IBM Software

REXX Compiler

REXX Symposium, Tampa
May 1, 2007

George Fulk
fulkgl@us.ibm.com

ON DEMAND BUSINESS™

George Fulk

- 1973 Mag card terminal, #987 golf-ball, 134.5 acoustic coupler to APL/SV
- 79,80 “Pre-professional”
- 81 hired full-time. APL expert. VM/CMS. Started using REX[X].
- 81-89 IBM Endicott, packaging assurance. Mainframe hardware organization. Math, custom built hardware and software.
- 89-93 IBM Boca Raton, OS/2 development DOS Emulation. OS/2 Rexx.
- 93-96 IBM OS for running all software. PC-DOS Rexx.
- 96 IBM Austin, OS/2 Warp 4
- 96-97 OS/2 running Win32/Office
- 97-98 Setup-top box. WebTV, Tivo.
- 99-00 PC-DOS. Embedded DOS, large hard drive, FAT32 file system.
- 00-05 Build/tools group.
- 04-now PC-DOS
- 05 ANT/WASD support
- 06-now Rexx compiler. oRexx/ooRexx.

REXX Compiler on z/OS and z/VM

- **IBM Compiler for REXX on zSeries Release 4**
  - VM, MVS: PID 5695-013
- **IBM Run Time Library for REXX on zSeries Release 4**
  - VM, MVS: PID 5695-014
  - VSE part of operating system
- **IBM Alternate Library for REXX on zSeries Release 4**
  - Free download
  - http://www-306.ibm.com/software/awdtools/rexx/rexxzseries/altlibrary.html
  - Included in z/OS 1.9 base operating system
- **Continued ongoing support for Release 4**
  - Release 4 has been available since 2003

REXX History

- **REXX = Restructured eXtended eXecutor**
- **1979mar29 Mike Cowlishaw (IBM Fellow) publishes initial specification**
- **Late 1979 first implementation internal to IBM on VM/CMS.**
- **Available to the general public in 1983 VM (3rd release)**
- **1985 first non-IBM version appears.**
- **1987 IBM announces Rexx to be the Procedures Language for SAA (Systems Application Architecture)**
  - MVS/TSO, AS/400, 1989 OS/2 1.2 EE
- **1989 REXX Compiler for MVS and VM released**

REXX History

- 1990 first (annual) Rexx Symposium
- 1990 Rexx 4.0 Language published. OS/2 1.3 first implementation.
- Early 1990s versions available for AIX/6000, PC-DOS, Netware, CICS.
- 1996 ANSI “Programming Language REXX”, X3.274-1996
- 1996 NetRexx for Java
- 1996 Object REXX released OS/2 version 4. 1997 Windows and Linux, 1998 AIX, 2000 Linux/390, 2002 Solaris.
- 2003 REXX Compiler Release 4 (aka Version 1.4)
- 2005 Open source ooRexx

REXX Compiler History

- **Mid-80's proof of concept at IBM research Haifa**
- **Late-80's implementation at IBM research Vienna**
- **1989 first release available from IBM**
- **Mid-90's REXX merged to a single location, IBM Boeblingen**
- **Late-90's/early-2000's contracted with a company in Russia to implement the Interpret command**
- **Last major release in 2003 (release 4)**
- **2006 compiler support transferred to IBM Austin**

