第一部分

ILE RPG/400 简介

在使用 ILE RPG/400 建立程序之前， 你必须对 ILE RPG/400 运行环境的某些特征有所了解。本部分就应该了解的以下方面提供信息：

- RPGⅣ语言概述
- 在 RPG 编程中 ILE 各组成部分
- ILE 程序建立策略

第一章　　RPGⅣ程序设计语言概述

本章在一个高层次上对RPGⅣ编程语言与其他RPG编程语言的区别作了介绍。在用RPGⅣ语言编程之前，必须熟悉并适应所有的这些特性。这里所讨论的特性是围绕以下几个方面的：

编写规范表
程序周期
指示器
操作码

关于RPGⅣ的更多信息，参见 ILE RPG/400 参考手册。

1.1　　RPGⅣ规范表

RPG 编码是在一些规范表的格式中写的，每种规范表都有一系列特定的功能。规范表是由许多项组成的，这些项与位置有关。根据项的类型及规范表的类型，每个项必须从一定的位置开始。

有六种类型的 RPG 规范表。每种规范表都是可选的。这些规范表必须按照以下的顺序输入到源程序中：

1、控制规范表，为编译程序提供生成和运行所需信息，例如程序名，日期格式，使用交替对照序列或文件翻译。
2、文件描述规范表，描述了程序中使用的所有文件。
3、定义规范表，描述了程序使用的数据。
4、输入规范表，描述了程序使用的输入记录和字段。
5、计算规范表，描述了对数据所做的计算及计算顺序。计算规范表也可以控制某些输入和输出操作。
6、输出规范表，定义了程序使用的输出记录及字段。

1.2　　编程周期

当系统处理数据时，它必须按一定的顺序来进行，这个逻辑顺序是由以下几方面提供的：

RPG/400 编译程序
程序代码

由编译程序提供的逻辑叫做程序周期。当让编译程序为你的程序提供逻辑时，就叫做编程周期。

程序周期是程序在文件结束前所重复的一系列步骤。根据编写的规范表，程序可能会执行或跳过周期中的某一步。

如果你想用程序周期来控制文件，就不用在源程序里指定从文件中读取记录的相关信息。编译程序会为这些操作及某些输出操作提供逻辑顺序。

如果不想让该周期来控制文件，那么你必须建立文件结束条件，通常是设置最后记录指示器（LR）为 ON。

图 1 显示了在一般的 RPG 程序周期流程中的一些步骤

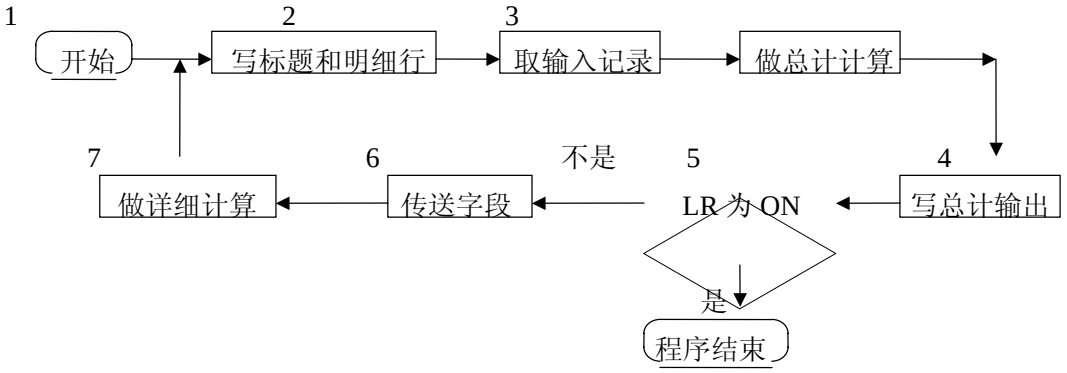


图 1　RPG 程序逻辑周期

1、RPG 处理所有的标题行和明细行（在输出规范表中 17 例为 H 或 D）。

2、RPG 读下一个记录并置记录标识指示器和控制级指示器为 ON。

3、RPG 处理总计计算（由控制级指示器 L1 到 L9，或 LR 指示器或 LO 项决定）。

4、RPG 处理所有的总计输出行（由输出规范表第 17 例是 T 标识）。

5、RPG 判断 LR 指示器是否为 ON。如果为 ON，则程序结束。

6、被选择的输入记录字段由记录转移到处理区，RPG 置字段指示器为 ON。

7、RPG 处理所有的明细计算（该计算规范表 7-8 列上没有控制级指示条件限制的），它使用周期开始时记录中的数据。

1.2.1　第一周期

通过程序周期的第一次和最后一次与其它周期有所不同，在第一次通过程序周期，读取第一条记录之前，程序要做三件事：

处理输入参数、打开文件、初始化程序数据。

写出由 1P 指示器（第一页指示器）条件限制的那些记录；

处理所有的标题和明细输出操作。

例如，在读第一条记录前，打印的标题行可能包括常量或页标题信息，或象 PAGE 和*DATE 这样的特殊字段，在第一周期，程序还跳过总计计算和总计输出步骤。

1.2.2 最后周期

程序通过周期的最后一次，这时，已没有可用记录，程序置 LR（最后记录）指示器和 L1 到 L9（控制级）指示器为 ON，程序处理所有的总计计算和总计输出，之后关闭所有文件，程序结束。

1.3 指示器

指示器是取值‘1’或‘0’的一个字节长的字符型字段。‘1’为 ON，‘0’为 OFF。每个指标器都有一个由两个字符构成的名字（例如：LR，01，H3），并且在某些规范表的某些项中仅以这个名字来引用，在另外的一些规范的某些项中用特殊的名字*INxx 来引用，其中 xx 为两个字符的名字。

指示器可以是操作的结果或用来决定（或控制）操作是否执行，指示器就好比程序逻辑流程的开关，根据它们的状态，决定程序运行期间应走的路径。

你可以使用几种不同类型的指示器，每一种表示不同事件。在 RPG 程序中，指示器可以在规范表中用设置项的方法定义，或由程序周期本身定义。在规范表中定义指示器的位置就决定了指示器的用途。一旦在程序中定义了指示器，它就可以限制或控制计算或输出操作。

RPG 程序在程序周期的特定时间设置或重设某些指示器。另外，可以在计算操作中明确地修改指示器的状态。

1.4 操作码

RPGⅣ编程语言允许对数据实施不同类型的操作。计算规范表中的操作码指出了要进行何种操作。例如，要读一个新记录，可以使用 READ 操作码。以下是可用的操作类型的列表。

算术操作	指示器设置操作
数组操作	信息操作
位操作	初始化操作
分枝操作	消息操作
调用操作	传递操作
比较操作	串操作
数据区操作	结构化编程操作
日期/时间/时间标识操作	子程序操作
说明操作	检测操作
文件操作	

1.5 ILE RPG/400 程序的例子

这部分用图讲解了一个使用 ILE RPG/400 程序处理工资单计算的例子。

1.5.1 问题提出

一个公司的工资部门要打印出某周公司雇员的工资清单，假设在系统中有两个磁盘文件，EMPLOYEE 和 TRANSACT。

第一个文件，EMPLOYEE，存储雇员记录，下图显示了雇员记录的格式。

EMP_REC

EMP_NUMBER	EMP_NAME	* EMP_RATE *
* *	* *	* *
1 6	22	27

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
A.........T.Name++++++RLen++TDpB......Functions++++++++++++++++++++*
A           R EMP_REC
A             EMP_NUMBER      5           TEXT('EMPLOYEE NUMBER')
A             EMP_NAME       16           TEXT('EXPLOYEE NAME')
A             EMP_RATE        5  2        TEXT('EXPLOYEE RATE')
A           K EMP_NUMBER
```

图 2 雇员物理文件的 DDS 源语句

第二个文件，TRANSACT，记录雇员们工作的时间和得到的奖金，下图显示了交易记录的格式。

TRN_REC

* TRN_NUMBER *	* TRN_HOURS *	* TRN_BONUS *
* *	* *	* *
1 6	10	16

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
A.........T.Name++++++RLen++TDpB......Functions++++++++++++++++++++*
A           R TRN_REC
```

```
A            TRN_NUMBER     5          TEXT('EMPLOYEE NUMBER')
A            TRN_HOURS      4  1       TEXT('HOURS WORKED')
A            TRN_BONUS      6  2       TEXT('BONUS')
```

图 3 物理文件 TRANSACT 的 DDS 源语句

每个雇员的工资算法是小时数“hours”（取自 TRASACT 文件）和比率“rate”（取自 EMPLOYEE）的乘积并加上取自 TRANSACT 文件的奖金“bonus”。

1.5.2 控制规范表

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
HKeywords++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
H DATEDIT(*DMY/)
```

当天的日期将以日，月，年的格式打印，并以“/”为分隔符。

1.5.3 文件描述规范表

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+...
FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords+++++++++++++++++++++++++++++

FTRANSACT  IP   E           K DISK
FEMPLOYEE  IF   E           K DISK
FQSYSPRT   O    F   80        PRINTER
```

在文件描述规范表中定义了三个文件。

TRANSACT 定义为输入文件，由 ILE RPG/400 程序周期控制从这个文件中读记录。

EMPLOYEE 定义为输入的全过程文件，由计算规范表控制从这个文件中读记录。

QSYSPRT 定义为输出的打印文件。

1.5.4 定义规范表

```
*..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+...
D+Name++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++

D Pay             S              8P 2
D Heading1        C                   'NUMBER  NAME            RATE    H-
D                                     OURS  BONUS    PAY       '
D Heading2        C                   '________ _____ _____ _________  _-
D                                     ____  ____ ___  __________'
```

使用定义规范表，说明变量“Pay”储存雇员每周的工资，两个常量“Heading1”和“HEADING2”来做打印表头。

1.5.5 计算规范表

```
*..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+...
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq..

C     TRN_NUMBER    CHAIN     EMP_REC                          99
C                   IF        NOT *IN99
C                   EVAL (H)  Pay = EMP_RATE * TRN_HOURS + TRN_BONUS
C                   ENDIF
```

计算表的代码项包括：

CHAIN 操作，用交易文件中的字段 TRN_NUMBER 在雇员文件中查找有相同雇员号的记录。

如果 CHAIN 操作成功。（指示器 99 为 OFF），则雇员的工资值确定，结果被四舍五入并保存于变量“pay”中。

1.5.6 输出规范表

```
*..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+...
OFilename++DF..N01N02N03Excnam++++B++A++Sb+Sa+.....................
O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat

OQSYSPRT    H     1P                    2  3
```

```
O                                              35 'PAYROLL REGISTER'
O                        *DATE          Y      60
O          H    1P                      2
O                                              60 Heading1
O          H    1P                      2
O                                              60 Heading2
O          D   N1PN99                   2
O                        TRN_NUMBER             5
O                        EMP_NAME              24
O                        EMP_RATE       L      33
O                        TRN_HOURS      L      40
O                        TRN_BONUS      L      49
O                        Pay                   60 '$     0.  '
O          D   N1P 99                   2
O                        TRN_NUMBER             5
O                                              35 '** NOT ON EMPLOYEE FILE **'
O          T    LR
O                                              33 'END OF LISTING'
```

输出规范表描述了哪些字段要输出到 QSYSPRT 中。

如果 1P 指示器为 ON，那么包含字符串常量‘PAYROLL REGISTER’的标题行及明细信息的标题就会被打印出来。1P 指示器是由 ILE RPG/400 程序在第一程序周期置为 ON 的。

