight in format sssss)

TIME function running



```
Value returned by  TIME('C') is 4:57pm
Value returned by  TIME('E') is 0
Value returned by  TIME('H') is 16
Value returned by  TIME('L') is 16:57:42.359000
Value returned by  TIME('M') is 1017
Value returned by  TIME('N') is 16:57:42
Value returned by  TIME('R') is 0.021000
Value returned by  TIME('S') is 61062
Press ENTER to end terminal session.




===> _

F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window
```

A Real World Use



S1 - ICDMO71A - ICDMO71A - BlueZone iSeries Display
File Edit Session Options Transfer View Script Help
Connections: ICDMO71A
Attention Clear Erase Input Print Reset CF01 CF02 CF03 CF04 CF05 CF06 CF07 CF08 CF12 System Request

```
 Columns . . . :    1  80                          Edit                          MIKEW_X/QREXSRC
 SEU==>  ____________________________________________________________                  ICTRIAL
 FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
0143.00 call lSay 'Press Enter to continue...                                   '    000000
0144.00      pull reply                                                              000000
0145.00  call clearscreen                                                            000000
0146.00 call lSay '                                                             '    000000
0147.00 call lSay 'Run this script on the backup node first.                    '    000000
0148.00 call lSay '                                                             '    000000
0149.00 call lSay 'Is this server going to be the backup node?                  '    000000
0150.00 call lSay '                                                             '    000000
0151.00 call lSay '                                                             '    000000
0152.00      Call evaluate_Y_N_reply                                                 000000
0153.00      Say ' '                                                                 000000
0154.00      If reply = 'Y' then Do                                                  000000
0155.00              SERVER_IS_BACKUP = 1                                            000000
0156.00              call str_tcp_svr_REXEC                                          000000
0157.00              end                                                             000000
0158.00      Else Do                                                                 000000
0159.00          SERVER_IS_BACKUP = 0                                                000000
0160.00     call lsay ' '                                                            000000
0161.00     call lSay ' You confirmed that this server is not the backup node.     '   000000
0162.00     call lSay ' You must run this script on the backup node first.         '   000000

 F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
 F16=Repeat find       F17=Repeat change          F24=More keys
```

S1 Ready (1) 10.17.8.153 MIKEWB MSG 11:43:52 2/21/2013 NUM 00:06:28 04, 001

Additional References

- The REXX Language Association

  www.rexxla.org

  Annual International REXX Symposium

  May 5-8 Comfort Suites Raleigh/Durham Airport
- Wikipedia (the free Web Encyclopedia)

  http://en.wikipedia.org/wiki/REXX
- The IBM REXX Language Page

  maintained by Uwe Berger at IBM Germany

  http://www-01.ibm.com/software/awdtools/rexx/
- The REXX Language –A Practical Approach to Programming (TRL-2)

  THE book by Michael Cowlishaw IBSN 0-13-780651-5
- Regina – open source REXX

  http://regina-rexx.sourceforge.net
- IBM Info Centre

  REXX/400 Programmer's Guide SC41-5728

  REXX/400 Reference SC41-5729

Thank you!

Rocket ®

Your Business ▪ Your Agenda ▪ Our Passion
Maximizing the value of IT together

Rocket®

REXX for CL Programmers!
TUG Mar 20, 2013

Mike Warkentin
Managing Director R&D
mwarkentin@rocketsoftware.com
(781) 577-4344