IBM大学合作项目书籍出版资助

教育部–IBM高校合作项目
精品课程系列教材

大型主机系统管理
REXX编程详解

高珍 主编

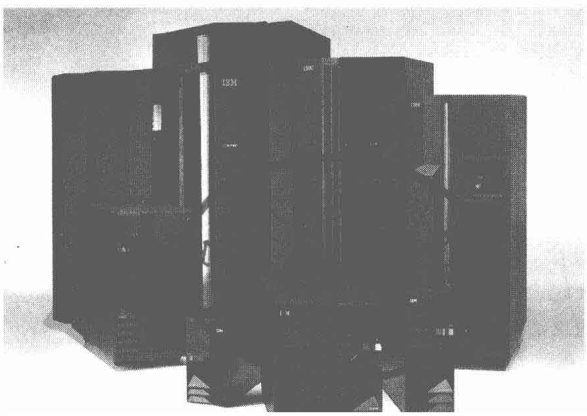
刘恒 王天琦 张润芸 庄焕焕 编著

清华大学出版社

IBM 大学合作项目书籍出版资助
教育部—IBM 高校合作项目精品课程系列教材

大型主机系统管理
REXX 编程详解

高　珍 主编
刘　恒　王天琦　张润芸　庄焕焕 编著



清华大学出版社
北　京

内 容 简 介

本书详细介绍了 REXX 的语法、函数和子例程的使用、REXX 与关键子系统的交互、REXX 程序的执行和调试等，并辅以案例和实验，是大型主机专业系统管理方向的重要教材，全书共分 10 章，是一本理论与实践并重的教材。

本书的第 1 章介绍大型主机上常用的几种脚本语言；第 2 章阐述主机脚本语言 REXX 的由来、发展、特点和组成等内容；第 3 章详细介绍 REXX 的基本语法，包括指令、表达式、程序控制流等；第 4 章介绍函数和子例程的使用；第 5 章重点阐述 REXX 数据处理技术，包括 REXX 强大的数据解析功能、数据栈和文件操作；第 6 章介绍 REXX 与主机子系统或工具的交互，包括 TSO、MVS 控制台、JES、ISPF 和 USS 等；第 7 章介绍了 REXX 与 ISPF 的交互；第 8 章介绍 REXX 程序的执行方式；第 9 章介绍 REXX 程序的调试技术；第 10 章介绍 3 个综合案例；第 11 章针对 REXX 主要知识点设计了 REXX 实验，并附有实验答案。

本书可以作为高等院校计算机学院、软件学院有关大型主机课程的教材，也可以作为从事大型主机工作的相关技术人员，尤其是系统管理人员的自学书籍，也可供希望学习和了解 REXX 编程技术的人员参考。

图书在版编目（CIP）数据

大型主机系统管理 REXX 编程详解/高珍主编；刘恒等编著. —北京：清华大学出版社，2012.5
ISBN 978-7-302-28005-7

Ⅰ. ①大… Ⅱ. ①高… ②刘… Ⅲ. ①大型计算机—程序设计 Ⅳ. ①TP338.4

中国版本图书馆 CIP 数据核字(2012)第 019875 号

责任编辑：龙启铭　顾　冰
封面设计：傅瑞学
责任校对：白　蕾
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张**：14　　**字　数**：324 千字
版　　次：2012 年 5 月第 1 版　　**印　次**：2012 年 5 月第1 次印刷
印　　数：1～3000
定　　价：29.00 元

产品编号：043362-01

FOREWORD 前言

对于系统管理员而言，掌握脚本编程技术的重要性不言而喻。不管如今系统架构如何庞大和复杂，系统管理维护的很多日常的基础性工作还是完全可以程式化、机械性地定义和描述，并通过机器加以执行完成的。系统脚本编程不仅仅是将管理员从重复的键盘敲击和命令输入中解放出来，更重要的是最大程度地避免了重复性工作所可能带来的“疲劳性”错误，对于拥有最高系统权限的管理员而言，这些错误很可能是灾难性的。而另一方面，各种脚本语言也从简单的“命令执行序列”发展成为了更加精巧完善且易于掌握的快速开发语言，吸引了更为广泛的关注，并且延伸和扩展到了更为高级和复杂的应用之中。