明细行的打印由 1P 指示器和 99 指示器决定，1P 指示器为 ON 时，明细行不打印。当找到对应的雇员记录时，99 指示器为 OFF，则打印明细行。如果 99 指示器为 ON，那么打印雇员号和字符串常量‘** NOT ON EMPLOYEE FILE**’。

总计行包含字符串常量‘END OF LISTING’。它在程序的最后一次周期时打印。

1.5.7 完整的源程序

下图包含了这个程序中使用的所有规范表，也就是应当输入的源程序。

```
*-------------------------------------------------------------------------*
     * 这个程序生成雇员每周工资的打印输出。                                *
*-------------------------------------------------------------------------*

    H DATEDIT(*DMY/)
```

```
*----------------------------------------------------------------------------*
      * 文件定义
*
*----------------------------------------------------------------------------*
     FTRANSACT  IP   E           K DISK
     FEMPLOYEE  IF   E           K DISK
     FQSYSPRT   O    F   80        PRINTER


*----------------------------------------------------------------------------*
      * 变量说明
*
*----------------------------------------------------------------------------*
     D Pay             S              8P 2


*----------------------------------------------------------------------------*
      * 常量说明
*
*----------------------------------------------------------------------------*
     D Heading1        C                   'NUMBER  NAME            RATE    H-
     D                                     OURS  BONUS    PAY       '
     D Heading2        C                   '______________________  ______  _-
     D                                     ____  _______  __________'


*----------------------------------------------------------------------------*
      * 对文件（TRANSACT）的每个记录。如果找到雇员，计算他的工资且打印详细  *
      * 信息。                                                      *
*----------------------------------------------------------------------------*
     C     TRN_NUMBER    CHAIN     EMP_REC                          99
     C                   IF        NOT *IN99
```

```
C                   EVAL (H)  Pay = EMP_RATE * TRN_HOURS + TRN_BONUS
C                   ENDIF


*-------------------------------------------------------------------------*
     * 报表格式:                                                          *
     *  -- 如果 1P 为 ON，打印标题行。                                     *
     *                                                                     *
     *  -- 如果找到记录，打印工资详细内容，否则打印例外记录。              *
     *  -- 当 LR 为 ON 时，打印'END OF LISTING'。                          *

----------------------------------------------------------------------------*
OQSYSPRT   H    1P                    2  3
O                                        35 'PAYROLL REGISTER'
O                        *DATE       Y   60
O          H    1P                    2
O                                        60 Heading1
O          H    1P                    2
O                                        60 Heading2
O          D   N1PN99                 2
O                        TRN_NUMBER       5
O                        EMP_NAME        24
O                        EMP_RATE    L   33
O                        TRN_HOURS   L   40
O                        TRN_BONUS   L   49
O                        Pay             60 '$     0.  '
O          D   N1P 99                 2
O                        TRN_NUMBER       5
O                                        35 '** NOT ON EMPLOYEE FILE **'
O          T    LR
O                                        33 'END OF LISTING'
```

图 4　工资单计算程序的一个例子

1.6　使用 OS/400 系统

控制用户和 AS/400 系统所有交互动作的操作系统叫 OS/400。在工作站上,OS/400 允许你做以下事情:

sign on 和 sign off 注册和注销
与显示工作站进行交互作业
使用联机帮助信息
输入控制命令和过程
对信息作出应答
管理文件
运行实用工具和程序
参考 SO/400 系统的资料 SC41—3000，那里给出了定购的资料内容。

1.6.1 与系统进行交互操作

你可以使用命令语言（CL）操纵 OS/400 系统。进入或选择 CL 命令就可以与系统进行交互式操作。AS/400 系统经常显示一系列的 CL 命令或适当的命令参数，然后由你选择想要的命令或参数。

1.6.1.1 常用的控制语言命令

下面列出了一些常用的 CL 命令以及它们功能和使用它们的原因。
表 1 常用的 CL 命令

操作	CL 命令	结果
使用系统菜单	GO MAIN	显示主菜单
	GO INFO	显示帮助菜单
	GO CMDRPG	列出有关 RPG 的命令
	GO CMDCRT	列出有关建立的命令
	GO CMDXXX	列出有关‘XXX’命令
调用	CALL 程序名	运行一个程序
编译	CRTxxxMOD	建立 xxx 模块
	CRTBNDxxx	建立连接的 xxx 程序
连接	CRTPGM	用 ILE 模块建立一个 ILE 程序
	CRTSRVPGM	建立一个服务程序
	UPDPGM	更新一个连接的程序目标
调试	STRDBG	启动 ILE 源语句调试程序
	ENDDBG	结束 ILE 源语句调试程序
建立文件	CRTPRTF	建立打印文件
	CRTPF	建立物理文件
	CRTSRCPF	建立源物理文件
	CRTLF	建立逻辑文件

1.7 AS/400 工具

AS/400 提供了有利于编程的全套工具。以下的产品可以帮助你更有效地开发 ILE RPG/400 应用程序。关于与这些产品有关的资料的信息，参见附录后的文献。

1.7.1 应用程序开发工具集/400（ADTS/400）

应用程序开发工具集/400（ADTS/400）提供了一套完整的主机环境工具。以适应应用程序开发者的需要。这套产品提供了对 AS/400 系统上源语句，目标及数据库文件管理的工具。其中包括 PDM、SEU 和 SDA。它提供了菜单驱动的界面，从这里你可以完成所有与应用程序开发相关的任务，例如目标管理、编辑、编译和调试。

1.7.2 应用程序开发管理程序/400

应用程序开发管理程序为应用程序的开发组织提供了高效的管理，也管理应用程序生存期的目标。这一性能使得一组开发人员可以通过编程开发管理程序（PDM）界面或直接通过命令来建立、管理和组织多个版本的应用程序。

1.7.3 协作开发环境/400

协作开发环境/400（CODE/400）加强了程序开发能力，并减少了主机上程序开发的负载。对于 RPG 应用程序的开发和维护。CODE/400 提供下面的工具。

语言灵敏度的编辑—包括标志高亮度，格式行，全套提示，和联机帮助。

增强语法检查—当录入语句时，对每一行的错误立即反馈。

程序校验—编译程序在没产生目标代码时就在工作站上进行全范围的语法和语义检查。

OS/2 界面，用以提交主机编译以及连接源语句级调试.

DDS 设计实用工具—让你方便的修改屏幕，报表和数据文件。

访问应用程序开发管理程序/400。

第二章 ILE 中的 RPG 编程

ILE RPG/400 是集成语言环境中的 RPGⅣ编程语言工具。它是 AS/400 系统上先进的 ILE 编译程序之一。

ILE 是 AS/400 系统上的一种新的编程方式。它是 AS/400 增强机器结构和 OS/400 操作系统功能的结果。ILE 编译程序家族包括：ILE RPG/400、ILE C/400*，ILE COBOL/400*，

和 ILE CL，图 5 描绘了操作系统支持 ILE 而做的改进。它说明了对原始程序模式（OPM）和扩展程序模式（EPM）语言的支持。

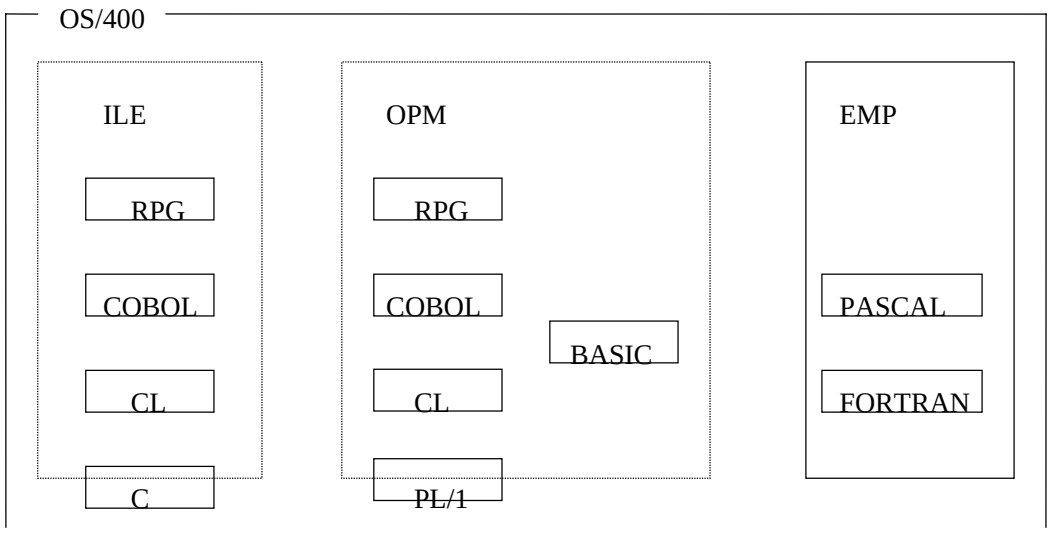


图 5　集成语言环境及其在操作系统中的地位

ILE 的 RPG 用户在以下应用开发领域实现了改进和加强。

程序建立

程序管理

程序调用

源语句调试

可连接的应用程序接口（APIs）

以上几个方面内容在下面做简要介绍，详细讨论请看以后章节。

2.1　　程序建立

在 ILE 中，程序建立包括。

1、把源代码编译为模块。

2、把一个或多个模块连接成为一个程序目标。

你可以使用建立连接的 RPG 程序命令（CRTBNDRPG），象在 OPM 环境中一样用一步处理方式来建立一个程序，这条命令先建立一个临时模块，然后再连接成程序目标，也允许你使用一个连接目录来连接其它的目标。

另外，你可以分别使用编译和连接命令来建立程序。这种两步的处理方式允许重复使用或更新一个模块而不用重新编译程序中的其它模块。另外，由于你可以连接任何 ILE 语言的模块，所以就可以建立和维护多种语言的程序。

用两步处理方式，你可以使用建立 RPG 模块命令（CRTRPGMOD）来建立模块目标，这条命令把源语句编译成为模块目标。模块是不可运行的目标，必须把它们连接为程序才能运行，

连接一个或多个模块应使用建立程序命令（CRTPGM）。

也可以把模块连接成为服务程序，服务程序是把可调用的子例程装配成独立连接的程序目标的一种方法。使用服务程序可以使程序设计模块化和易于维护，你可以使用非本单位的第三方开发的模块，或者相反地，把你的模块封装供第三方使用。建立服务程序应使用建立服务程序命令（CRTSRVPGM）。

图 6 显示了程序建立的两个步骤。

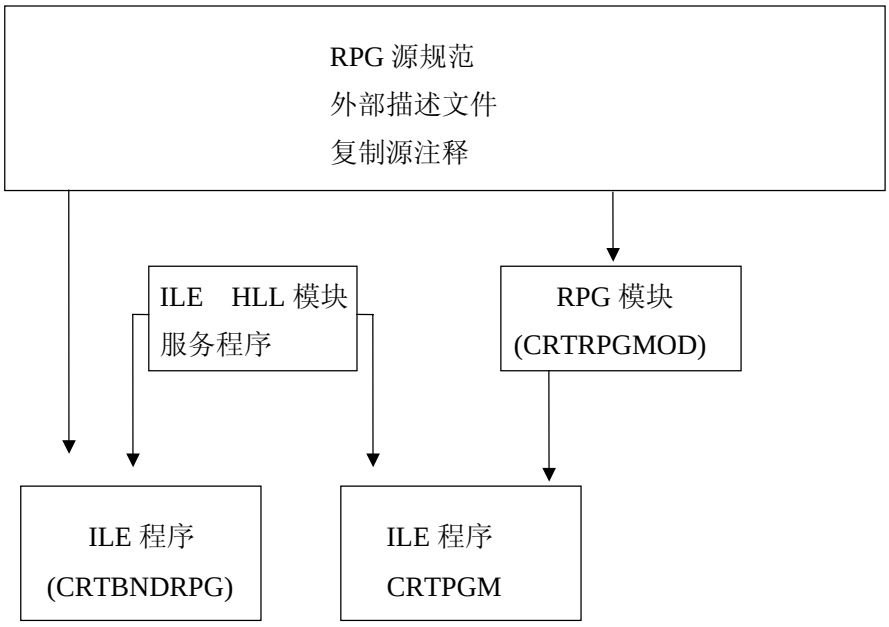


图 6　ILE 中的程序建立

程序一旦建立了，可以使用 UPDPGM 或 UPDSRVPGM 来更新，这是很有用的。因为这意味着只需要有新的或修改的有效模块目标就可以更新程序。

关于一步处理的详细信息，参见第五章 2.2 节中的“使用 CRTBNDPGM 命令建立程序”。关于两步处理的详细信息，参见第六章 2.3 节中的“使用 CRTRPGMOD 和 GRTPGM 命令建立程序”。关于服务程序的详细信息，参见第七章 2.4 节中的“建立服务程序”。

