



Treehouse Software, Inc.



REXX-like Macro Language for UNIX and Windows

Product Overview

TREEHOUSE SOFTWARE, INC.
2605 Nicholson Road, Suite 230
Sewickley, PA 15143
Phone: 724.759.7070
Fax: 724.759.7067
E-mail: sedit@treehouse.com
<http://www.treehouse.com>

This page intentionally left blank.

From Mainframe to UNIX or Windows

Mixed Environments Many sites are moving from the mainframe to UNIX and Windows, or are implementing combined mainframe/UNIX/ Windows environments. These sites are finding that the change takes time. It is not a simple or quick process.

Productivity The mainframe, UNIX, and Windows platforms differ significantly. Just becoming accustomed to the operating system differences is time-consuming. Users are forced to learn different ways to achieve the same result, based on the computing platform they are using. At the same time, they must learn new tools, new languages, and new ways of approaching problems. Learning and retaining these different skill sets can initially decrease productivity and can frustrate productive, experienced mainframe users. These users want to get to work immediately, but their lack of experience with UNIX and Windows tools makes it difficult for them to accomplish even the most simple tasks.

To eliminate some of the problems involved in mixed environments, sites need familiar tools that operate in the same manner on UNIX and Windows as they do on the mainframe. Using similar tools on both platforms minimizes retraining requirements, makes the best use of existing skills, and ensures almost immediate productivity in both environments.

**S/REXX:
The Solution**

This is why the S/REXX macro language was created.

Introducing S/REXX

Compatibility

S/REXX is a macro language for UNIX and Windows which emulates the IBM mainframe REXX language. S/REXX is compatible with the instructions and built-in functions of Cowlshaw REXX level 4.0. This "instant familiarity" helps programmers make a smooth transition from the mainframe to UNIX or Windows.

Power

S/REXX provides additional capabilities that make it more flexible, more powerful, and easier to use than mainframe REXX. For example, S/REXX has no limits on implementation size or shape. S/REXX also places no limitations on:

- Procedure size
- Expression complexity
- Number of nested parentheses
- Number and content of variables
- Recursive function depth
- Argument number and size

In addition, S/REXX extends the REXX language to take advantage of UNIX and Windows operating system resources and facilities.

Mainframe Familiarity

S/REXX enables programmers to perform the same tasks on UNIX and Windows as they did on the mainframe with IBM's REXX. S/REXX provides a familiar language that can be used to create macros, perform system administration functions, and even develop applications.

EXECIO Support

S/REXX includes an implementation of the EXECIO instruction to facilitate the porting of REXX procedures from VM/CMS. The EXECIO instruction reads and writes to files, and prints on native UNIX printers. All options of the CMS EXECIO instruction are supported.

The example below shows an excerpt from an S/REXX program. Note its similarity to mainframe REXX programs.

```
#!/home/xed/srex  
  
say 'Enter a positive number'  
pull rep .  
  
if test_nump(rep) then say 'Valid number'  
else                 say 'Invalid number'  
  
exit  
test_nump:  
if datatype(rep) = 'NUM' & rep > 0 then return 1  
else                                     return 0
```

UNIX-Specific Extensions

S/REXX offers several capabilities not provided by REXX on the mainframe. These additional capabilities allow S/REXX to take full advantage of the UNIX and Windows environments:

- Execution of S/REXX programs by C programs
- Execution of UNIX commands and use of the command output within S/REXX programs
- Support for "Choices", a menu replacement appropriate for use in dialog boxes
- Capability to access the S/REXX external data queue from within C programs
- Support for SEDIT (an XEDIT and ISPF/PDF compatible editor for UNIX and Windows, available from Treehouse Software)

Shared Globals S/REXX can dynamically load external procedures by appending them to the main file. The loaded procedure then becomes an internal procedure, and can share global variables. This feature allows users to build a collection of general utility routines that can be incorporated into different programs. With mainframe REXX, external procedures included in separated files cannot share global variables with the main procedure.

C Language Interface S/REXX can interface with any external application offering a C program interface. The S/REXX programming interface allows users to create new ADDRESS environments. Sending data to these new environments activates user-supplied C subroutines which can retrieve and set S/REXX variables. It is also possible to add, in a similar way, user-supplied, built-in functions written in C. These capabilities make S/REXX an incredibly powerful development tool for UNIX and Windows.