如同 Perl 之于 UNIX/Linux 一样，对于大型机操作系统平台 z/OS 而言，REXX 以其与 z/OS 系统组件的广泛交互支持的特点、强大实用的文本处理功能以及极其简单灵活的结构化语法，成为大型机平台上使用最为流行的一种脚本语言。熟练掌握 REXX 的语言特性能极大提升大型机系统管理的效率和质量，也为各种进一步的复杂应用与扩展提供了可能性。本书作者在实际工作中发现，REXX 语言相关资料颇为繁杂，且由于其语言本身的灵活性，相关的应用描述和技术实践分散于不同的 IBM 官方发布的相关文档章节以及开发交流社区的文章之中，加之有中文详述的提及更是少之又少，造成了一些实践工作中学习的不便。本书作为第一本系统讲述大型机 REXX 编程实践的中文图书，旨在通过汇编总结相关文档中与日常应用最为紧密的知识点，并结合作者大量的实际应用经验和体会，力图使读者对 REXX 语言编程有一个直观全面的认识，为更加深入钻研 REXX 编程技术以及大型机系统管理的复杂扩展应用打下基础。

本书在编写、完稿至出版期间一直得到 IBM 公司大学合作部的大力支持，特别是李晶辉经理和万泽春经理对本书出版的支持和肯定；黄小平高级工程师对本书内容的编写、定稿都提出了宝贵的指导性意见；同济大学软件学院为本书的出版提供了有力的支持；软件学院的很多学生包括但不局限于郭意亮、张璇华、冯学奋、刘樽鹤和瞿苑等都参与了本书的文字校稿工作；本书还获得“IBM 大学合作项目书籍出版资助计划”的资助，并获得“同济大学十二五规划教材”支持，在此一并表示衷心的感谢。

本书遵循由浅入深的原则，采用了一般计算机编程语言讲解的通用逻辑顺序，大致可分为概括介绍、语法与程序规范、子系统交互以及执行/调试/实例分析四大部分，全书穿插了常用知识点与实例分析，希望对读者有所启迪。在成书过程中作者虽查阅了大量文献、编写调试实例程序验证力图严谨，但难免有所疏漏，望各位业内同仁以及读者予以斧正。

作　者

2012 年 2 月于同济大学

CONTENTS 目录

第1章

主机脚本语言概述

脚本语言（Script Programming Language）是为了缩短传统的编写—编译—链接—运行（Edit-Compile-Link-Run）过程而设计的计算机编程语言。各种不同的计算机平台中存在很多可用的“组件”，人们开发了对应的系统脚本语言，使之能够快速组合这些系统平台中可用的“组件”，从而完成特定需求的脚本开发。

在设计一个应用程序时，用户可能需要考虑选择编译型语言或者解释型语言来编写程序源代码，而这两种类型的语言都有各自的优缺点。通常，确定使用解释型语言是缘于开发时间的限制或者未来程序的修改方便的考虑；而在确定使用解释型语言的同时也付出了代价：即系统需要以更高的执行开销换取了开发速度。这是由于在每次执行时，解释型语言的每一行源码需要被重新翻译解释，对系统而言意味着巨大的开销。因此，解释型语言通常更加适合于即兴需求，而不是那些预定需求。

脚本语言和一般程序设计语言的一个重要不同在于脚本语言通常是被解释执行，而系统程序设计语言通常是被编译执行。解释型语言由于不需要编译时间，而是通过快速的转换，使用户能快捷地编写并运行程序。由于这种特性，在系统管理以及测试中，程序员们编写了各种脚本来执行例行操作或者一次性任务。而脚本程序因为其灵活、简单、易于开发等特点，在主机系统管理领域有着广泛的应用。例如，系统管理员可能需要执行一些常规并且确定性的任务，例如输入相应命令、提交某个作业、检查数据集状态、为特定的程序分配数据集、打印文件等，通过编写相应的系统脚本，将这些操作所需的交互命令和任务组织成脚本文件的形式，能显著减少在这些例行任务上所花费的时间。将这些执行例行任务所需要的所有指令或操作集合到一个系统脚本程序中，可以节约大量人工操作时间、降低任务执行的错误概率、增强功能可复用性，并能极大提升系统管理工作的效率。z/OS 中所提及的系统脚本语言主要包括 CLIST、REXX 和 USS 中的 SHELL，后面将分别予以介绍。