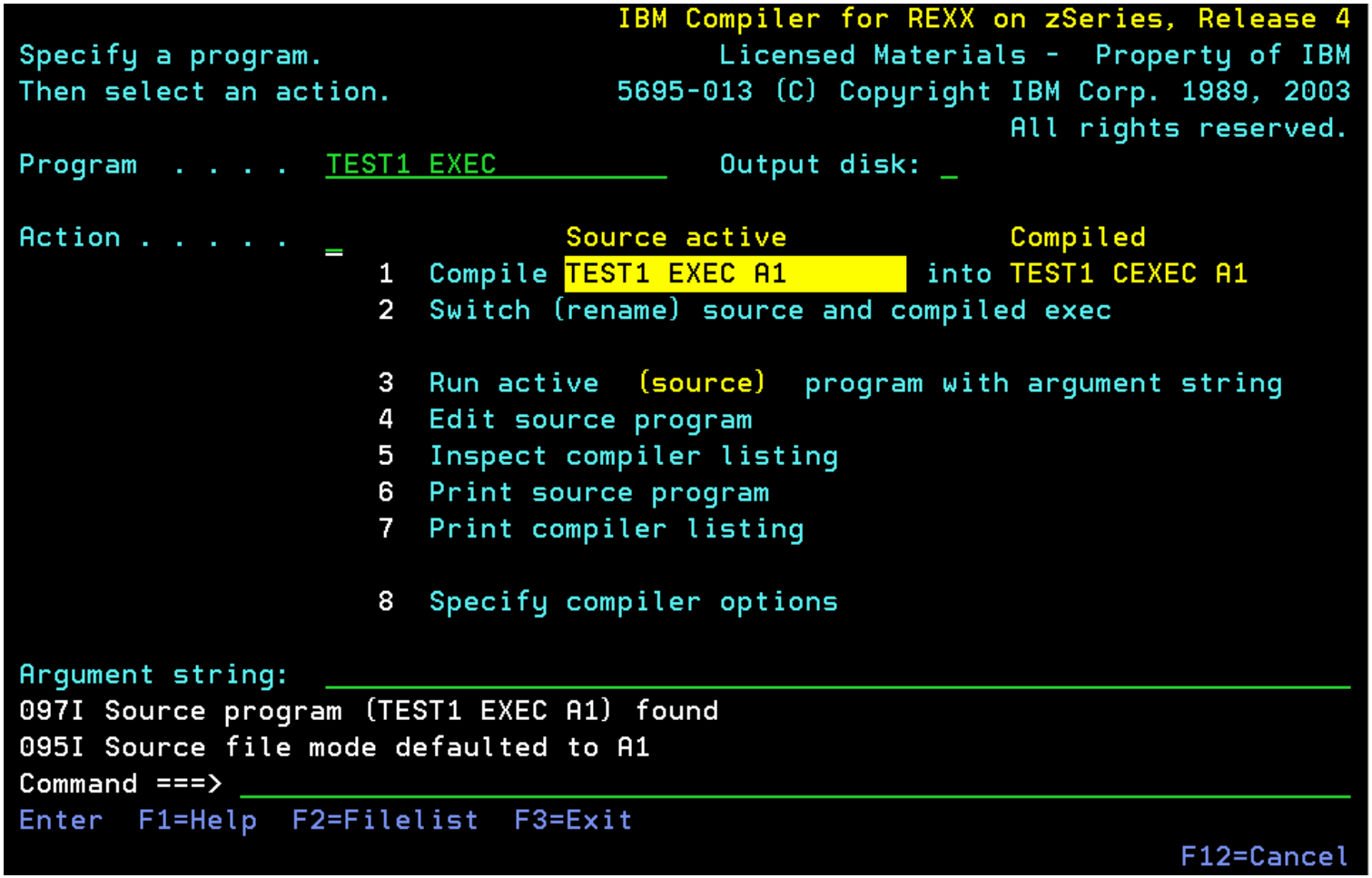
REXX introduction… (deleted several pages)

this page intentionally left blank

Invoking Compiler under z/VM

- **Invoke on command line**
  - REXXC test1 exec (xref
  - REXXC *source_file* (*options*
  - REXXC or REXXC ? or HELP REXXC
- **Invoke with full screen panel**
  - REXXD