Access to UNIX Facilities

In addition to extending the power and flexibility of the REXX language, S/REXX offers many built-in functions that provide access to UNIX and Windows facilities, including the following:

FD Returns the directory part of a complete UNIX path name

Example: a = fd ("/usr/john/foo.c")

returns: "/usr/john"

FN Returns the filename part of a complete UNIX path name

Example: a = fn("/usr/john/foo.c")

returns: "foo"

FT Returns the file-type part of a complete UNIX path name

Example: a = ft("/usr/john/foo.c")

returns: ".c"

RM Removes a file and returns its complete UNIX path name

Example: say rm("~/foo")

returns: /usr/john/foo

Utilizes UNIX Output S/REXX also enables users to retrieve the result of UNIX commands directly into a REXX expression with the use of backquotes. For example:

```
a=`hostname`
```

retrieves the station hostname and stores it in the variable 'a'.

GUI Extensions

As programmers become more familiar with the features of the UNIX and Windows environments, they will want to take advantage of the various UNIX and Windows Graphical User Interface (GUI) features. S/REXX is ready to grow with them.

In S/REXX, programmers may display OpenLook or MOTIF dialog boxes with input fields, toggles, and buttons, like the one pictured below:

The image shows a graphical user interface dialog box titled "test_dy". It contains the following elements:

- A "DISMISS" button in the top left corner.
- An "EXCHANGE THIS:" label followed by a text input field.
- A "WITH THIS:" label followed by a text input field.
- Two pairs of input fields: "From Column: 1" and "To Column: 123", and "From Line: 12" and "To Line: 13".
- Two checked checkboxes: "Consider Case" and "Whole Word".
- Two dropdown menus: "Net: B" and "Host: C".
- A "DO IT" button at the bottom center.

With its many UNIX-oriented extensions, S/REXX is a language that programmers may continue using for years to come, rather than an interim solution that they eventually discard.

Enhanced Trace Mode

The S/REXX trace mode is more detailed than in the CMS REXX implementation. For example, consider the following REXX commands:

```
trace I
a = 2
str = "This is a"
id = 1
tab.a = substr(str||" string", id+1, 2)
```

Mainframe Trace

When these commands are executed, mainframe REXX displays the following:

```
5 *-* a = 2
   >L> "2"
6 *-* str = "This is a"
   >L> "This is a"
7 *-* id = 1
   >L> "1"
8 *-* tab.a = substr(str || " string",id + 1,2)
   >V> "This is a"
   >L> " string"
   >O> "This is a string"
   >V> "1"
   >L> "1"
   >O> "2"
   >L> "2"
   >F> "hi"
```