1.1 CLIST 语言简介

CLIST 语言是一种解释型语言。和其他解释型语言编写的程序类似，CLIST 程序也易于编写和测试。术语 CLIST（读作 See List）表示“命令(Command)的序列(List)”，这是由于大多数的基本 CLIST 程序都是 TSO/E 的命令的序列。当用户调用一个这样的

CLIST 程序时，它将按顺序调用对应的 TSO/E 命令。

CLIST 编程语言主要用于以下几个方面。

- 执行日常的例行任务（例如输入 TSO/E 命令）。
- 调用其他的 CLIST 程序。
- 调用其他语言编写的应用程序。
- ISPF 应用程序（例如显示面板以及控制应用程序流）。

一个 CLIST 程序可以执行范围广泛的各种任务，主要可以归为以下三类。

- 执行日常例行任务的 CLIST。
- 作为结构化应用程序的 CLIST。
- 管理其他语言编写的应用程序的 CLIST。

1. 执行日常例行任务的 CLIST

用户可以编写相应的 CLIST 程序，它们能显著减少用户在这些日常的例行任务上所花费的时间。通过将执行任务所需要的所有指令集合到一个 CLIST 程序中，用户可以减少花费时间、键盘敲击次数以及任务执行中的错误，并且提升用户的工作效率。一个 CLIST 程序由单纯的 TSO/E 命令组成，或者由 TSO/E 命令与 CLIST 语句所混合组成。

2. 作为结构化应用程序的 CLIST

由于 CLIST 语言包括了编写一个完整的、结构化的应用程序的基本工具。任何 CLIST 都可以调用另外的 CLIST 程序，并作为一个“嵌套的”CLIST 程序引用。CLIST 同样包含了独立的子程序，称之为“子过程”。嵌套的 CLIST 程序和子过程使用户能够将 CLIST 程序划分成逻辑单元，并将公共功能在一处实现。

在交互式应用程序中，CLIST 可以调用命令显示 ISPF 面板。反之，ISPF 面板也可以根据用户在面板上的输入，来调用对应的 CLIST 程序。

3. 管理其他语言编写的应用程序的 CLIST

假设用户需要访问用其他语言编写的应用程序，但这些应用程序的相关接口可能不易使用或记忆，需要编写 CLIST 程序在用户和应用程序之间提供一个易于使用的接口，这要好过重新编写一个新的应用程序。一个 CLIST 程序可从终端上接收消息并通过程序逻辑判断用户需求。CLIST 程序可以设置环境并执行命令来调用程序，完成所相应的任务请求。

4. CLIST 语言的其他应用以及特色

除了请求执行 TSO/E 命令，CLIST 程序也可以执行更为复杂的编程任务。CLIST 语言包括了用户在开发大的结构化应用程序时所需用到的编程工具。CLIST 程序可以执行任何复杂度的任务，从显示一系列全屏面板到管理其他语言所编写的应用程序。

CLIST 语言特色如下。

- 具有处理数值数据所需的广泛的算术与逻辑操作符集。
- 具有很强的处理字符数据的字符串处理功能。
- CLIST 语句能很好地用于构建程序、执行 I/O 操作、定义和修改变量、进行错误

处理以及警示中断。

5. 如何执行 CLIST 程序

要执行一个 CLIST 程序，可使用 EXEC 命令。在 ISPF 的命令行中，命令开头需要输入 TSO。在 TSO/E EDIT 或者 TEST 模式下，在执行 EXEC 操作时使用 EXEC 子命令（执行在 EDIT 或者 TEST 模式下的 CLIST 程序只能够请求执行对应的 EDIT 或者 TEST 的子命令和 CLIST 语句，但用户可以使用 END 子命令以结束 EDIT 或者 TEST 模式，使 CLIST 程序能够请求执行 TSO/E 命令）。