REXXD under z/VM



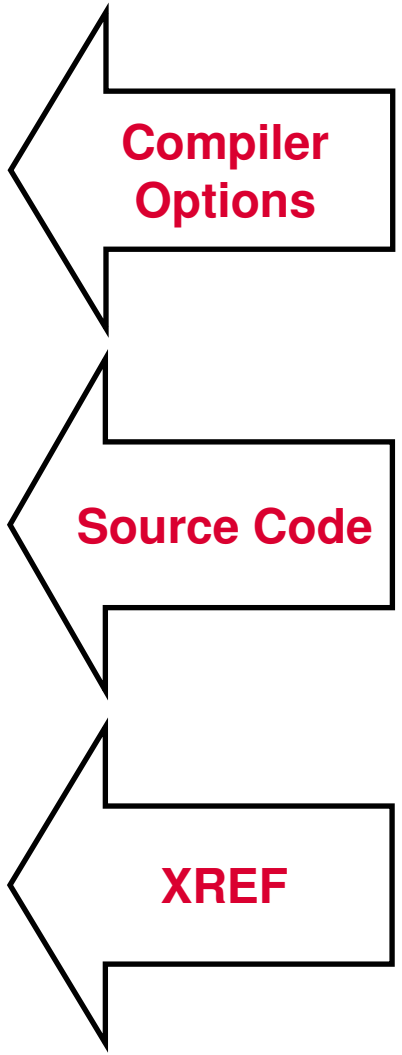
REXXD test1 exec

Compiler Listings

```
1===> Compilation Summary                          TESTDATE EXEC     A1     00001
 IBM Compiler REXX/370 3.0  LVL PQ00090    Time: 17:54:31                   00002
                                                                            00003
 Compilation successful                                                     00004
 Compiler Options                                                           00005
                                                                            00006
      NOALTERNATE                                                           00007
        CEXEC     (TESTDATE CEXEC    A1)                    RECFM=F,LREC  00008
.....................................                                       00009
1===> Source Listing                               TESTDATE EXEC     A1     00010
 IBM Compiler REXX/370 3.0  LVL PQ00090    Time: 17:54:31                   00011
  If  Do Sel  Line C ----+----1----+----2----+----3----+----4----+----5--   00012
                                                                            00013
                1   /* REXX Program: TESTDATE EXEC                   */     00014
                2   say date(u)      '... US       format     MM/DD/YY'     00015
                3   say date(e)      '... European fomat      DD/MM/YY'     00016
                4   say date(s)      '... Standard            YYYYMMDD'     00017
                5   say '---- convert S-date(20011130) to US Format ----'   00018
                6   say date(u,'20011130',s)                                00019

 IBM Compiler REXX/370 3.0  LVL PQ00090    Time: 17:54:31                   00021
 Item                          Attribute Line References                    00022
                                                                            00023
 ----- Labels, Built-in Functions, External Routines -----                  00024
 DATE                          BUILT-IN  2 3 4 6                            00025

 ----- Constants -----                                                      00027
                                                                            00029
 /YY'                                                                       00030
 '... Standard            YYYYM LIT STR   4                                 00031
 MDD'                                                                       00032
 'to US Format -----'                                                       00036
```