2.2　程序管理

ILE 为在程序运行期间管理程序流程，共享资源，处理语言的语义提供了一般的基础，对于 RPG 使用者来说这意味着你可以比以前更有效地对资源进行控制。

一个 ILE 程序必须要激活到一个活动组中。活动组是在程序建立时指定的，它是为作业分配的工作空间，这样可让一个或多个程序在这个空间内运行。如果在程序被调用时，程序指定的活动组不存在，那么为了保证程序的活动状态，要在作业中建立活动组。

活动组是管理 ILE 应用程序资源和行为的关键因素。例如，你可以把落实控制操作指定在活动组级别。同样可以指定文件覆盖和共享打开的数据路径在正在运行的应用程序的活动组范围内。最后，程序结束的动作也受到它所在的活动组的影响。

关于活动组的详细信息，参见 2.5.6 节中的“管理活动组”。

你可以使用 ILE 语言提供的可连接的 API，动态地为一个运行时数组分配存储空间。这些 API 允许单一语言或混合语言的应用程序访问存储管理功能的中心系列，并为语言提供一个存储模式。

2.3 程序调用

通过 ILE，可以用传统的动态程序调用把 ILE RPG/400 程序与 OPM RPG/400 程序相连接。程序使用动态调用的方式来调用其它程序，用 CALL 语句指定被调用程序的名字，在运行期间判定被调用程序的名字地址，就和以前程序间传送控制一样。

然而，你可以编写与快速的静态调用相关的 ILE 程序。静态调用包括过程间的调用。过程是一组自身控制的代码，它能完成某个任务然后返回到调用者。

一个 ILE RPG/400 模块由一个过程构成。由于过程名是在连接时确定的（建立程序的时候）所以静态调用要比动态调用快一些。

静态调用也允许使用操作描述符，省略参数，并且允许传递多个参数。（省略参数是没有数据传递充当占位的参数）。操作描述符和省略参数当调用连接的 AP1 或用其它 ILE 语言写的过程时是很有用的。RPGⅣ提供了 CALLB 操作码，以支持过程调用和参数省略。关于运行程序的详细信息，参见第八章 2.5 节的“运行一个程序”。关于程序/过程调用的详细信息，参见第九章 2.6 节的“调用程序和过程”。

2.4 源程序调试

在 ILE 中，你可以对任何单一语言或混合语言的 ILE 应用程序进行源语句级的调试。在程序运行期间用调试命令控制程序的流程。可以在程序运行前设置有条件的或无条件的断点。在调用程序后，可以指定语句号码做单步运行，显示或修改变量的值。当程序由于遇到断点，单步命令或运行期间错误而停止时，在程序停止处能显示相关的模块，这时，可以录入更多的调试命令。

关于调试程序的详细信息，参见第十章 3.1 节中的“程序调试”。

2.5 可连接的 AP1

ILE 提供了一些可以连接的 AP1 以补充当前 ILE RPG/400 提供的功能。可连接的 AP1 提供了程序调用和活动能力，条件和存储管理，数学函数和动态屏幕管理。

在 ILE RPG/400 应用程序中可以考虑使用下面一些 AP1：

CEETREC—表示立即终止条件
CEE4ABN—异常结束
CEEFRST—释放存储空间
CEEGTST—取得堆式存储空间

CEECZST—重分配存储空间

CEEDOD—分解操作描述

关于这些 ILE 可连接 AP1 的详细信息，参见第八章 2.5 节中的“运行程序”和系统 API 参考手册。

第三章 程序建立策略

使用 ILE，你可以用很多方法建立程序，这部分介绍了使用 ILE RPG/400 或其它 ILE 语言建立 ILE 程序的三种一般策略。

1、使用 CRTBNPRPG 命令来建立程序与 OPM 存有最大的兼容性。

2、使用 GRTBNDRPG 建立 ILE 程序。

3、使用 CRTRPGMOD 和 CRTPGM 命令建立 ILE 程序。

第一种策略是作为临时性的推荐。它是为了考虑到已经有 OPM 应用程序的用户，可能由于时间不够而不能把他们的应用程序一次转成 ILE。 第二种也是一种临时的策略，它给你熟悉 ILE 的时间。但是也允许你立即使用它们的某些性能。第三种策略包含更广，且提供很大的灵活性。

第一和第二种策略都使用一步建立程序的方法，命令名为 CRTBNDRPG，第三种策略采用两步建立程序的方法。先用 CRTRPGMOD 命令，然后用 CRTPGM 命令。

3.1 策略 1 OPM 兼容的应用程序

策略 1 可以产生与 OPM 程序兼容的 ILE 程序。可用 RPGⅣ的增强功能，但不是全部。当要向 ILE 过渡时，你可能需要这样一个临时程序。

3.1.1 方法

使用下面的步骤建立程序：

1、使用 CVTRPGSRC 命令把原有的源程序转换为 RPG IV。注意要转换源程序中使用/COPY 的成员。

2、用 CRTBNDRPG 命令建立程序目标。指定 DFTACTGRP（*YES）。指定 DPTACTGRP（*YES）意味着程序目标只在默认的活动组中运行。（默认的活动组是所有 OPM 程序运行的组）。这样就使程序目标与 OPM 程序，在复盖范围、打开范围和 RCLRSC 的区域一致。

当使用这种方法时，就不能用 ILE 的静态连接。就是说，你的程序不能包括 CALLB 操作。而且，在建立程序时不能在 CRTBNDRPG 命令中使用 BNDIR 和 ACTGRP 参数。

3.1.2 OPM 兼容程序的例子

图 7 显示了 OPM 兼容程序的运行时样本。它由一个 CL 程序和两个 RPG 程序组成。在这个例子中，一个 RPG 程序被转换成 ILE，而其余的程序没有变化。

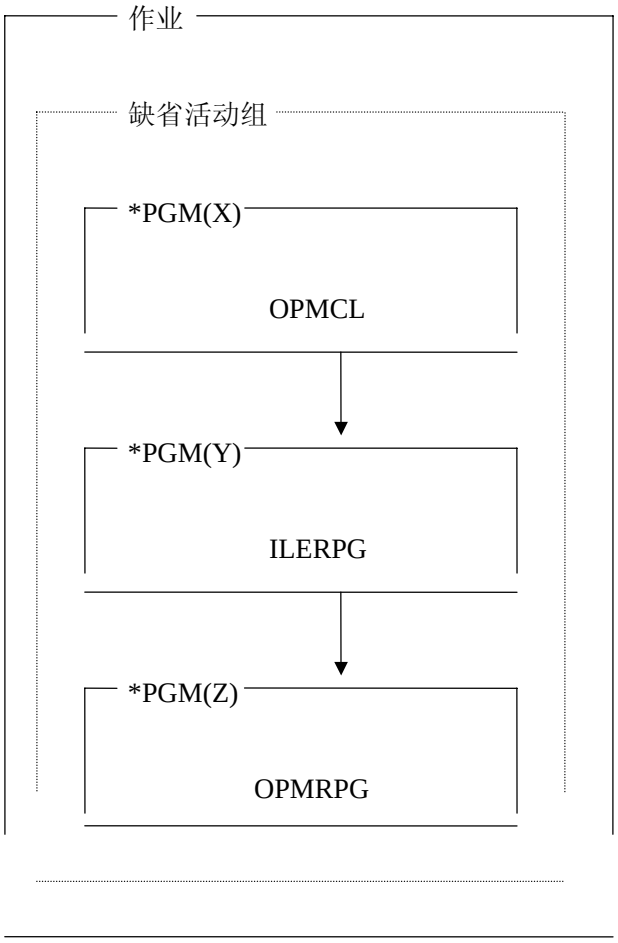


图 7　OPM 兼容的应用程序

3.1.2.1 ILE 作用

以下是在你的应用程序处理时 ILE 的作用：

程序调用　OPM 程序象从前一样动作，当启动作业时，系统自动地建立 OPM 默认活动组，所有的 OPM 应用程序在其中运行，程序可以通过用动态调用的方法调用该默认活动组中的其它程序。

数据　静态数据的存储空间是在程序激活时建立的，并一直存在直到程序结束。当程序结时（无论是正常或非正常结束），程序的存储空间被删除，若要清除没有结束而返回的程序的存储空间，应使用回收资源命令（RCLRSC）。

文件　　文件处理与前面版本所讲的相同，文件在程序正常或非正常结束时被关闭。

错误　　就象在前面版本所讲的那样，编译程序独立地处理每个程序中的错误。所看到的出现在程序中的错误与以前是相同的。然而，现在是错误可以由ILE 条件管理程序做程序之间的通信，以使你能看到程序间的不同信息。这些信息可能会有新的信息标识，因此如果你的 CL 程序监控某个指定的信息标识，那么要修改那个信息标识。

3.1.3　相关信息

转换为 RPGIV—附录 1.2.2 中的“转换你的源程序”
一步建立处理—第五章 2.2 节中的“使用 CRTBNDRPG 命令建立程序”
ILE 静态连接—第九章 2.6 节中的“程序和过程调用”和 ILE 概念
异常处理差异—3.2.1.1.1 中的“OPM 和 ILE 异常处理之间的差别”

3.2　策略 2　使用 CRTBNDRPG 的 ILE 程序

策略 2 产生一个具有先进 ILE 静态连接的程序。由于可以使用连接目录连接几个模块或服务程序，因此，程序可以包括用 CALLB 的静态过程调用，你也可以指定程序运行的活动组。

3.2.1　方法

使用以下步骤来建立一个程序：

1、如果是由 RPGⅢ源程序开始，那么首先使用 CVTRPGSRC 命令把你的源程序转换成 RPGⅣ。

转换时，要确认转换所有的/COPY 成员和调用的程序。另外，如果使用 CL 来调该程序的话，那么一定要确认用 ILE CL 程序而不是用 OPM CL 程序。

2、确定程序运行要使用的活动组。

也以象这个例子一样，在应用程序名字后命名它。

3、如果需要，给出所用的连接目录的名字。这里假定你要用连接目录，那么该目录是已经建立好的。例如，要把你的源程序连接到第三部分程序上，那么要知道连接目录的名字。

4、使用 CRTBNDRPG 建立 ILE 程序，指定 DFTACTGRP(*NO)，指定 ACTGRP 参数中的活动组。如果有的话，还要指定 BNDDIR 参数中的连接目录。