1.2　REXX 语言简介

REXX（REstructured eXtended eXecutor）即重构的可扩充执行器语言，是 IBM 在 20 世纪 80 年代所发明的一种程序设计语言。主要用于 IBM 的主机系统，但在大部分其他平台上也可以找到对应解释器或编译器。REXX 是 IBM 随其大型主机、中型机操作系统和其他更低端操作系统一起捆绑的脚本和命令语言。REXX 几乎可以在世界上任何操作系统上运行。用户可以免费下载用于各版本的 Windows、Linux、UNIX、BSD、Mac OS 和 DOS 以及很多其他系统的 REXX。它甚至可以在用于手持设备的三大主流操作系统 Windows CE、Palm OS 和 Symbian/EPOC32 上运行。

REXX 也可用在 Java 环境中，例如 REXX 的一个分支叫做 NetRexx 能够与 Java 完全无缝的协同工作。NetREXX 程序可以直接使用任何 Java 类，并可以用来编写任何 Java 类。它将 Java 的安全性和性能特点带给了 REXX 程序，并将 REXX 的算术运算和简单性的特点带给了 Java。而作为单一语言的 NetREXX，可以同时用于脚本开发和应用程序开发。

1.2.1　一个简单的 REXX 程序

下面举例说明 REXX 语言的易学和强大。例如计算 1+2+…+10，然后把这些数字和所得的和显示出来。REXX 有强大的字符和数字处理功能，可以轻松实现。在这里也可以看出 REXX 程序很容易读懂。

```
/* REXX */
/* Count to ten and add the numbers up */
sum = 0
do count = 1 to 10
   say count
   sum = sum + count
end
say "The sum of these numbers is" sum"."
```

1.2.2 REXX 语言的一些不足

本章介绍 REXX 程序中存在的一些不足。

1. 难以维护

REXX 作为一种结构化语言，它使程序和算法能够以清晰和结构化的方式书写。但是，因为 REXX 的程序格式非常自由，变量的使用非常灵活，使得对不规范的 REXX 程序难以维护。所以在编写 REXX 程序时，应该尽量在开始部分写清楚程序的作用及主要的算法，在关键语句和变量处加以注释说明。

2. 执行效率不高

REXX 是一种解释型和编译型语言。正如前面提到的那样，这种解释型语言和其他诸如 COBOL 语言的不同之处在于，在它执行之前不需要编译 REXX 源程序。也正是因为 REXX 的这一特性，导致源程序在运行时才被逐句解释执行，效率比编译执行的语言要差。但是用户可以选择在执行之前编译 REXX 源程序生成可执行代码（Load Module），从而大大提高其执行速度。

1.2.3 REXX 语言的主要应用

REXX 编程语言的典型应用如下。

- 执行日常例行任务，例如输入 TSO/E 命令。
- 调用其他 REXX 程序。
- 调用其他语言编写的应用程序。
- ISPF 应用程序。
- 对问题的一次性快速解决方案。
- 系统编程。

一般而言，REXX 可以方便地对主机文件进行读写等操作。如果有大量的主机文件需要手工查找、修改等并且这些操作都具有共性，不妨使用 REXX 帮忙解决。例如，用户想将一个程序库中所有使用了某一些子程序的主程序全部找来；就可以用 REXX 解决；如果让用户对所有程序都加上一行一模一样的句子，这样手工修改起来势必会很麻烦，而且工作量也非常庞大，就可以考虑写一个 REXX 程序；如果用户想在一个程序中访问两个数据库，在 COBOL 中实现起来相对麻烦，而 REXX 就显得方便很多；还有 REXX 可以通过调用 SDSF 服务查找打印用户需要的作业信息，通过调用系统命令显示或修改系统配置等。后面将详细介绍关于 REXX 语言语法和应用的有关内容。

1.3 USS 的 Shell 简介

USS（UNIX System Service）所提供的 Shell 功能，顾名思义，就是 z/OS UNIX 交互接口所提供的面向 UNIX 命令和 Shell 语言的解释器。所支持的 Shell 脚本语言同标准