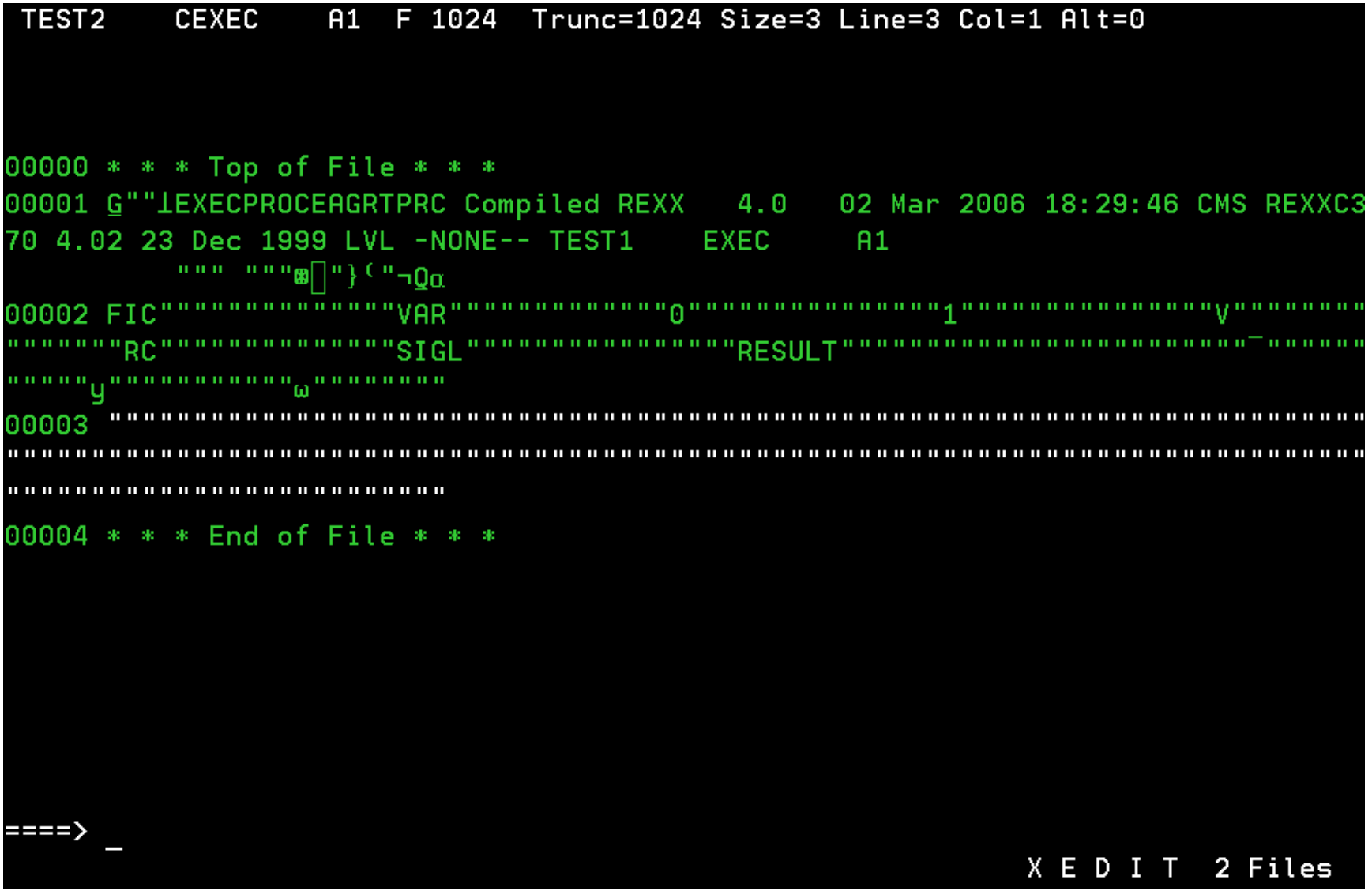


Source to Executable

- **Source → Object(s) → Executable**
  - `rexxc test exec ( nocexec obj(test object)`
  - `load test object a`
  - `genmod test module a`
- **Source → Executable**
  - `rexxc test exec ( cexec(public exec) noobj`
- **MVS**

```
//COMPILE EXEC REXXC,OPTIONS='CEXEC NOOBJ'
//REXX.SYSIN DD DSN=FULKGL.SHARE.REXX(TEST),DISP=SHR
//REXX.STEPLIB DD DSN=RXT.REXX.V140.SFANLMD,DISP=SHR
//REXX.SYSCEXEC DD DSN=FULKGL.SHARE.CEXEC(PUBLIC),DISP=SHR
//SYSPRINT DD DSN=SYSOUT*
```

Executable from: rexxc test2 (cexec



REXX Compiler Libraries

- **Requires a REXX library to execute compiled programs**
- **Compiled REXX is not a LE language**
- **2 choices: Run-time library and Alternate library**
  - Run-time library. Program product.
  - Alternate library. Free. Uses the native system's REXX interpreter.
- **Compiled and library code runs in 31-bit mode**
  - base/displacement instead of relative addressing
  - BALR and other old opcodes. Can run on old hardware.
  - No z/Architecture in plan today.

Primary and Alternate Libraries

- **Primary Library (PID 5695-014)**
  - `rexxc test exec (cexec(test1 exec)`
  - `exec test1`
    - test1 → primary library
- **Alternate Library**
  - `rexxc test exec (cexec(test2 exec) alt sl`
  - `exec test2`
    - test2 → alternate library → system interpreter
- **Will run whichever library is loaded in memory**
  - Alternate library execution requires ALT and SLINE

Alternate library shipped with R1.9 of z/OS

- **Starting with release 1.9 (Sept 2007) of z/OS the alternate library is shipped with the base OS.**
- **Identical to the free, downloadable, distributable alternate library.**
  - No need for software developers to include the alternate library with their shipped packages.
  - No need for users to download and install the alternate library.

Compiler Advantages

- **Program performance**
  - Known value propagation
  - Assign constants at compile time
  - Common sub-expression elimination
  - stem.i processing
- **Source code protection**
  - Source code not in deliverables
- **Improved productivity and quality**
  - Syntax checks all code statements
  - Source and cross reference listings
- **Compiler control directives**
  - %include, %page, %copyright, %stub, %sysdate, %systime, %testhalt

Performance: String Processing

```
type test1 exec

/* REXX */
PARSE VERSION v; say v
N=TIME('E')
DO I=1 TO 5000000
    A = 'STRING'
    N = I || A
END
SAY TIME('E')

Ready;
exec test1
REXX370 4.02 01 Dec 1998
17.377110
Ready;
rexxc test1 exec (ce(test2 exec)
Ready;
exec test2
REXXC370 4.02 23 Dec 1999
1.755673
Ready;
_
                                                    RUNNING    VMSDVM6
```