S/REXX Trace

While S/REXX displays a more detailed trace:

```
5 *-* a = 2
   >>> A <-- "2"

6 *-* str = "This is a"
   >>> STR <-- "This is a"

7 *-* id = 1
   >>> ID <-- "1"

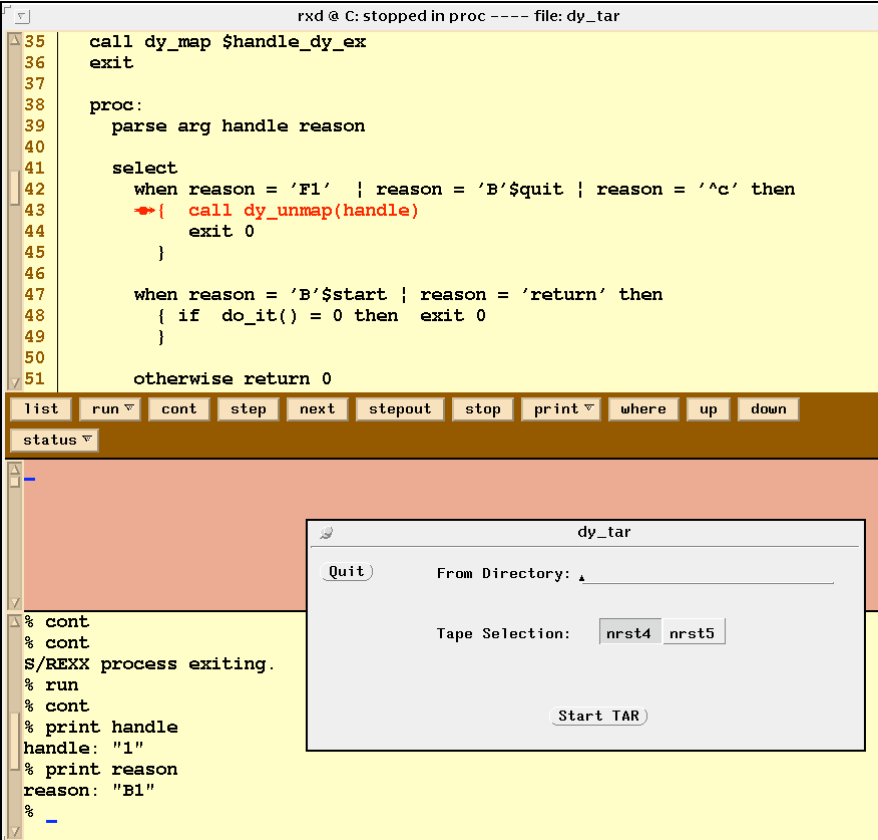
8 *-* tab.a = substr(str||" string",id+1, 2)
   >C> TAB.A --> "TAB.2"
   >V> STR --> "This is a"
   >O> "This is a" || " string" --> "This is a string"
   >V> ID --> "1"
   >O> "1" + "1" --> "2"
   >F> SUBSTR() --> "hi"
   >>> TAB.2 <-- "hi"
```

The detailed S/REXX trace information makes it much easier to debug S/REXX programs. Debugging can be further enhanced by purchasing the optional S/REXX Debugger product (also available from Treehouse Software).

Optional S/REXX Debugger

The separately-priced S/REXX Debugger provides an interactive graphical debugging environment for S/REXX. It allows users to step through the execution of S/REXX programs, finding and fixing bugs quickly.

The S/REXX Debugger allows users to view the source program, stepping through its execution, setting or removing breakpoints, changing source statements, etc. Consider the S/REXX Debugger screen below:



The screenshot displays the S/REXX Debugger interface. The top window shows the source code for a program named 'dy_tar'. The code includes a procedure 'proc:' that takes arguments 'handle' and 'reason'. It uses a 'select' statement to handle different reasons. A red arrow points to line 43, which is 'call dy_unmap(handle)'. Below the source code is a command area with buttons for 'list', 'run', 'cont', 'step', 'next', 'stepout', 'stop', 'print', 'where', 'up', and 'down'. A 'status' dropdown is also present. The bottom area is divided into two sections: the left section shows the debugger's output, including commands like '% cont', '% print handle', and '% print reason', and the right section shows the I/O area for the 'dy_tar' process, which includes a 'Quit' button, a 'From Directory:' field, a 'Tape Selection:' field with buttons for 'nrst4' and 'nrst5', and a 'Start TAR' button.

```
35  call dy_map $handle_dy_ex
36  exit
37
38  proc:
39    parse arg handle reason
40
41    select
42      when reason = 'F1' | reason = 'B'$quit | reason = '^c' then
43        =>{ call dy_unmap(handle)
44           exit 0
45        }
46
47      when reason = 'B'$start | reason = 'return' then
48        { if do_it() = 0 then exit 0
49        }
50
51    otherwise return 0
```

list run cont step next stepout stop print where up down
status

% cont
% cont
S/REXX process exiting.
% run
% cont
% print handle
handle: "1"
% print reason
reason: "B1"
%
-

dy_tar
Quit From Directory: _____
Tape Selection: nrst4 nrst5
Start TAR

The Source area displays the source code for the program being debugged. The Breakpoint column allows the user to set breakpoints at specific lines in the code. The Command area is used to enter debugging commands. The I/O area is used by the S/REXX process to display its output, and permits the user to enter test strings to be sent to the S/REXX process.

Documentation

The S/REXX documentation consists of a fully-indexed User's Guide. S/REXX instructions and functions are fully explained. Dialog management, the programming interface, and the S/REXX Debugger are also described and illustrated.

Support

Treehouse Software provides technical support for S/REXX and its other products from its headquarters in Sewickley, Pennsylvania. User questions are answered quickly, problems discussed directly, and change/enhancement requests given to developers in a timely manner.