注意，如果指定了 ACTGRP(*CALLER)并且这个程序被一个在默认活动组中的程序调用，那么这个程序在复盖范围、打开范围和 RCLRSC 的范围方面的活动将和 ILE 语义一致。

这种策略的主要不利方面是，你不能有一个永久的模块目标，使它以后可以与其它模块相连接以建立 ILE 程序，而且任何过程调用必须是在连接目录中标识的模块或服务程序。如果想连接两个或更多的模块而不使用连接目录，那么你就需要使用第三种策略。

3.2.2 使用 CRTBNDRPG 的 ILE 程序例子

图 8 显示了一个应用程序运行期间的视图，在应用程序中，ILE CL 程序调用了一个和服务程序相连接的 ILE RPG/400 程序。该应用程序在名为 XYZ 的活动组中运行。

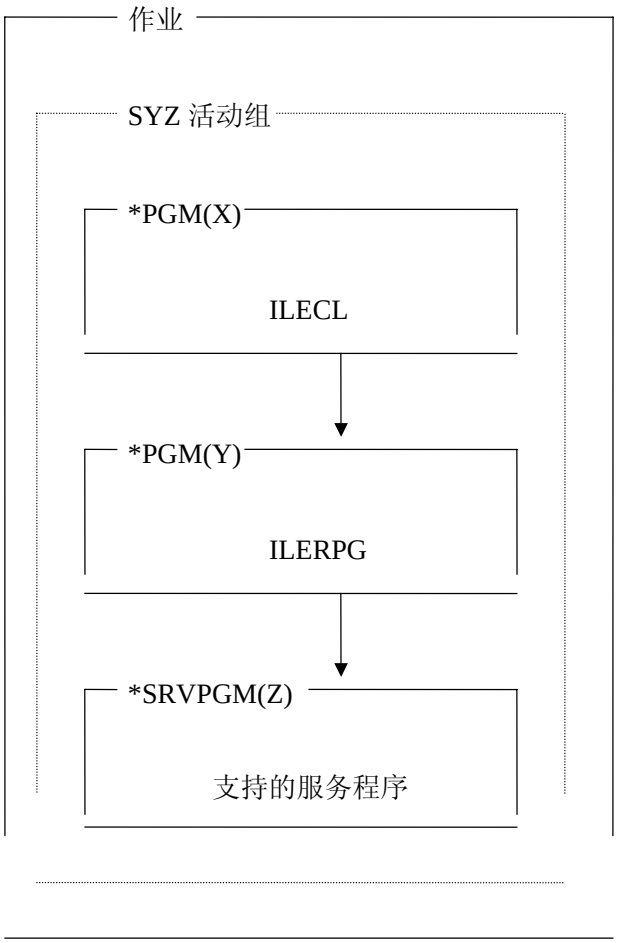


图 8　使用 CRTBNDRPG 的 ILE 程序

3.2.2.1 ILE 的作用

下面详细列出此种方式下 ILE 的作用：

程序调用　当应用程序启动时，如果活动组不存在，那么系统自动地建立它。

应用程序可包括动态的程序调用或静态的过程调用，连接程序中的过程使用静态调用来彼此调用，过程调用 ILE 和 OPM 程序用动态调用。

数据　程序存储空间的生存期与活动组的生存期相同。存储空间一直保持活动状态，直到活动组被删除为止。ILE RPG/400 在运行期间维护数据。以　便使程序结束的语义和重新初始化的数据与 OPM RPG 相一致，即使当

OPM RPG 程序结束时而实际的存储空间没有被删除。如果前次的过程调用

以 LR 为 ON 结束或非正常结束，数据都要重新初始化。

程序中被说明为进口和出口的数据（分别以 EXPORT 和 IMPORT 键字说明）。对于每个模块来说都是外部的，它们只对连接成一个程序的那些模块来说是可知的。

文件　在默认情况下，由系统做的文件处理（包括打开、共享、复盖和落实控制）影响到活动组级。在不同活动组中的程序不能在数据管理级共享文件。如果你想跨活动组共享文件，那么必须在复盖命令中用 SHARE (*YES)参数指定它在作业级打开文件，或在建立文件时用 SHARE(*YES)参数指定。

错误　在同一活动组中调用 ILE 程序或过程时，如果由于前面的原因得到一个异常时，它会显示一个询问信息，那么现在调用的程序就会首先看到这个异常信息。

如果调用的程序有错误指示器或*PSSR，那么获得异常的程序或过程就会非正常结束而不显示询问信息。调用的程序也会做相同的动作（错误指示器会置为 ON，并激活*PSSR）。

调用不同活动组中的 OPM 程序或过程时，那么异常处理就与 OPM RPG 一样，每个程序各自处理自身的异常。看到信息可能有新的信息编码，因此，如果监视某条特定的信息，那么你需要修改那个信息编码。

每种语言处理它本身的错误并且能处理用其它 ILE 语言编写的模块中发生的错误。例如，如果程序中有错误指示器的编码，那么 RPG 可以处理任何 C 的错误。C 也可以处理任何 RPG 的错误。

3.2.3　相关信息

转换到 RPGⅣ	附录 1.2.2 中的“转换你的源程序”
一步建立方法	第五章 2.2 节中的“使用 CRTBNDRPG 命令建立程序”
活动组	2.5.6 节中的“活动组管理”
RCLRSC	2.5.6.5 节中的“回收资源”命令
ILE 静态连接	第九章 2.6 节中的“程序和过程调用”和 ILE 概念
异常处理差异	3.2.1.1.1 节中的 OPM 和 ILE RPG/400 异常处理的差异。
复盖和打开的作用范围	4.2.1 中的“复盖和更改文件输入输出”及 4.2.4 中的“共享打开的数据路径“，也参考 ILE 概念。

3.3　策略 3 使用 CRTRPGMOD 的 ILE 应用程序

这种策略允许你全面地利用 ILE 所提供的概念。然而，越灵活，涉及的内容越多。

本部分讨论了建立程序的三种方案：

单一语言的应用程序

多种语言的应用程序

高级应用程序

ILE 作用与 1.3.2.2.1 中“ILE 作用”相同。

使用本方法应对基本的 ILE 概念有所了解。

3.3.1 方法

因为这种方法是最灵活的，它包括你建立 ILE 应用程序的若干种途径。以下给出了需要做的主要步骤。

1、用专用的命令从源成员建立模块。例如，用 CRTRPGMOD 处理 RPG 源成员，CRTCLMOD 处理 CL 源成员，等等。

2、确定应用程序的 ILE 特性，例如：

确定哪一个模块作为应用程序的起始点，选择的入口模块就是第一个取得控制的模块。在 OPM 应用程序中是命令处理程序，或是菜单选择项调用的程序。

确定程序运行的活动组。（与在命名的活动组中运行类似，该活动组的名字是基于应用程序的名字）。

3、确定是否连接某些模块来建立服务程序。如果是的话，那么使用 CRTSRVPGM 命令建立服务程序。

4、用 CRTPGM 命令连接适当的模块和服务程序。指定根据第 2 步确定的特性参数值。

使用这种方法建立的应用程序可以在完全保护方式下运行。也就是说，只限于它本身的活动组中。而且，可以用 UPDPGM 和 UPDSRVPGM 命令方便地更新。使用这些命令可以增加或替换一个或多个模块，而不必重新建立程序目标。

3.3.2 单一语言的 ILE 应用程序方案

在此方案中，把多个源文件编译成一个模块，然后把它连接成为供 ILE RPG/400 调用的程序。图 9 显示了这个应用程序的运行时视图。

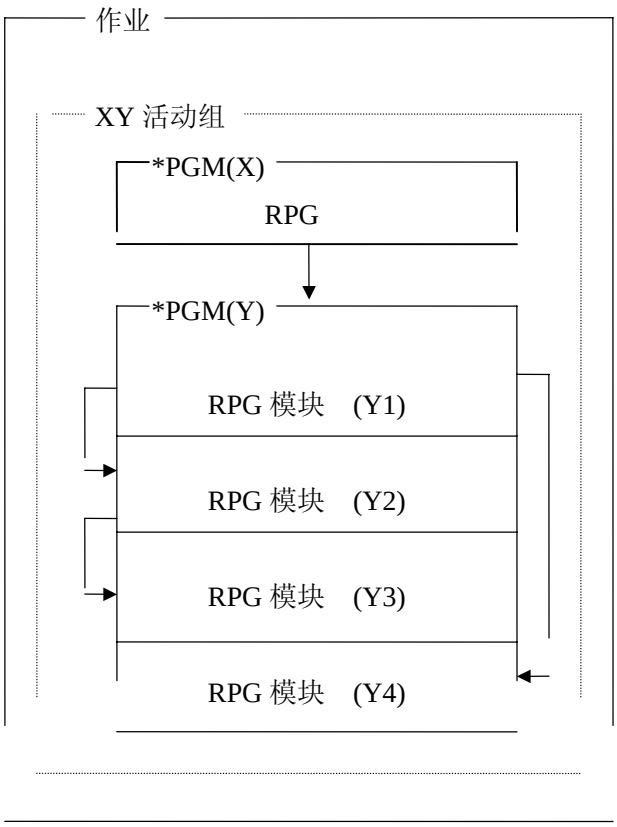


图 9　使用 CRTRPGMOD 和 CRTPGM 的单一语言应用程序

程序 X 对程序 Y 的调用是动态调用，程序 Y 中模块间的调用是静态调用。

参见 1.3.2.2.1 节中的“ILE 作用”，其中介绍了应用程序处理调用，数据，文件和错误过程中 ILE 作用的详细资料。

3.3.3　多种语言的 ILE 程序方案

在此方案中，建立了集成的多语言的应用程序。用某种 ILE 语言写的主模块，调用其它 ILE 语言写的过程。主模块打开文件，而其它模块共享这些文件。由于使用了不同语言，你可能认为是不兼容的。然而，ILE 保证了这一点。

图 10 显示了包括一个多种语言 ILE 程序的应用程序运行期间视图，在程序中一个模块调用不可连接的 API，QUSCRTUS（建立用户空间）。

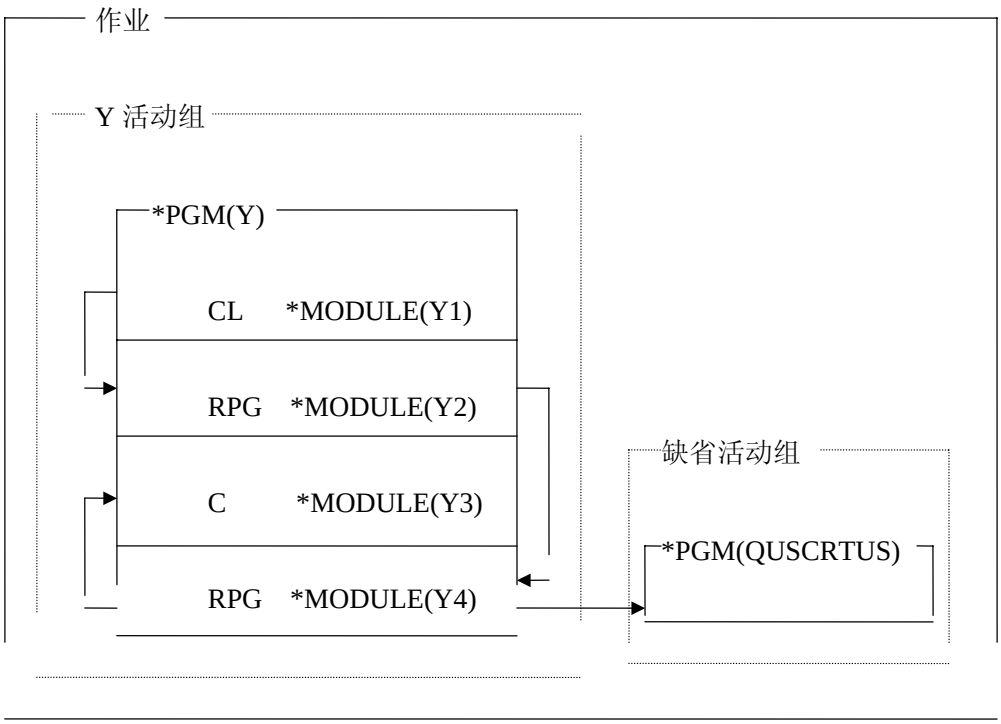


图 10　多种语言的应用程序

程序 Y 对 OPM API 的调用是动态调用，程序 Y 中的各模块间的调用是静态调用。

参见 1.3.2.2.1 节中的“ILE 作用”，其中介绍了应用程序处理调用，数据，文件及错误的过程中 ILE 作用的详细资料。

3.3.4　高级应用程序方案

在此方案中，你用了包括服务程序在内的 ILE 功能的全部益处。特别当服务程序在调用它的程序的活动组中运行时，使用了模块和服务程序间的过程的连接调用，特别能提高性能。

