OS/2 Procedures Language 2/REXX

Richard K. McGuire and Stephen G. Price
IBM
OS/2 Procedures Language
2/REXX
"A Practical Approach to Programming"
and
"Adding REXX Power to Applications"

Richard K. McGuire
Stephen G. Price

IBM Corporation
G09/20M
P.O. Box 6
Endicott, NY 13760

(C) Copyright IBM Corp 1989, 1992
OS/2 Procedures
Language
2/REXX

A
Practical
Approach to
Programming

What is REXX?

- Powerful end-user programming language
- Easy to learn, easy to remember
- Can powerfully extend any application
- Common language available on all SAA systems
- Becoming an ANSI standard (X3J18 Committee)
Why REXX?

- Small, easy to use, yet powerful language
- Programming interfaces for application extension
- Rapid development of an interpreter, performance boost of compiler technology

Keep the Language Small

- Friendlier to new users
- Documentation is smaller and simpler
- Few exceptions or special cases (low "astonishment factor")
- Users can "embrace" the entire language
Natural Datatyping

- No internal or machine representation is exposed to the user
- Single number concept

<table>
<thead>
<tr>
<th>Say &quot;The interest is a*b%&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Say 5 + 1.0 + 0.54 + 1.23e-2</td>
</tr>
</tbody>
</table>

OS/2

Rexx

No Defined Size or Shape Limits

- Data sizes limited only by available memory
- Limits are set using "human readable" values
- SmallTalk-like dynamic data-typing

OS/2

Rexx
Powerful Symbol Manipulation

- Natural concatenation
- Powerful string parsing ability
- Many functions for string and word manipulation

Parse Arg first initial last
Say "Hello" first'.

pos = wordpos(first, list)
if pos <> 0 then
    nickname = word(list, pos)

OS/2

Rexx

System Independence

- The REXX language is independent of both operating system and hardware
- Suitable for any system or application environment
- Part of the IBM Systems Application Architecture

OS/2

Rexx
REXX Uses

- Tailoring user commands (*.CMD* files)
- End-user problem solving
- Universal macro or scripting language
- Prototyping Applications
- Education

OS/2

REXX

Universal Macro Language

- Editors
- Spreadsheets
- Language preprocessors
- Communication programs
- Rexx can be the macro language for any application

OS/2

REXX
REXX is a Good Introduction to Programming

- Easy to learn
- Easy to program
- Few new concepts required
- Powerful debugging features
- No separate compile or link step

OS/2

What's New in OS/2 2.0?

- Interpreter runs in 32-bit mode
- Dramatically improved performance
- New 32-bit interfaces
- 16-bit interfaces still supported
- On-line REXX reference manual
- OS/2 utility functions
- New 32-bit sample programs in toolkit
- On-line programming interfaces reference
- RXHLLAPI interface
- SAA Communications interface
- Communications Manger configuration
- LAN utilities

OS/2

Rexx

223
More than a Fancy .CMD Language

- Fill multiple roles on OS/2
- Places more power in the hands of users
- Powerful automation of OS/2 operations
- Powerful extensions to OS/2 applications

OS/2     Rexx

OS/2 Procedures Language

2/REXX

Adding REXX Power to Applications
Creating New REXX Functions

```rexx
call SysCls
version = SysOS2Ver()
call SysSleep 10
```

Function Registration

- REXX external functions are registered with `RxFuncAdd`
- Acts as a form of program linkage

Call `RxFuncAdd 'SysCls', 'REXXUTIL', 'SysCls'`
RXSTRINGs

- External functions are passed arguments as RXSTRINGs.
- Defined as a pointer and length pair defining a REXX character string.

Call SysFileTree 'C:\*.*', 'List.', 'F'

<table>
<thead>
<tr>
<th>strlength</th>
<th>ptr</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>C:*.*0</td>
</tr>
<tr>
<td>length</td>
<td>List\0</td>
</tr>
<tr>
<td>strp</td>
<td>F\0</td>
</tr>
</tbody>
</table>

**OS/2**

**Rexx**

RXSTRING Return Values

- External functions pass an RXSTRING value back to REXX.
- The function can use the buffer provided by REXX or create a new one.

```rexx
REXX

External Function

REXX
```

**OS/2**

**Rexx**
Function Packages

- REXX external functions can be registered from C code also
  
  ```
  RexxRegisterFunctionDll(
      "SysCls", "REXXUTIL",
      "SysCls");
  ```

Accessing REXX Variables

**OS/2**  
**Rexx**

**OS/2**  
**Rexx**

**Accessing REXX Variables**

**Call SysFileTree file, 'List.'**

**REXX Program**

**APIENTRY SysCls(PSZ Name, ...**

- "set a list variable ";

```
rc = RexxVariablePool(Request);
```
Using REXX for Macros

- An application can call the REXX interpreter to run any REXX program

OS/2

Invoking REXX

- An application can call use the REXX interpreter with the RexxStart programming interface

Rexx

```rexx
rc = RexxStart(argc, argv, "FACTOR.CAL", NULL, NULL, RXFUNCTION, NULL, &return, &retstr);
```

OS/2

Rexx
Application External Functions

```
RexxRegisterFunctionEx("CalcPrecision", 
                  (PFN)CalcPrecision); 
RexxStart(argc, argv, "FACTOR CAL", ..

APIRET APIENTRY CalcPrecision(PS2 name, 
   LONG argc, RXSTRING *argv, PSZ queue, 
   RXSTRING *result)

/* Calculate factorial * /
   parse arg number, factor
   numeric digits CalcPrecision()
```

OS/2

```
Rexx
```

Subcommand Handlers

```
RexxRegisterSubcomEx("Calc-Precision", 
                  (PFN)CalcCommand, NULL); 
RexxStart(argc, argv, "FACTOR CAL", NULL, 
           "CalcCommand", RXFUNCTION, NULL, ..

APIRET APIENTRY CalcCommand( 
   RXSTRING command, PUSHORT flags, 
   RXSTRING result)

/* Calculate factorial */
   parse arg number, factor
   numeric digits CalcPrecision()
   If (datatype(number, 'W' then 
       SET DISPLAY Error
```

OS/2

```
Rexx
```

229
And Still More...

- Exits to tailor REXX program behavior
- REXX programs executed directly from storage
- Macro Space repository for REXX programs
- Halting a running REXX program
- Tracing a running REXX program
- Subcommand handlers as dynamic link libraries

OS/2

REXX
The Universal Macro Language

- Same language used for all applications
- Places control into user hands, making people more productive
- Easily added to any application

OS/2

Rexx