S/REXX technical support is available 24 hours a day, 7 days a week through Treehouse Software's technical support network.

Operating System Support

S/REXX and the S/REXX Debugger support Microsoft's Windows NT and Windows 95, IBM's AIX, Hewlett Packard's HP/UX, Sun Microsystems' SunOS and Solaris, Silicon Graphics' IRIX, Santa Cruz Operation's (SCO) UNIX, DEC UNIX, PC Unixware, Linux, and other popular UNIX operating systems.

Pricing

S/REXX is available under a no-obligation, 30-day free trial agreement. During this free trial period, sites have the opportunity to verify the productivity improvements they obtain from S/REXX. The benefits seen during this period can justify the immediate purchase of S/REXX.

License Options

Each fixed or floating license allows a maximum of four users to access S/REXX.

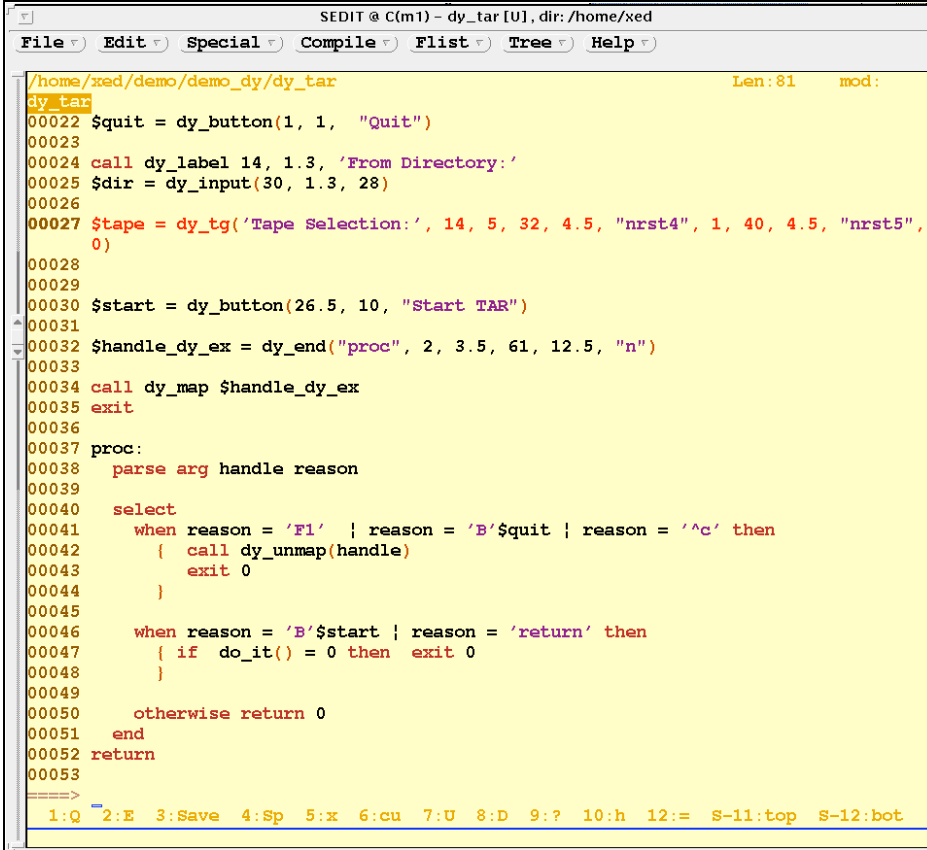
Fixed license option: The fixed license is always accessed on one CPU only, with up to four users on the same CPU. There can be an unlimited number of licenses per CPU. Users cannot use SEDIT on more than one CPU for each four-user fixed license. This is true even if the CPU is networked with other CPUs. For example, for each four user license, two users cannot access S/REXX on one CPU while the other two users access S/REXX on another CPU. For each license, all four users must share one CPU.

Floating license option: The floating license allows the user to install SEDIT on one CPU, but they may access it on any one CPU within the network at a time (i.e., the license "floats" from one CPU to another). The CPUs must be connected to a network utilizing the Networked File System (NFS). A license server assures that the user is only running a single license of SEDIT on one CPU at a time. Once one user logs into SEDIT on one CPU, the entire four-user license "floats" to that CPU and the remaining three users can only access SEDIT on that specific CPU.

