Good Rexx Coding Practices

Les Koehler – IBM Tampa, FL

Objectives

Performance
Maintainability

Sources Include

- Various FORUMs on the IBMVM conferencing disk
- Private correspondence with REXX Development
- Private correspondence with REXX inventor (Mike Cowlishaw)
- Many years of personal experience with REXX (first user outside of Hursley)
- Exposure to a multitude of internal tools
- Examination of thousands of lines of REXX code by various authors worldwide

- Many years of personal experience with REXX (first user outside of Hursley)
- Exposure to a multitude of internal tools
- Examination of thousands of lines of REXX code by various authors worldwide

Performance

1. Use "Address Command" at the beginning of execs, or at least before you issue any commands to CP/CMS.   This speeds execution and avoids unintended execution of execs and synonyms.
2. For XEDIT macros, use "address command" on commands directed to CP/CMS.
3. Xedit commands should be explicitly sent to the appropriate process (COMMAND, MACRO, CMS, or CP).
4. Quote and upper case all commands to underlying systems. This includes PIPE commands to the Command and CP stages.
5. Never quote built-in and external REXX functions *unless* you have a subroutine that usurps the function and must then invoke the function itself.
6. Put prolog comments before a subroutine label.

7. Avoid large Block Comments like
   ```
   /******/
   /*      */
   /*      */
   /*      */
   /******/
   ```
   Disadvantages:
   - ☞ Eats up extra file space
   - ☞ Harder to maintain
   - ☞ May consume extra cycles if scanned

Instead use one of these styles:

/*	/*	/*	/*
**	*	|	
**	*	|	
*/	*/	*/	*/

8. Do not put a "Do" on a separate line. Instead, include it on the appropriate statement line. This saves screen and paper vertical space and makes the code easier to maintain.
9. Be aware that multiple “& | ¬”on an "If" or "When" reduce performance since they are all evaluated. This may also cause hard-to-find errors if a variable is not initialized.
10. Use "Iterate" where appropriate, perhaps in place of "Nop".
11. Use "Parse" to assign default values:

```
parse value copies(’0 ’,10) with
def1, def2 ... def10
```

In place of multiple assignment statements when blanks are insignificant and each variable is a single word:

```
parse value oldvar1 oldvar2 with,
newvar1 newvar2 .
```

12. Use "Left(var,1)" instead of "substr(var,1,1)"
13. Use "Right(var,1)" instead of "substr(var,length(var),1)"
14. Use multiple stemmed variables instead of one huge stemmed variable.
15. Use 'Content addressable arrays' where appropriate.
16. It is generally pointless to put a blank around comparison operators or around the "=" of an assignment statement.

Maintainability:

1. Indent your code (two spaces is enough)
2. Outdent your "End" statements thusly:
   ```
   If a=2 Then Do
      /* Whatever */
      /* Whatever */
   End
   ```
3. Indent the code in a subroutine and outdent the Return statement.
4. Put an 'eye-catcher' box before a subroutine. My REXLABEL macro-generates one that looks like this:
   ```
   /***************************
    *   NAME_OF_SUBROUTINE   *
    ***************************/
   Name_of_subroutine:
   ```

5. Put a blank line (NOT a boxed comment) between logical segments of code.
6. Do not abbreviate commands. The only exception allowed would be for CP commands inside a long running loop.
7. Don't code 'long lines' that aren't viewable w/o wrapping in Xedit. Presume the next person will be limited to 72 characters of file area.

8. Use "Signal on Syntax" and "Signal on Error" and include the appropriate labels.
"Signal on Novalue" is often wise also.
9. Use 'ESTATE' instead of 'STATE', just in case the file should grow to more than 64K lines.
10. Always end a 'Parse' with a '.' (or some varname, like Trash) to pick up any leftover stuff.
11. Use descriptive variable names. The extra key-stroking will be well worth it in the long run.
12. Do not use REXX keywords or function names as variable names
13. Avoid the use of the stack whenever possible. If you *must* use it, surround the command with MAKEBUF/DROPBUF.
14. Remember that REXX puts a blank between 'words'. Thus the style:

```
say 'Error on line 'sigl
```

is explicitly telling REXX to do what it would have done anyway if you had coded:

```
say 'Error on line' sigl
```

which is easier to read anyway!
15. Do not use long prologs. Use epilogs instead.
16. The history of changed execs should be maintained using a naming convention like this:

SEXEC1
SEXEC2

etc.
Substitute XEDIT or REXX for EXEC as appropriate.