图 11 显示了一个 ILE 程序被连接到两个服务程序上的例子。

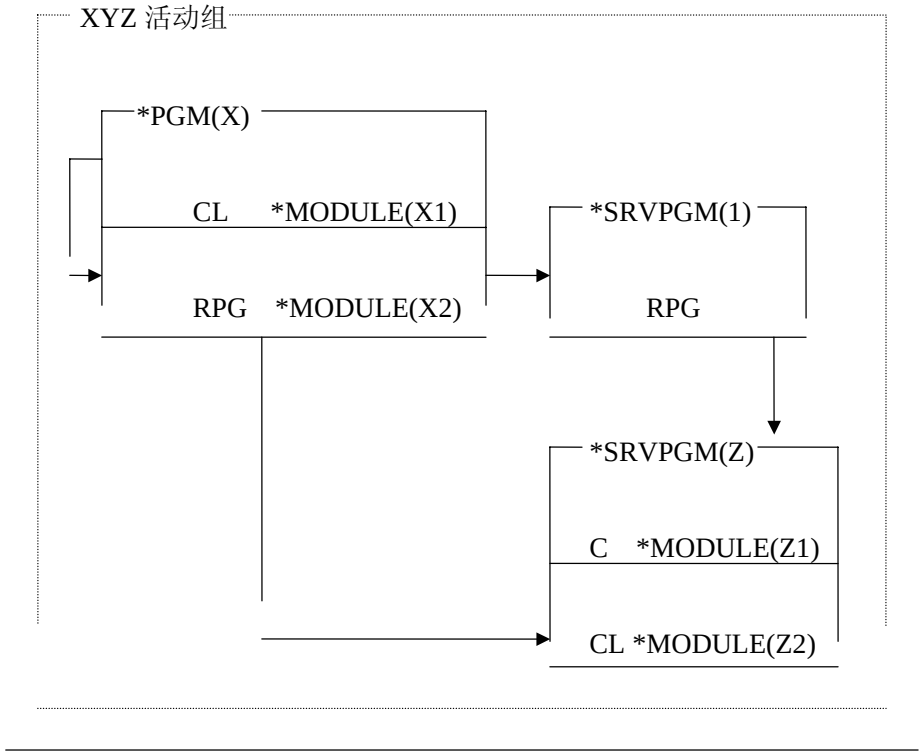


图 11　高级应用程序

程序 X 对程序 Y 和 Z 的调用是静态调用。

参见 1.3.2.2.1 节中的“ILE 作用”，其中介绍了应用程序处理调用，数据，文件及错误的过程中 ILE 作用的详细资料。

3.3.5　相关信息

二步建立方法	第六章 2.3 节中的“用 CRTRPGMOD 和 CRTPGM 命令建立程序”。
活动组	2.5.6 节中的“管理活动组”
ILE 静态连接	第九章 2.6 节中的“过程与程序调用”，也见于 ILE 概念。
异常处理	第十一章 3.2 节中的“处理异常”，也见于 ILE 概念。
服务程序	第七章 2.4 节中的“建立服务程序”，也见于 ILE 概念。
更新程序	2.3.4.1 节中的“使用 UPDPGM 命令”

3.4　应避免的一种策略

ILE 提供了很多建立程序和应用程序的选择。然而，不是所有的都是同样好的，通常，你应该避免出现这样的情况：一个应用程序包含 OPM 和 ILE 程序，而它们分别跨跃 OPM 默认活动组和命名活动组。换言之，应避免图 12 中显示的方案。

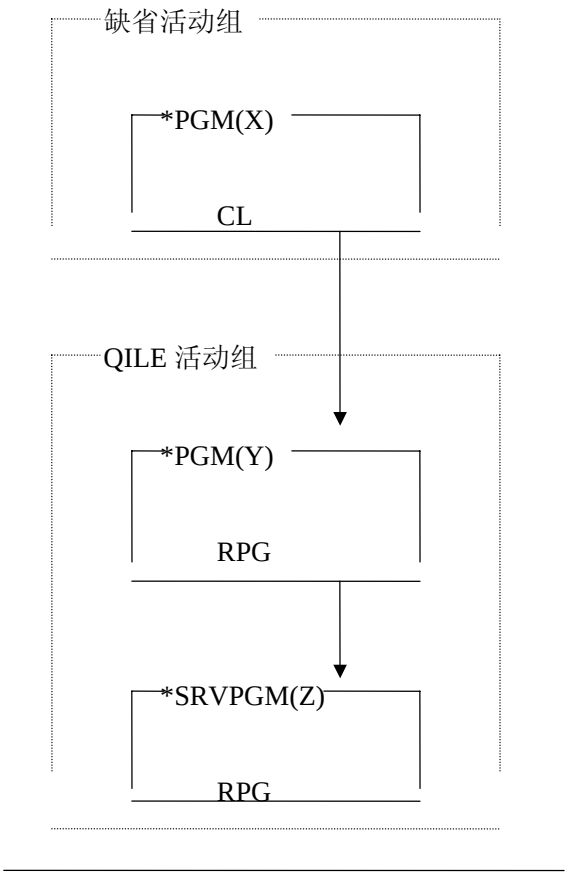


图 12 应避免的方案，一个应用程序被分开放在 OPM 默认活动组和命名的活动组中。

这样做就混合了 OPM 的行为和 ILE 的行为。例如，默认活动组中的程序可能要求 ILE 程序在结束时释放他们的资源。然而，活动组不结束的话这是不可能发生的。

类似的，当应用程序分开放在默认活动组和命名活动组中时，复盖和共享的 ODP 的作用范围将更加难于管理。在默认情况下，命名活动组的作用范围是在活动组级，但是默认活动组的作用范围是调用级或作业级，而不是活动组级。

第二部分

建立并运行 ILE RPG/400 应用程序

此部分提供了建立和运行 ILE RPG/400 程序所需的信息。包括：

- 录入源语句
- 建立模块
- 读编译清单
- 建立程序
- 建立服务程序
- 运行程序
- 传递参数
- 运行时间管理
- 调用其它的程序或过程

以下简要介绍一些 ILE 的术语和概念，这些术语和概念将在 ILE 概念中作全面地讨论。

第四章　录入源语句

本章提供了你录入 RPG 源语句所需要的信息，也简要地介绍了为完成本步骤必需的工具。

要把 RPG 源语句录入到系统中，可采用以下方法：

交互地使用 SEU

交互地使用 CODE/400

开始，把源语句录入到一个叫 QRPGLESRC 的文件中去，文件 QRPGLESRC 的新成员自动地以 RPGLE 为默认类型。接着，建立模块和连接成为程序目标的默认的源文件叫做 QRPGLESRC。IBM 提供了在 QGPL 库中的 QRPGLESRC 源文件，它的记录长度为 112 个字符。

注意：在录入源程序时可用混合大小写的方式，但是，ILE RPG/400 的编译程序将把多数的源程序转换成大写，但不转换文字，数组数据或表数据。

4.1　建立库和源物理文件

源语句是录入到源物理文件的成员中的。在你录入程序之前，库和源物理文件必须已经存在。

要建立一个库，应使用 CRTLIB 命令。要建立源物理文件，使用 CRTSRCPF 命令。建议文件记录长是 112 个字符。该记录长度考虑到了图 13 所示的新的 ILE RPG/400 结构。

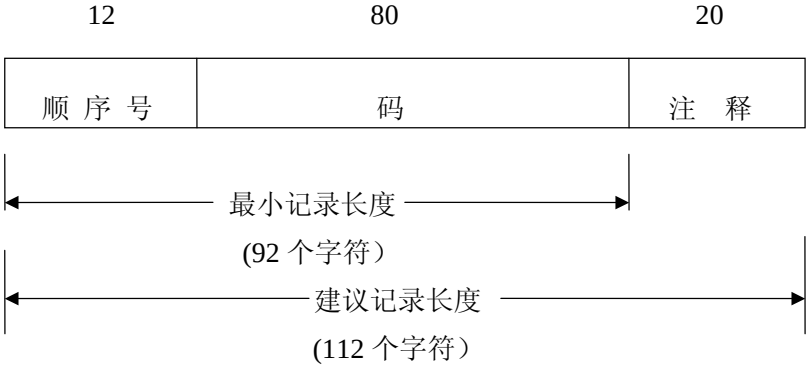


图 13　ILE RPG/400 记录长度分析

因为系统默认的源物理文件记录长度是 92 个字符，所以应该指定记录长为 112。如果指定了小于 92 的记录长度，那么系统可能会因为截短了源代码而不进行编译。

注意：如果你在一个新的库中建立文件，那么你必须先建立库，然后建立源文件。

关于建立库和源文件的更多信息，参见 ADTS/400：源语句录入实用程序(SEU)和 ADTS/400：程序开发管理工具(PDM)。

4.2 使用源语句录入实用程序(SEU)

可以用源语句录入实用程序(SEU)来输入源语句，SEU 提供了不同规范表样板的提示和语法检查。要启动 SEU，使用 STRSEU（启动源语句录入实用程序）命令。关于用这种方法启动并使用 SEU，参见 ADTS/400：源语句录入实用程序。

如果源文件的名字是 QRPGLESRC，那么当 SEU 编辑一个新成员时，自动地将源成员类型设为 RPGLE。否则的话，必须在建立成员时指定 RPGLE。

如果想在键入 STRSEU 后得到提示，按 F4 键。就会出现 STRSEU 显示屏，列出参数，并给出默认值，如果在要求提示前已给出了某些参数值，那显示屏幕上将会填上那些值。

在下面的例子中,输入一个程序的源语句,这个程序将打印一个取自文件中的雇员信息。它告诉你如何做下面的事情：

建立库

建立源物理文件

启动 SEU 编辑任务

输入源语句

1、为了建立名为 MYLIB 的库，键入 CRTLIB(MYLIB)。

CRTLIB 命令建立一个名为 MYLIB 的库。

2、建立名为 QRPGLESRC 的源物理文件，键入 CRTSRCPF FILE(MYLIB/QRPGLESRC) RCDLEN(112) TEXT('Source physical file for all RPG progroms')。

CRTSRCPF 命令在库 MYLIB 中建立了源物理文件 QRPGLESRC。

3、启动一个编辑任务，并建立源成员 EMPRPT 键入。

STRSEU SRCFILE(MYLIB/QRPGLESRC)

SRCMBR(EMPRPT)