S/REXX Is Closely Integrated With SEDIT, the UNIX Text Editor

S/REXX supports the use of any UNIX and Windows editor, including vi, emacs, etc. S/REXX is integrated with SEDIT, a UNIX and Windows editor which emulates the mainframe XEDIT and

ISPF/PDF editors. SEDIT offers both character-based and GUI editing modes. The example screen below shows the SEDIT GUI interface:



```
SEDIT @ C(m1) - dy_tar [U], dir: /home/xed
File Edit Special Compile Flist Tree Help
/home/xed/demo/demo_dy/dy_tar Len:81 mod:
dy_tar
00022 $quit = dy_button(1, 1, "Quit")
00023
00024 call dy_label 14, 1.3, 'From Directory:'
00025 $dir = dy_input(30, 1.3, 28)
00026
00027 $tape = dy_tg('Tape Selection:', 14, 5, 32, 4.5, "nrst4", 1, 40, 4.5, "nrst5",
0)
00028
00029
00030 $start = dy_button(26.5, 10, "Start TAR")
00031
00032 $handle_dy_ex = dy_end("proc", 2, 3.5, 61, 12.5, "n")
00033
00034 call dy_map $handle_dy_ex
00035 exit
00036
00037 proc:
00038   parse arg handle reason
00039
00040   select
00041     when reason = 'F1' | reason = 'B'$quit | reason = '^c' then
00042       { call dy_unmap(handle)
00043         exit 0
00044       }
00045
00046     when reason = 'B'$start | reason = 'return' then
00047       { if do_it() = 0 then exit 0
00048       }
00049
00050     otherwise return 0
00051   end
00052 return
00053
1:Q 2:E 3:Save 4:Sp 5:x 6:cu 7:U 8:D 9:? 10:h 12:= S-11:top S-12:bot
```

The combination of S/REXX and SEDIT results in a powerful development system, which provides a very familiar environment to former mainframe programmers.

S/REXX can be used to write macros for SEDIT, just as REXX can be used to write macros for XEDIT or ISPF/PDF on the mainframe. In many cases, users may find that S/REXX runs former mainframe macros with little or no modification.

SEDIT is also available from Treehouse Software. Contact your Treehouse Software representative for more information.

Conclusion

The trend among mainframe sites of moving to UNIX or Windows or of maintaining combined mainframe/UNIX/Windows environments often results in decreased productivity, retraining costs, and time delays. S/REXX can help to alleviate all of these difficulties.

Because S/REXX is compatible with the instructions and functions of mainframe REXX, it allows users who are accustomed to REXX to become productive immediately in many UNIX and Windows application development and system administration functions.

S/REXX also offers many complementary facilities that provide greater ease of use and flexibility by taking advantage of the UNIX and Windows environment. These additional S/REXX facilities ensure that users will increase their productivity on the UNIX and Windows platforms.

In addition, the separately-priced S/REXX Debugger enables users to quickly locate and fix S/REXX program bugs.

XEDIT, REXX, VM/CMS, AIX, and ISPF/PDF are products of IBM. Windows, Windows NT and Windows 95 are trademarks of Microsoft Corporation. UNIX is a trademark of X/OPEN Company LTD. IRIX is a product of Silicon Graphics. HP/UX is a product of Hewlett-Packard. SCO UNIX is a product of Santa Cruz Operation. DEC UNIX is a product of Digital Equipment Corporation. SunOS, and Solaris are trademarks of Sun Microsystems, Inc. Unixware and OpenLook are trademarks of Novell. S/REXX, S/REXX Debugger, and SEDIT are a registered trademarks of the Benaroya Company, and are marketed exclusively in the U.S. and Canada by Treehouse Software, Inc. Any other products or trademarks mentioned herein are the property of their respective holders.

This page intentionally left blank.



Treehouse Software, Inc.

TREEHOUSE SOFTWARE INC.

2605 Nicholson Road, Suite 230

Sewickley, PA 15143

Phone: 724.759.7070

Fax: 724.759.7067

E-mail: tsi@treehouse.com

<http://www.treehouse.com>