Performance: Arithmetic Operations

```
type test1 exec

/* REXX */
PARSE VERSION v; say v
N=TIME('E')
DO I=1 TO 5000000
    N = I * I
END
SAY TIME('E')

Ready;
exec test1
REXX370 4.02 01 Dec 1998
33.500449
Ready;
rexxc test1 exec (ce(test2 exec)
Ready;
exec test2
REXXC370 4.02 23 Dec 1999
1.665910
Ready;

_                                          RUNNING   VMSDVM6
```

Performance

▪ Arithmetic operations ▪ String and word processing	6 to 10 times faster
▪ Constants and variables ▪ References to procedures and built-in functions ▪ Changes to values and variables	4 to 6 times faster
▪ Assignments ▪ Reused compound variables	2 to 4 times faster
▪ Host commands ▪ File I/O	Minimal improvement

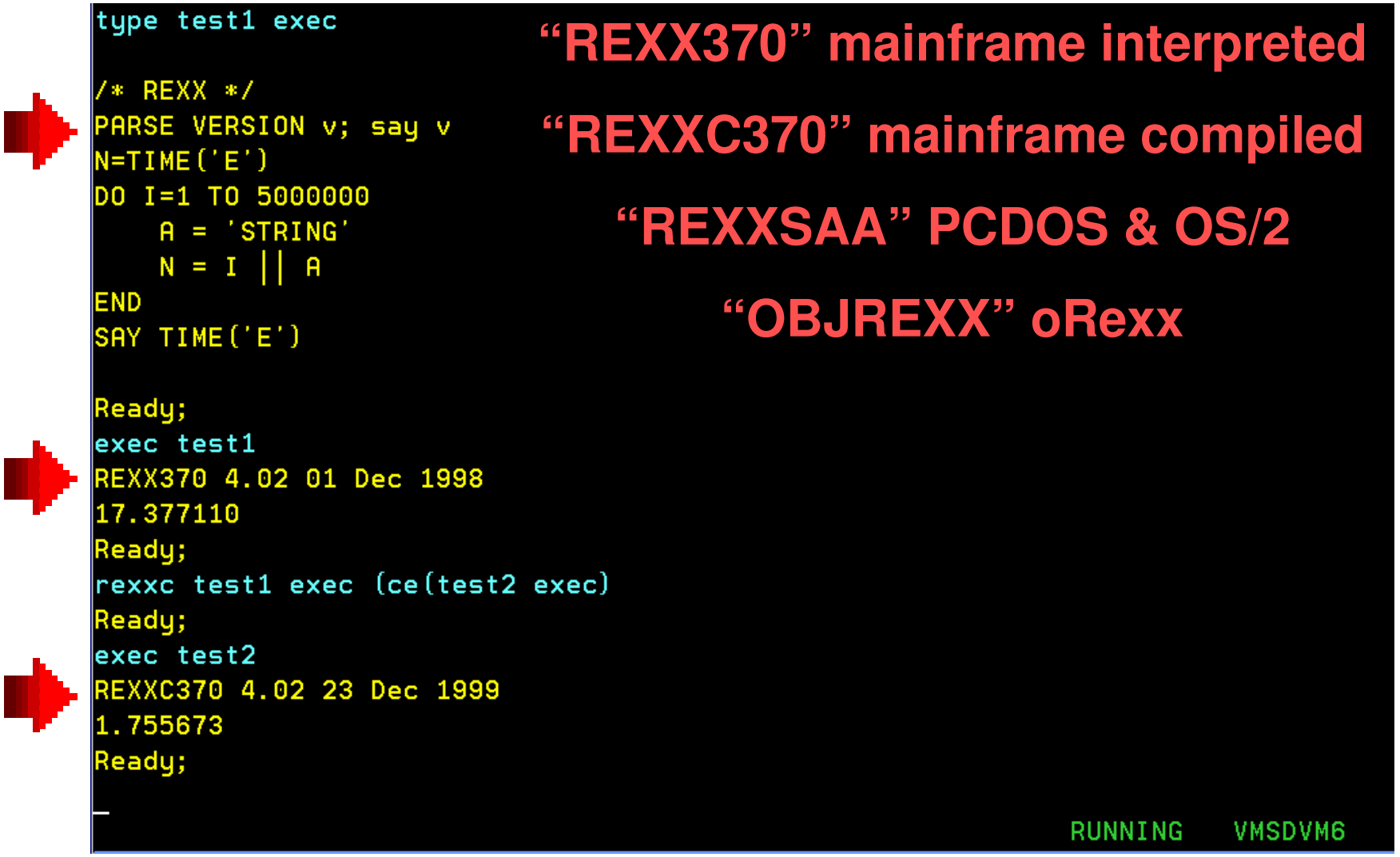
Source Code Protection

- **Protects your intellectual property**
- **Protects your code from manipulation**
- **Keeps your code maintainable**