TYPE(RPGLE)OPTION(2)

键入 OPITION(2)指出要启动编辑任务来建立一个新的成员。STRSEU 命令在库 MYLIB 的文件 QRPGLESRC 中建立一个新成员 EMPRPT，并启动编辑任务。

SEU 的编辑屏如图 14 所示，注意屏幕自动地转换，以使第 6 列（规范表类型列）位于左边界。

```
    *
*
    *  Columns . . . :    6  76           Edit
MYLIB/QRPGLESRC  *
    *  SEU==>
EMPRPT  *
    *  FMT H
HKeywords++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  *
```

```
    *         *************** Beginning of data
************************************  *
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    * ,,,,,,,
*
    *         ****************** End of data
****************************************  *
    *
*
    *  F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor
*
```

```
    *  F16=Repeat find       F17=Repeat change          F24=More keys
*
    * Member EMPRPT added to file MYLIB/QRPGLESRC.
+   *
    *
*
```

图 14　新成员的编辑屏

在 SEU 的编辑屏中键入图 15 中给出的源语句，使用以下的 SEU 行命令来得到各表提示：

IPF—文件描述规范表的提示

IPI—输入规范表的提示

IPC—计算规范表的提示

IPCX—有扩展参数 2 的计算规范表的提示

IPO—对于输出规范表的提示

IPP—对于输出规范表的连续行的提示

```
F*****************************************************************
F* 模 块 名：EMPRPT                                              *
F* 有关文件：EMPMST（物理文件）                                  *
F*           PRINT （打印文件）                                      *
F* 说    明：这个程序打印 EMPMST 中的雇员信息。                  *
F*****************************************************************
FQSYSPRT   O    F   80        PRINTER
FEMPMST    IP   E           K DISK

D TYPE            S              8

IEMPREC       01

C                   IF        ETYPE = 'M'
C                   EVAL      TYPE = 'MANAGER'
C                   ELSE
C                   EVAL      TYPE = 'REGULAR'
C                   ENDIF

OQSYSPRT   H    1P                   2  6
```

```
0                                                    50 'EMPLOYEE INFORMATION'
0          H     1P
0                                                    12 'NAME'
0                                                    34 'SERIAL #'
0                                                    45 'DEPT'
0                                                    56 'TYPE'
0          D     01
0                         ENAME                      20
0                         ENUM                       32
0                         EDEPT                      45
0                         TYPE                       60
```

图 15　成员 EMPRPT 的源程序

5、按 F3(Exit)退出屏幕，键入 Y(Yes)保存 EMPRPT。

这样，成员 EMPRPT 被保存。

图 16 显示了被 EMPRPT 所引用的文件的 DDS 源语句。

```
          A*********************************************************************
          A* 说      明：这是物理文件 EMPMST 的 DDS，它的记录格式名为 EMPREC。*
          A*              文件中对每个雇员都有一条记录。                       *
          A*********************************************************************
          A*
          A          R EMPREC
          A            ENUM            5  0       TEXT('EMPLOYEE NUMBER')
          A            ENAME          20          TEXT('EMPLOYEE NAME')
          A            ETYPE           1          TEXT('EMPLOYEE TYPE')
          A            EDEPT           3  0       TEXT('EMPLOYEE DEPARTMENT')
          A            ENHRS           3  1       TEXT('EMPLOYEE NORMAL WEEK
HOURS')
          A          K ENUM
```

图 16　对应于 EMPRPT 的 DDS

使用 CRTBNDRPG 命令由此源语句建立程序，参见 2.2.1.1 中的“建立 OPM－兼容的程序

目标”。

4.3 使用 DB2/400 SQL 语句

在 ILE RPG/400 程序中嵌入 SQL 语句，就可以访问 OS/400 的 DB2*，按以下的规则输入 SQL 语言：

在计算规范表中输入 SQL 语句。

在 7－15 列使用/EXEC SQL(/应在第 7 列)启动 SQL 语句。

可以在/EXEX SQL 的同一行中输入 SQL 语句。

也可以在接下的行中使用续行符（在第 7 列输入一个＋）来续写源语句。

在第 7－15 列用/END-EXEC(/在第 7 列)来表示 SQL 语句结束。

注意：程序中 SQL 语句不能超过第 80 列。

图 17 显示了嵌入 SQL 语句的例子。

```
...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
     C
     C              (ILE RPG/400 calculation operations)
     C
     C/EXEC SQL     (the starting delimiter)
     C+
     C+        (continuation lines containing SQL statements)
     C+
     .
     .
     .
     C/END-EXEC     (the ending delimiter)
     C
     C              (ILE RPG/400 calculation operations)
     C
```

图 17　ILE RPG/400 程序中的 SQL 语句

必须录入单个的命令来处理 SQL 语句。

详细信息参见 DB2/400 SQL 编程和 DB2/400 SQL 参考手册。有关源语句录入实用程序 SEU 处理 SQL 语法检查的信息，请参见 ADTS/400。

4.4 在程序中包括图形数据

在RPGⅣ语言中，现在提供对图形数据类型(G)字段的全面支持。

关于 ILE RPG/400 图形支持的详细信息，参见 ILE RPG/400 参考手册。

第五章 使用 CRTBNDRPG 命令建立程序

本章介绍了如何用 CRTBNDRPG 命令，从 RPGⅣ的源语句来建立一个 ILE 程序。使用这条命令可以建立以下两类的 ILE 程序。

设有静态连接的 OPM 兼容程序

有静态连接单模块的 IEL 程序

建立第一种类型还是第二种类型的程序，取决于 CRTBNDRPG 命令中的 DFTACTGRP 参数是规定为*YES 还是*NO。建立第一种类型的程序使程序在打开范围，复盖范围和 RCLRSC 的区域和 OPM 程序一致。具有这种高级的兼容性是因为它们象 OPM 程序一样在同一个活动组中运行，即默认活动组。

然而这种高兼容性导致不能进行静态连接，静态连接指的是调用过程的能力（在其它模块或服务程序中）及使用过程指针的能力。换言之，不能在程序中使用 CALLB 操作码，也不能在程序建立时连接其它模块。

建立第二种类型的程序，使程序具有静态连接的 ILE 特征。可以在程序建立时指定程序将要在其中运行的活动组以及静态连接的某些模块。另外，可以在程序中使用 CALLB 操作码。

5.1 使用 CRTBNDRPG 命令

建立连接的 RPG(CRTBNDRPG)命令只用一步处理由 RPGⅣ源语句建立程序目标。也允许通过使用连接目录连接其它的模块或服务程序。

该命令启动 ILE RPG/400 编译程序，并在 QTEMP 库中建立一个临时的模块。然后，把它连接到类型为*PGM 的程序目标中，一旦程序目标建立，用于建立程序的临时模块就被删除。

当你想从独立的源代码（不用连接其他模块的代码）建立程序目标时，CRTBNDRPG 命令是很有用的。因为它有建立与连接二个步骤，另外它允许建立与 OPM 兼容的程序。

注意：如果想保留一些模块目标，以便与其它目标连接成为程序，那么必须用 CRTRPGMOD 命令建立目标。详细信息参见第 6 章 2.3 节中的“使用 CRTRPGMOD 和 CRTPGM 命令”建立程序。

可以交互地或批处理或从控制语言(CL)程序中使用 CRTBNDRPG 命令。如果交互地使用该命令并需要提示的话，键入 CRTBNDRPG，然后按 F4(提示)。如果需要帮助，那么键入 CRTBNDRPG，然后按 F1(帮助)。

表 2 总结了 CRTBNDRPG 的参数，并给出默认值。

表 2　　CRTBNDRPG 的参数和他们的默认值，按功能分组

程序标识	
PGM(*CLIRLIB/*CTLSPEC)	指定程序名和库
SRCFILE(*LIB/QRPGLESRC)	标识源文件和库
SRCMBR(*PGM)	标识包含源规范表的成员
TEXT(*SRCMBRTXT)	程序的简要描述
程序建立	
GENLVL(10)	程序能建立的错误级别
OPTION(*GEN)	*GEN/*NOGEN，决定程序是否建立
PBGVIEW(*STMT)	指定调试视图的类型，如果需要调试，是否包含于程序中
OPTIMIZE(*NONE)	确定优化级别
REPLACE(*YES)	确定是否替代已存在的程序
BNNDIR(*NONE)	指定用于符号分析的连接目录
USRPRF(*USER)	指定运行程序的用户配置文件
AUT(*LIBCRTAUT)	指定建立程序的授权类型
TGTRLS(*CURPENT)	指出目标运行的软件版本
编译清单	
OUTPUT(*PRINT)	确定是否产生编译清单
INDENT(*NONE)	确定是否在清单中显示缩进，并指出标志缩进的字符
OPTION(*XREF*NOSECLVL *SHOWCRY*EXPDDS*EXT)	指定编译清单的内容
数据转换选项	
CVTOPT(*NONE)	指定外部描述文件的数据类型的处理方式
ALWNULL(*NO)	确定程序是否接受容许空值字段的值
FIXNBR(*NONE)	确定无效的区十进制数是否被整理转换为压缩型数据.
运行时的考虑	
DFTACTGRP(*YES)	指出该程序是否总在 OPM 默认活动组中运行
ACTGRP(QILE)	指出程序运行的活动组
STRSEQ(*NEX)	指出使用的排序顺序表
LANGID(*JOBRUN)	与 SRTSEQ 并用来指定排序顺序的语言标识
TRUNCNBR(*YES)	指定发生数字溢出时采取的动作

参见附录 C，1.3 中的“建立命令”，其中有 CRTBNDRPG 的语法图解和参数描述。

5.1.1 建立一个与 OPM 兼容的程序目标

在此例子中使用 CRTBNDRPG 命令，由源程序 EMPRPT，如 2.1.2 节中的图 15 所示，建立一个与 OPM 兼容程序。

1、要建立目标，键入

CRTBNDRPG PGM(MYLIB/EMPRPT)

SRCFILE(MYLIB/QRPGLESRC)

TEXT('ILE RPG/400 program') DFTACTRP(*YES)。

CRTBNDRPG 命令在库 MYLIB 中建立程序 EMPRPT，它将在默认活动组中运行。这时产生编译清单。

注意：DFTACTGRP(*YES)提供，OPM 兼容性，有了这项就不允许你输入 ACTGRP 和 BNDDIR 的参数值，另外，如果源程序中包含 CALLB 操作码，那么将会出错误信息且编译结束。

2、键入以下 CL 命令中的一个来看生成的编译清单。

DSPJOB 并选择选项 4（显示假脱机文件）

WRKJOB

WRKOUTQ 队列名

WRKSPLF

5.1.2 建立用于源语句调试的程序

在此例子中建立可以用源语句调试程序 EMPRPT。CRTBNDRPG 和 CRTRPGMOD 的 DBGVIEW 参数决定了在编译时产生何种调试数据。该参数提供了六种选项：

*STMT—允许用编译清单在语句处显示变量值，设置断点，此时，不显示源语句。

*SOURCE—建立与输入的源语句相同的显示。

*COPY—建一个源语句显示和包括带有/COPY 成员的源语句的显示。

*LIST—建立与编译清单相似的显示。

*ALL—建立以上所有类型的显示。

*NONE—不产生调试数据。

EMPRPT 的源语句如 2.1.2 节中的图 15 所示。

1、要建立目标，键入：

CRTBNDRPG PGM(MYLIB/EMPRPT)DBGVIEW(*SOURCE)这个程序以它的源成员 EMPRPT 为名建立在库 MYLIB 中。注意在默认方式下，将会以参数 DFTACTGRP(*YES)建立，这个程序目标可以用源语句视图调试。