UNIX 的类似，z/OS UNIX 有两个版本的 Shell，即 z/OS Shell 和 tcsh Shell，它们统称 z/OS UNIX Shell。

z/OS Shell 基于 UNIX System V Shell，其中部分功能源自著名的 UNIX Korn Shell。该 Shell 遵守 POSIX 标准 1003.2，并被采用为 ISO/IEC International Standard 9945-2: 1992。同时，z/OS Shell 向上兼容 Bourne Shell。

tcsh Shell 完全兼容并加强了 Berkeley UNIX C Shell，csh 是一个通用的语言解释器，既可用于交互式 Shell，也可用作 Shell 脚本的解释器。

Shell 脚本语法简要介绍如下。

- Shell 命令的组合
- While 循环
- For 循环
- DO.....DONE
- IF...ELIF...ELSE....fi
- TEST 用于条件测试，下例中代码用于目录的测试

```
if
    test - d $ 1
then
    echo "$ 1 is a directory "
fi
```

可以看到，Shell 脚本中可以使用 z/OS UNIX Shell 命令、分支语句、变量以及环境设定等。同一般的 UNIX Shell 脚本类似，USS 的 Shell 程序就是一个包含若干行 Shell 或 z/OS UNIX 命令，依据 Shell 的语法规则所写成的文件。解释器将 Shell 命令以脚本形式书写和存储，并以命令序列的方式在 Shell 下解释执行。如果用户熟悉其他平台下的 Shell 脚本书写，相信对 USS 下的 Shell 应该能够很快掌握运用。下面是一个 Shell 脚本的示例，该例中脚本程序检测某个 Java 类（java_memory.class）是否存在，如果存在，则执行该类；若不存在，脚本检测该类的源代码（java_memory.java）是否存在，如果存在，则将源码编译后执行。

```
if
  test -f  java_memory.class
then
  echo "the Java program is present. Let's execute it now"
  java  java_memory
elif
  test -f  java_memory.java
  then
    echo "the Java source is there . Let's compile it first!"
    javac java_memory.java
    echo "Now we will execute the Java code."
    java  java_memory
```

```
    else
      echo " there is no Java source present to compile"
fi
```

z/OS 的 Shell 编程环境以及所支持的 Shell 脚本，与 TSO/E 环境以及所支持的 CLIST 和 REXX 脚本类似，都是将完成特定任务或者一组任务所对应的命令或者所调用的相关过程，组织成脚本程序的形式，并由对应的环境解释执行，提高与环境交互工作的效率。

第2章

REXX 简介

REXX 可以用于多种平台之上，这是由 REXX 本身设计的特点所决定的。REXX 以约 20 条指令作为一个小的核心。这个核心周围有大约 70 个内置函数。这种设计使得 REXX 本身可以得到最大程度的扩展和延伸，使其可以使用包括数据库访问、GUI、XML、Web 服务器编程、MIDI 接口在内的各种形式的接口。这些易扩展性不仅使用户可以快速开发出简易程序，也可以使 REXX 用于持久健壮性程序的开发。事实上，借助特定的跨平台接口，REXX 脚本在 Linux、Windows、大型主机、手持设备之间具有广泛的可移植性。REXX 内核结构如图 2-1 所示。

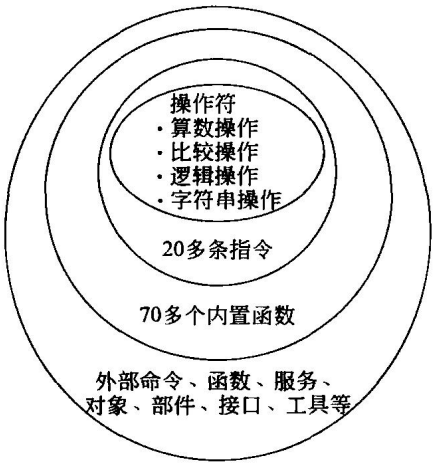


图 2-1　REXX 内核结构[①]

本章将详细介绍有关 REXX 语言本身的功能和使用，其中对语言语法规范的介绍是与具体平台实现无关的通用性内容。

2.1　REXX 的发展历史

REXX 最初创建于 1979 年，由当时 IBM 英国实验室的工程师 Mike Cowlishaw 设计，

① 摘自 http://www.ibm.com/developerworks/cn/db2/library/techarticles/dm-0508fosdick/index.html