Improved Productivity and Quality

- **Debugging: cross reference listing**
- **Syntax check of all statements**
- **Syntax check without code execution**
- **Compiler error messages**
- **Lists all errors – no stopping at first error**

How to tell interpreted versus compiled REXX



Compiler options

- **ALTERNATE – marks the resulting binary as capable of being run with the alternate library.**
- **CEXEC(filespec) – specifies the output is a "compiled-exec" type of file.**
- **COMPILE – is output binary produced.**
- **CONDENSE – packs binary. Unpacked at run time by the library.**
- **DLINK – TEXT file contains external references.**
- **DUMP – used to debug compiler**
- **FLAG – minimum severity for messages**

Compiler options, page 2

- **FORMAT – listing file options**
- **IEXEC – interpretable program**
- **LIBLEVEL(6) – library level required**
- **LINECOUNT(55) – listing file option**
- **MARGINS(1 *) – listing file option**
- **OBJECT(filespec) – binary output is TEXT**
- **OLDDATE – date/time other than compiler time**
- **OPTIMIZE – debug compiler**

Compiler options, page 3

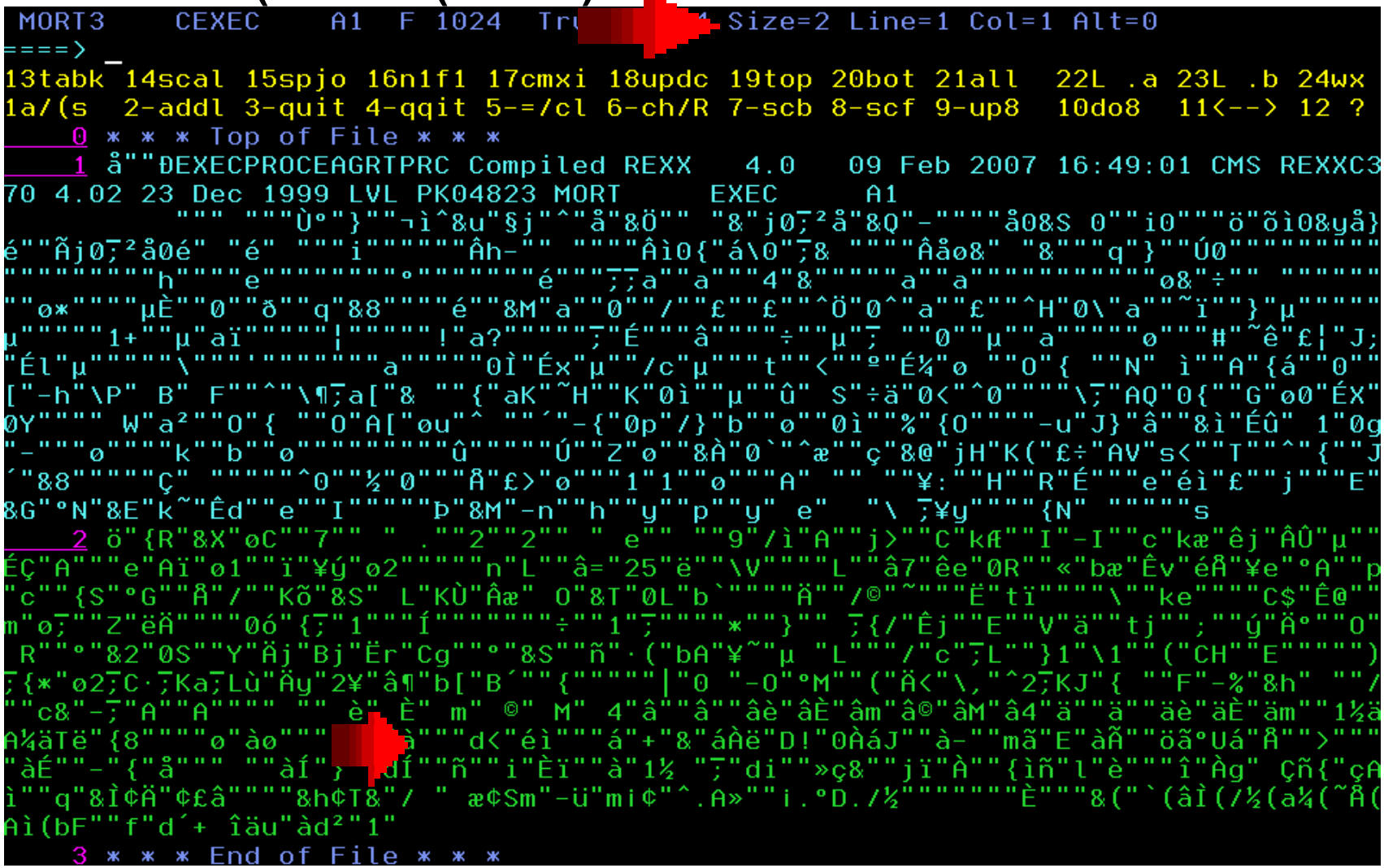
- **PRINT(filespec) – listing file**
- **SAA – check source for SAA**
- **SLINE – includes source lines in binary**
- **SOURCE – listing file option**
- **TERMINAL – quiet/verbose compile**
- **TESTHALT – include code for HALT trapping**
- **TRACE – program uses the TRACE command**
- **XREF – listing file option**