2、要调试程序键入：

STRDBG EMPRPT

图 18 显示了键入以上命令后的屏幕显示。

---

*

*

```
      *                              Display Module Source
*
      *
*
      *  Program:   EMPRPT         Library:   MYLIB          Module:   EMPRPT
*
      *       1
F*****************************************************************  *
      *       2       F* 模 块 名：  EMPRPT                                  *
      *       3       F* 有关文件：  EMPMST（物理文件）
        *
      *       4       F*             PRINT     （打印文件）
        *
      *       5       F* 说     明：这个文件打印 EMPMST 中的雇员信息
           *
      *       6       F*                                                    *
      *       7
F*****************************************************************  *
      *       8       FQSYSPRT   O    F   80        PRINTER
*
      *       9       FEMPMST    IP   E           K DISK
*
      *      10
*
      *      11       D TYPE            S              8
*
      *      12
*
      *      13       IEMPREC        01
*
      *      14
*
      *      15       C                   IF        ETYPE = 'M'
*
      *
More...  *
      *  Debug . . .
*
```

```
      *
*
      *  F3=End program   F6=Add/Clear breakpoint   F10=Step   F11=Display
variable      *
      *  F12=Resume       F13=Work with module breakpoints     F24=More keys
*
      *
*
      *
*
```

图 18　EMPRPT 的显示模块源语句屏幕

在这个屏幕上（显示模块源语句屏幕）你可以输入调试命令来显示或修改字段值，设置断点来控制调试程序流程。

关于调试的详细信息，参见第 10 章 3.1 节中的“程序调试”。

5.1.3　建立一个使用静态连接的程序

在此例子中使用 CRTBNDRPG 命令建立一个程序 COMPUTE，在程序建立时，连接一个服务程序。假设要把程序 COMPUTE 与购买来的高级计算的服务程序相连接。要连接源程序的目录名为 MATH，这个目录包含一个服务程序的名字，该服务程序中包括几个组成它的过程。要建立目标，键入：

```
CRTBNDRPG PGM(MYLIB/COMPUTE)
          DFTACTGRP(*NO) ACTGRP(GRP1) BNDDIR(MATH)
```

在程序建立时，源程序将与在连接目录 MATH 中指定的服务程序相连接。这意味着调用服务程序中的过程，将比动态调用要省时间。

当调用程序时，将在活动组 GRP1 中运行，CRTBNDRPG 的 ACTGRP 默认值为 QILE。但是，建议在唯一的活动组中运行你的应用程序，以确保相关联的资源被完全保护。

注意：为了能输入 ACTGRP 和 BNDDIR 的参数值，必须把参数 DFTACTGRP 的值设置为*NO。

关于服务程序的详细信息，参见第七章 2.4 节“建立服务程序”。

5.2　使用编译清单

本部分讨论了如何得到清单，如何用它得到帮助来处理下面问题：

修正编译错误

修正运行时错误

为维护工作准备文档

参见附录 D，附录 1.4 节中的“编译清单”，其中有清单各部分的信息和完整的样本清

单。

5.2.1 获取编译清单

要获得编译清单应在 CRTBNDRPG 命令或 CRTRPGMOD 命令中指定 OUTPUT(*PRINT)。（这是它们的默认值），如果指定了 OUTPUT(*NONE)将不产生清单。

指定 OUTPUT(*PRINT)将产生至少包含以下部分的编译清单：

序（命令选项总计）

源清单，包括：

—联机诊断信息

—匹配字段表（如果使用了有匹配字段的 RPG 周期的话）

附加的诊断信息

输出缓冲区中的字段位置

/COPY 成员表

编译时数据，包括：

—交替对照顺序记录和表或 NLSS 信息和表

—文件传输记录

—数组记录

—表记录

信息总计

最终总计

代码生成报告（只在有错误时出现）

连接报告（仅对 CRTBNDRPG 命令，只在有错误时出现）。如果在建立命令的 OPTION 参数中指定相应的值的话，那么编译清单会包括以下附加的信息。

*EXPDDS 外部描述文件规范表（出现在清单的源语句部分）

*SHOWCPY /COPY 成员的源记录（出现在清单的源语句部分）

*EXPDDS 键字字段信息（自己一部分单元）

*XREF 交叉引用清单（自己一部分单元）

*EXT 外部引用清单（自己一部分单元）

*SECLUL 二级信息文本（出现在信息总计部分）

注意：对建立命令而言，除了*SECLUL 以外，以上所有的值都是 OPTION 参数的默认值。不必修改 OPTION 参数，除非不想要某个清单部分或者想在清单中包括第二级文本。

5.2.1.1 剪裁编译清单

可以改制页标题或改制行距来剪裁编译清单。

改制页标题：页标题信息包括产品信息行和由/TITLE 命令提供的标题。产品信息行包括 ILE RPG/400 编译程序和库版权注意信息，成员，源程序所在库，模块建立的日期和时间，还有清单的页数。

可以用/TITLE 编译命令指定清单中标题。这条命令让你指定在每页编译清单顶部出现的内容。这条信息，要优先于通常页标题信息。如果该命令是源成员中的第一条记录，那么这条信息也会出现在序言部分。

也可以改变清单中页标题和其它信息中使用的日期分隔符，日期格式和时间分隔符。一般地，编译程序根据作业属性来决定这内容。要修改它们应使用修改作业(CHGJOB)命令，用这条命令可以：

选择以下日期分隔符之一：*SYSLVL，*BLAND，斜线(/)，连接号(-)，句点(.)，或逗号(，)。

选择以下日期格式之一：*SYSLVL，*YMD，*MDY，*DMYAKG *JUL。

选择以下时间分隔符之一：*SYSLVL，*BLAND，冒号(:)，逗号(，)或句点(.)。

这些值用在清单中出现时间和日期字段的地方。

改制行距：通常清单的每部分都从新的一页开始；清单的每一页都以产品信息开始（除非源成员包含/TITLE 命令）。这产品信息出现在第二行而标题出现在第一行。

使用/EJECT 和/SPACE 编译命令可以控制编译清单的间隔和页码，/EJECT 命令强制产生一个页中断，开始另一页。/SPACE 命令控制清单的行距。关于这些命令的详细信息参见 ILE RPG/400 参考手册。

5.2.1.2 编译清单中的结构化操作缩进

如果源程序规范表中包括结构化操作码（例如 DO－END 或 IF－ESLE－END），你可能希望在源程序清单中这些操作有缩进显示。INDENT 可以让你指定是否显示缩进，并指定标记缩进的字符。如果不想缩进应指定 INDENT(*NONE)，这是默认值。如果想要缩进的话，那么最多指定两个字符来标记缩进。

例如，要指定让结构化操作缩进，并以显示(|)和空格标记，应指定 INDENT('|')。

如果要求缩进，那么正常情况下出现在源程序清单中的一些信息要被删除，这样才能可以缩进。以下的内容不会出现在清单中：

DO 编号

最后一次更新

页/行

如果在指定缩进的同时指定了清单调试视图，那么在调试视图中不会出现缩进。

图 19 显示了缩进产生的源程序清单。缩进标记为‘| ’。

```
      Line   <---------------------- Source Specifications
-------------------------------------------->><---- Comments ----> Src  Seq
      Number ....1....+....2....+<-------- 26 - 35
--------->....4....+....5....+....6....+....7....+....8....+....9....+...10 Id
```

```
Number
          33
C*****************************************************************
000000        002000
          34 C*     MAINLINE
*                                       000000        002100
          35
C*****************************************************************
000000        002200
          36 C                    WRITE                         FOOT1
002300
          37 C                    WRITE                         HEAD
002400
          38 C                    EXFMT                         PROMPT
002500
          39 C*
000000        002600
          40 C                    DOW                           NOT *IN03
002700
          41 C      CSTKEY        | SETLL                       CMLREC2
----20                              002800
          42 C                    | IF                          *IN02
002900
          43 C                    | | MOVE                      '1'           *IN61
003000
          44 C                    | ELSE
003100
          45 C                    | | EXSR                      SFLPRC
003200
          46 C                    | END
003300
          47 C                    | IF                          NOT *IN03
003400
          48 C                    | | IF                        *IN04
003500
          49 C                    | | | IF                      *IN61
003600
          50 C                    | | | | WRITE                 FOOT1
```

```
003700
          51 C                   |||| WRITE               HEAD
003800
          52 C                   ||| ENDIF
003900
          53 C                   ||| EXFMT                PROMPT
004000
          54 C                   || ENDIF
004100
          55 C                   | ENDIF
004200
          56 C                  ENDDO
004300
          57 C*
000000       004400
          58 C                  SETON
LR----                             004500
```

图 19　有缩进的清单的样本源程序

5.2.2　改正编译错误

对改正编译错误有用的编译清单主要部分有：

源语句部分

附加信息部分

/COPY 表部分

各类总计部分

出现在源语句部分的联机诊断信息，在源语句段上找到的错误，编译程序立即标识，其它错误是在编译期间接到附加信息后标识出的。提示这些错误的信息是在源语句部分和附加信息部分中标识的。

为了帮助改正编译错误，也许需要在清单中包括第二信息文本—特别对于 RPG 初学者。要想这样做的话，需在建立命令中指定 OPTION(*SECLVL)参数。这样就会把第二级信息加到信息总计中的信息清单中。

最后，要记住编译清单是程序的一个记录。因此，如果有错误的话，可以使用清单来检查源程序。除了源语句外，要检查的编译清单部分还包括：

匹配字段表

如果你使用有匹配字段的 PRG 程序周期，那么可以用这个表来确定编译程序是否象预期的那样识别他们。

以偏移量定义的输出字段与文本或字段名一起列出起始和结束位置，以此确定编译程是否象预想的那样识别输入。

编译时数据。

列出 ALTSEQ 和 FIRANS 记录和表，NLSS 信息和表也列出来。表和数组都被明确的标识出来。用这些来确定编译程序是否象预想地那样识别它们。

5.2.2.1 使用联机诊断信息

有两种联机诊断信息：定位类和非定位类。

定位类信息确切地指出了错误发生的位置，图 20 显示了定位类联机诊断信息的例子。

```
      Line   <---------------------- Source Specifications
--------------------------><---- Comments ----> Do  Page  Change  Src  Seq

Number ....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
...8....+....9....+...10 Num Line  Date    Id   Number

         63  C                   SETOFF
_12___                                                  003100
      ======>
aabb
      ======>
cccc

      *RNF5051 30 a Resulting-Indicator entry is not valid; defaults to blanks.
      *RNF5051 30 b Resulting-Indicator entry is not valid; defaults to blanks.
      *RNF5053 30 c Resulting-Indicators entry is blank for specified operation.
```

图 20　定位类联机诊断信息样本

在这个例子中，没有在 71－72 或 73－74 列而是错误地在 72－73 列设置了一个指示器，三个定位符号‘aa’，‘bb’，和‘cc’指出了发生错误的地方。正确的列位置和解释的变量为高亮度显示.在这里,信息 RNF5051 指出被‘aa’和‘bb’标记的字段没有有效的指示器。因为没有有效的指示器，所以编译程序假定字段为空，但。SETOFF 操作要求有指示器。因此产生了另一个错误，就是信息 RNF5053 中字段‘CCCC’所指出的。

