REXXTACY: The REXX to C Translator

Anthony T. Green
Ruddock and Associates, Inc.

June 9, 1992

1 Introduction

1.1 REXXTACY - The REXX to C translator

REXXTACY is a package which allows you to convert a REXX program into a DOS or OS/2 executable. It does this by first translating the REXX source into C code. You must then invoke the Microsoft C compiler, and link the resulting object file with the REXXTACY runtime library.

The process of converting a REXX program to into C code consists of invoking the REXXTACY program with your REXX source file as an argument. REXXTACY will analyse the REXX source (`myprog.cmd`), perform some optimizations on the code, and emit a C source file (`myprog.c`). This C source is ready to compile with the Microsoft C compiler.

1.2 Using this manual

The organization of this manual has been loosly based on IBM's OS/2 REXX reference manual. Most sections of that manual have corresponding sections in this document. This will make it easier for the user who wishes to find any implementation differences between particular commands or control structures.

Since there are only a small number of differences or omissions from REXX in REXXTACY, the manual will seem rather sparse at points. Please be aware that this manual is not intended to be a REXX reference manual - but rather a description of the REXXTACY implementation.

1.3 Using REXXTACY

REXXTACY has a number of command line options. Run `REXXTACY -?` for usage information...

```
REXXTACY 2.0 - (C) 1992 Ruddock and Associates.  All rights reserved.

Options -o      Optimize REXX program
        -p      Debug parser.
        -z      Debug peephole optimizer.
        -u      Debug du & ud chain generation.
        -g      Debug constant and copy propagations.
        -b      Dump basic block information.
        -h      Display this message.
        -?      Display this message.
```

C files created with REXXTACY have the same name as the REXX file name, but ending in `.c`.

Most of the other switches are for debugging (and entertainment) purposes only. They tell REXXTACY to emit information pertaining to REXXTACY's internal optimization efforts. If ever you encounter a bug in REXXTACY you should run it again with all of these switches on, and pipe the output to a file. This file is very useful in helping the author track down problems.

Many errors in a REXX program will be caught at compile time by REXXTACY. However, REXXTACY provides extensive runtime error checks. A compiled REXX program processing faulty data will emit descriptive error messages — with line numbers that refer back to the original REXX source file.

2 General Concepts

2.1 Structure and General Syntax

REXXTACY input can be structured exactly like REXX input. It supports implicit and explicit end of statement markers, statement continuation markers and C style comments. However, hexadecimal and binary strings are not supported at this time.

2.2 Expressions and Operators

All of REXX's logical, arithmentic and comparison expressions and operators are supported by REXXTACY — including string abuttals and operator precedence orders.

2.3 Clauses and Instructions

REXXTACY supports all five REXX clause types:

1. NULL Clauses
2. Labels
3. Assignments
4. Keyword Instructions
5. Commands

2.4 Assignments and Symbols

Constant symbols, simple symbols, compound symbols and stems are all supported by REXXTACY. Assignments to these symbol types operate exactly as they do in REXX. One limitation of the current version of REXXTACY is that REXX keywords are are always recognized as keywords, independent of the lexical context in which they appear. This means that, for instance, you may not have a variable called `SAY`.

2.5 Commands to External Environments

The only command processing environment that REXXTACY currently supports is CMD (for OS/2) and COMMAND (for DOS). Commands to this environment may be issued exactly as they are in REXX, however, they are executed in a subcommand shell rather than as a callback to the parent shell (as in the case of OS/2).

3 Keyword Instructions

3.1 `ADDRESS`

Not implemented. All commands are routed to the OS/2 or DOS command shell.

3.2 `ARG`

Implementation identical to that of REXX.

3.3 CALL

Implementation identical to that of REXX, with the omission of:

- CALL ON
- CALL OFF

3.4 DO

Implementation identical to that of REXX.

3.5 DROP

Not implemented.

3.6 EXIT

Implementation identical to that of REXX.

3.7 IF

Implementation identical to that of REXX.

3.8 INTERPRET

Not implemented.

3.9 ITERATE

Not implemented.

3.10 LEAVE

Not implemented.

3.11 NOP

Implementation identical to that of REXX.

3.12 NUMERIC

Not implemented.

3.13 OPTIONS

Not implemented.

3.14 PARSE

Implementation identical to that of REXX except for the omissions of `PARSE SOURCE`.

3.15 PROCEDURE

Implementation identical to that of REXX.

3.16 PULL

Implementation identical to that of REXX.

3.17 PUSH

Not implemented.

3.18 QUEUE

Not implemented.

3.19 RETURN

Implementation identical to that of REXX.

3.20 SAY

Implementation identical to that of REXX.

3.21 SELECT

Implementation identical to that of REXX.

3.22 SIGNAL

Not implemented.

3.23 TRACE

Not implemented.

4 Functions

Function calls in REXXTACY operate exactly as they do in REXX. Built in functions that are supported in REXXTACY include:

- `LENGTH`
- `LINEIN` (No support for device names, line or count parameters)
- `LINES` (No support for device names)