Compiler NOCONDENSE option

rexxc mort (CEXEC(mort2) SLINE ALT NOCOND

Compiler CONDENSE option

rexxc mort (CEXEC(mort3) SLINE ALT COND



Compiler Control Directives, page 1 of 3

- **/*%COPYRIGHT IBM Corp. 2006 */**
- **/*%INCLUDE member */**
- **/*%INCLUDE ddname(member) */**
- **/*%INCLUDE cmsfilename */ /*filetype=COPY*/**
- **/*%Page*/**

Compiler Control Directives, page 2 of 3

- **/*%Stub stubname*/**
  - OBJECT option must be used with compile
- **CPPL (command processor parameter list)**
- **EFPL (external function parameter list)**
- **CPPLEFPL (CPPL and EFPL)**
- **CALLCMD (calling TSO/E command line using TSO/E call command or from another Rexx exec)**
- **MVS (calling JCL EXEC PGM=)**
  - ADDRESS LINKMVS or ADDRESS ATTACHMVS
- **MULTI (multi-purpose stub)**
  - ADDRESS LINK or ADDRESS ATTACH

Compiler Control Directives, page 3 of 3

- **/*%Sysdate*/**
- **/*%Systime*/**
- **/*%Testhalt*/**
  - Inserts the code to support the HALT condition.

%COPYRIGHT

rexxc mort (CE(mort3) SLINE ALT COND

Added to code… /*%COPYRIGHT IBM 2007*/

Live example (dataset allocation)

Allocate New Data Set	.rexx	.jcl	.cexec	.object	.module
Volume serial	REXX01	REXX01	REXX01	REXX01	REXX01
Space units		TRACK	TRACK	TRACK	TRACK
Primary quantity	5	5	5	30	5
Secondary quantity	5	5	25	5	10
Directory blocks	5	5	5	5	5
Record format	VB	FB	FB	FB	U
Record length	255	80	80	80	80
Block size	6233	3120	27920	9040	9040
Data set name type	PDS	PDS	PDS	PDS	PDS

Live example (JCL)

```
//BUILD   JOB (ACCOUNT),'SHARE',CLASS=A,MSGCLASS=R,LINES=999999,
//        NOTIFY=FULKGL,MSGLEVEL=(1,1),REGION=4096K,TIME=1440
//*****************************************************************
//**
//**
//*****************************************************************
//** (1) COMPILE REXX PROGRAM
//**   INPUT:  REXX.SYSIN = REXX SOURCE CODE
//**           REXX.STEPLIB = COMPILER LIBRARY (REXXC SFANLMD)
//**   OUTPUT: REXX.SYSPUNCH = OBJECT OUTPUT
//**           REXX.SYSCEXEC = CEXEC OUTPUT
//**
//*****************************************************************
//COMPILE     EXEC REXXC,
//            OPTIONS='XREF OBJECT NOSLINE'
//REXX.SYSIN    DD DSN=FULKGL.SHARE.REXX(MORT),DISP=SHR        SOURCE
//REXX.STEPLIB  DD DSN=RXT.REXX.V140.SFANLMD,DISP=SHR    COMPILER-LIB
//REXX.SYSPUNCH DD DSN=FULKGL.SHARE.OBJECT(MORT),DISP=SHR      OBJECT
//REXX.SYSCEXEC DD DSN=FULKGL.SHARE.CEXEC(MORT),DISP=SHR
//SYSPRINT      DD SYSOUT=*
//**
```

Live example (JCL)

```
/*************************************************************
//** (2) CREATE OBJECT DECK WITH MULTISTUB (EAGSTMP BY USING REXX
//** EXEC REXXL, ALIAS OF EAGCML)
//**    INPUT:  SYSIN = OUTPUT FROM COMPILER
//**            SYSEXEC = LOCATION FOR REXXL (REXX SOURCE FOR BIND)
//**    OUTPUT: SYSOUT = TEMPORARY OBJECT DECK (INPUT TO LINK)
//**
//*************************************************************
//STUB     EXEC PGM=IRXJCL,PARM='REXXL MULTI'
//SYSIN    DD  DSN=FULKGL.SHARE.OBJECT(MORT),DISP=SHR          OBJECT
//SYSEXEC  DD  DSN=RXT.REXX.V140.SEAGCMD,DISP=SHR               REXXL
//SYSOUT   DD  DSN=&&SYSPUNCH(GLFTEST),                    STUBBED-OBJ
//         DISP=(NEW,PASS,DELETE),UNIT=SYSDA,SPACE=(TRK,(2,1,2)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120,DSORG=PO)
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//**
```

Live example (JCL)

```
//*****************************************************************
//** (3) LINK THE STUB WITH COMPILED REXX
//**   INPUT:  SYSLIN = (INPUT FROM STUB)
//**           SYSLIB = LIBRARY FOR LINK/REXX
//**   OUTPUT: SYSLMOD = RESULTING END PRODUCT MODULE
//**
//*****************************************************************
//LINK     EXEC PGM=HEWL,PARM='LIST,AMODE=31,RMODE=ANY,RENT,MAP'
//SYSLIN   DD  DSN=&&SYSPUNCH(GLFTEST),DISP=(SHR,PASS)    STUBBED-OBJ
//SYSLIB   DD  DSN=RXT.REXX.V140.SEAGLMD,DISP=SHR            LINK-LIB
//SYSLMOD  DD  DSN=FULKGL.SHARE.MODULE(MORT),DISP=SHR          MODULE
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1024,(200,20))
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//**
```

Live example (JCL)

```
//*****************************************************************
//** (4) RUN COMPILED STUB
//**   INPUT:  STEPLIB = LOCATION OF MODULE & RUN-TIME LIBRARY
//**
//*****************************************************************
//RUN      EXEC PGM=MORT,PARM='160000 6.375 30',REGION=512K
//STEPLIB  DD  DSN=FULKGL.SHARE.MODULE,DISP=SHR                MODULE
//         DD  DSN=RXT.V140.SEAGLPA,DISP=SHR                   RT-LIB
//SYSTSPRT DD  SYSOUT=*
//**
//**
//*****************************************************************
//** (5) RUN CEXEC
//**   INPUT:  STEPLIB = LOCATION OF RUN-TIME LIBRARY
//**
//*****************************************************************
//RUNCE    EXEC PGM=IKJEFT01,REGION=4M
//STEPLIB  DD  DSN=RXT.V140.SEAGLPA,DISP=SHR                   RT-LIB
//SYSTRPRT DD  SYSOUT=*
//SYSTSIN  DD  *
EX 'FULKGL.SHARE.CEXEC(MORT)' '165000 6.25 30'
/*
//**
//**
//************************ END ************************************
```

Live example (Rexx)

```
/* REXX */
PARSE VERSION v1; PARSE SOURCE v2; SAY v1 v2
ARG prin rate yrs
monthly = prin * (rate/1200) * (1+1/(((1+rate/1200)**(12*yrs)-1)))
say "Monthly payment = $" || format(monthly,,2)
EXIT 0
```

Summary

- **Source code protection**
- **Performance enhancement**
- **Compile time code checks**
- **Support and service**

- **User's Guide and Reference**

http://publibfi.boulder.ibm.com/epubs/pdf/h1981605.pdf

Closing

- **Product web sites**
  - http://www-306.ibm.com/software/awdtools/rexx/rexxzseries/
    - Publications
    - Pre-requisites
    - Announcements
    - Support
- **E-mail: George Fulk, fulkgl@us.ibm.com**