错误是按它出现的顺序来排列的。一般的，要先改正先出现的错误，然后再改后面的。

非定位类联机诊断信息也指示错误。由于错误的性质，它们不能指出在源语句中的确定

位置。而只能指出问题。图 21 显示了非定位类联机诊断信息的例子。

```
      Line   <---------------------- Source Specifications
--------------------------><---- Comments ----> Do  Page  Change  Src  Seq

Number ....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
...8....+....9....+...10 Num Line  Date    Id   Number

          6  FINTERN    IS   F   72        DISK
12345              000600
          7  FQSYSPRT   O    F  256        DISK
000700
          8  FF7145A    CT   F   28        DISK
000800
          9  FF7145B    CT   F   28        DISK
000900
         10  F*
12345 100792       001000

      *RNF2025 30        6 INTERN defaults to primary file.


         11  DARRA             S             28    FROMFILE(F7145A)
TOFILE(F7145A)                                                  001100
```

图 21　非定位类联机诊断信息样本

此例子中有几个文件规范表，其中一个定义做次文件，但是没有定义主文件。因为主文件是必需的，所以产生了错误信息(RNF2025)。编译程序不能指出应定义哪一行的文件做主文件。当下一种规范表开始时，它就会认为不存在这样的行。

5.2.2.2　使用附加诊断信息

附加诊断信息部分指出了当一行或多行编码整体看起来时会出现的错误，这些信息一般不放在出现问题的编码行，编译程序在写有问题的程序源码时，还不知道错误是否存在。但是，当能够发现错误时，信息行就会包括与此相关的源语句行号。

5.2.2.3 使用 SEU 显示编译清单

SEU 的分屏/显示功能(F15)允许显示输出队列中的编译清单，可以在修改源码的同时，查看上一次的编译结果。

在显示时可以查找错误，并改正那些有错误的源语句。要查找错误，在显示屏幕的 SEU 命令行上键入 F *ERR。有第一（或下一个）错误的行就会高亮度显示，并且这个错误的第一级信息文本在屏幕底部出现。可以把光标放在底部的信息上，然后按 F1（帮助）键，来显示第二级信息文本。

只要有可能，编辑清单中的错误信息会指出有错误行的 SEU 顺序号。行号出现在信息文本之前。

关于显示编译清单的完整信息，参考 ADTS/400：源语句录入实用程序。

5.2.3 改正运行时错误

清单的源语句部分，对于改正运行时错误也是有用的。许多运行时错误都指出了错误的语句号。清单左边的行号与运行时错误信息的语句号相对应。源语句 ID 号和编译清单右边的 SEU 顺序号指出了源成员和记录。

这样就可使用两者来确定哪一行需要检查。

5.2.3.1 选择有 Debug 窗口选项的编译清单可选项

改正运行时错误经常用到程序调试。考虑以下问题可能在调试程序时有帮助：

如果使用源语句调试那么可选择以下调试显示：*STMT，*SOURCE，*LIST，*COPY，*ALL。

如果要在调试时用编译清单作为辅助工具可以指定 OUTPUT(*PRINT)从而得到一个清单。如果使用源语句显示进行调试的话，那么编译清单是很重要的。因为设置断点的行号就是那些在清单中指出的行号。

如果考虑做调试，那么可能需要用参数

DBGVIEW(*ALL)，OUTPUT(*PRINT)和 OPTION(*SHOWCPY)来编译源程序，这样允许使用源语句显示或清单显示，并包括/COPY 成员。

如果指定了 DBGVIEW(*LIST)，那么调试时有效的信息取决于 OPTION 参数，只有指定了 OPTION(*SHOWCPY *EXPDDS)，才能包括/COPY 成员和外部描述文件。

5.2.4 使用编译清单做维护

没有错误的程序的编译清单可用来作为文档，当：

文档用于向新的程序员教授程序。

日后更新程序。

在这种情况下，应当有完整的清单，它是由规定参数 OUTPUT(*PRINT)和 OPTION(*XREF *SHOWCPY *PDDS *EXT)产生的。注意对于建立命令以上参数的值都是默认值。

对程序维护有用的是在清单的序言部分，它告诉你：

谁编译了模块/程序

用哪个源码来产生的模块/程序

当编译模块/程序时使用了哪些选项

在以后要改动程序时，可能需要知道命令选项（例如，选择的调试显示或使用的连接目录）。

以下的 OPTION 参数的指定值提供给附加信息：

*SHOWCPY 和*PPDS 提供了程序的完整描述。

包括组成 COPY 成员的所有规范表和由外部描述文件产生的规范表。

*XREF 允许检查模块/程序中的文件，字段及指示器的使用。

*EXT 允许查看哪个过程或字段做为程序的进口或出口。它也指出了生成外部描述文件和数据结构描述的实际文件。

5.3 访问 RETURNCODE 数据区

CRTBNDRPG 和 CRTRPGMOD(参见 2.3.1.1 节中的“使用 CRTRPGMOD 命令”)命令都用最后一次编译的状态来建立和更新一个数据区。这个数据区命名为 RETURNCODE，它是 400 个字符长放置于 QTEMP 库中。

要访问 RETURNCOPE 数据区，应把*DTAARA DEFINE 语句的因子 2 规定为 RETRUNCODE。

数据区 RETURNCODE 的格式如下：

字节	内容及意义
1	对于 CRTRPGMOD，字符’1’表示模块已经建立在指定的库中了，对于 CRTBDNRPG，字符’1’表示与程序同名的模块被建立在 QTEMP 中。
2	字符’1’表示因为有编译错误所以编译失败。
3	字符’1’表示由于有源语句错误而导致编译失败。
4	没有设定，总是’0’。
5	字符’1’表示翻译程序未被调用，原因是没有在 CRTRPGMOD 中或 CRTBNDRPG 中指定 OPTION(*NOGEN)，或者是在调用翻译程序前编译失败。
6-10	源语句的条数。
11-12	命令中的错误级别。
13-14	诊断信息的最高错误级别。
15-20	模块(CRTRPGMOD)或程序(CRTBNDRPG)中发现的错误的总数。
21-26	编译日期
27-32	编译时间
33-100	未设置，总是空值。

101-110	模块(CRTRPGMOD)名或(CRTBNDRPG)名。
111-120	模块(CRTRPGMOD)所在库名或程序(CRTBMDRPG)所在库名。
121-130	源文件名
131-140	源文件所在库名
141-150	源文件成员名
151-160	编译清单文件名
161-170	编译清单库名
171-180	编译清单成员名
181-329	未设置，总为空值
330-334	精确到十分之一秒的总共用去的编译时间
335	未设置，总为空值
336-340	精确到十分之一秒的编译时间
341-345	精确到十分之一秒的翻译时间
346-379	未设置，总为空值
380-384	精确到十分之一秒的总共用于编译的 CPU 时间
385	未设置，总为空值
386-390	精确到十分之一秒的用于编译的 CPU 时间
391-395	精确到十分之一秒的翻译占用 CPU 时间
396-400	未设置，总为空值

第六章 用 CRTRPGMOD 和 CRTDGM 命令建立程序

两步程序建立步骤包括使用 CRTRPGMOD 命令把源语句编译成模块，然后使用 CRTPGM 命令把一个或多个模块连接成为程序。使用这种可以建立永久模块。这样做的结果，是使一个应用程序模块化，而不必重新编译整个应用程序。也允许在不同的应用程序中都使用同一个模块。

本章的内容包括：

由 RPG Ⅳ源语句建立模块目标

使用 CRTPGM 命令把模块连接成程序

阅读连接清单

修改模块或程序

6.1 建立模块目标

模块是 ILE 编译程序的输出，它是不可运行的目标（类型为*MODULE），是 ILE 程序的基本连接模块。

一个模块由一个或多个过程组成。允许的过程个数取决于语言类型。一个 ILE RPG/400 模块由一个过程组成，它有自己的 LR 语义、周期、文件控制块和静态存储空间。

模块的建立包括源成员的编译，如果编译成功，建立一个*MODULE 目标。*MODULE 目标包括模块中进口和出口引用的清单。如果在编译时有要求的还包括调试数据。模块本身不能运行。必须连接一个或多个模块成为可运行的程序目标，也可以连接一个或多个模块建立一个服务程序（类型为*SRVPGM）。然后，通过静态过程调用访问连接在一起的模块中的过程。

这种连接模块的方法允许做下面的事情：

重复使用一段代码，一般会使程序变得更短小。短小的程序性能更好且易于调试。

维护共享编码，使其向程序的其他部分传送错误代码的机会很少，更有效地管理大的程序。模块方式允许把以前的程序分割为可以独立管理的多个部分。如果程序需要改进，只需重新编译那些被改动的模块。

可以根据任务的需要选择最合适的语言建立模块，然后连接这些模块建立多种语言编写的程序。

有关模块的详细信息，请参考 ILE 概念。

6.1.1 使用 CRTRPGMOD 命令

使用(CRTRPGMOD)命令建立模块。可以交互地，或作为批处理输入流的一部分，或在命令语言(CL)程序中使用该命令。

如果交互地使用该命令并需要提示的话，键入 CRTRPGMOD，然后按 F4(提示)键，如果需要帮助，键入 CRTRPGMOD 然后按 F1(帮助)键。

表 3 列出了 CRTRPGMOD 命令的参数及系统提供的默认值。命令的语法图解和参数描述见于附录 C，1.3 节中的“建立命令”。

表 3.按功能分组的 CRTRPGMOD 参数及其系统默认值

模块标识	
MODULE(*CURLIB/*CTLSPEC)	确定建立的模块名和库名
SRCFILE(*LIBL/QRPGLESRC)	指定源文件和库名
SRCMBR(*MODULE)	指定包含源语句的文件成员
TEXT(*SRCMBRTXT)	为模块提供简要描述
模块建立	

GENLVL(10)　　确定模块建立的错误级别（0-20）
OPTION(*GEN)　　*GEN/*NOGEN，决定模块是否建立
DBGVIEW(*STMT)　　指定要包含在模块中的调试视图的类型
OPTIMIZE(*NONE)　　确定优化级别
REPLACE(*YES)　　确定是否替换已存在的模块
AUT(*LIBCRTAUT)　　指定建立模块的权限
TGTRLS(*CURRENT)　　指定目标要运行的软件版本
编译清单
OUTPUT(*PRINT)　　确定是否产生编译清单
INDENT(*NONE)　　确定是否在清单中显示缩进，如选，并指定标记缩进的字符
OPTION(*XREF　*NOSECLVL　*SHOWCPY(*EXPDDS　*EXT)　　指定编译清单包含的内容
数据转换选项
CVTCPT(*NONE)　　指定如何处理外部描述文件中的变量数据
ALWNULL(*NO)　　确定模块是否接受来自允许空值字段的值
F1XNBR(*NONE)　　确定无效的区位十进制数据是否修整转换为压缩型数据
运行时的考虑
SRTSEQ(*HEX)　　指定要使用的排序顺序表
LANGID(*JOBRUN)　　与 SRTSEQ 一起使用，指定排序序列的语言识别程序
TRUNCNBR(*YES)　　指定当发生数字溢出时采取的动作

如果需要，那么 CRTRPGMOD 命令产生一个绝大部分与 CRTBNDPGM 命令所产生的相同的清单。（由 CRTRPGMOD 建立的清单绝不会包括代码生成和连接单元）。

关于使用编译清单的信息，参见 2.2.2 节的“使用编译清单”。

在附录 D 中给出了一个样本编译清单，即附录 1.4 节中的“编译清单”。

6.1.1.1 使用 CRTRPGMOD 默认值建立模块

在本例中使用 CRTRPGMOD 命令及其参数默认值建立一个名为 INCALC 的模块目标。INCALC 的源语句如图 22 所示。

1.为了建立模块目标，键入：

CRTRPGMOD　MODULE(MYLIB/INCALC)
SRCFILE(MYLIB/QRPGLESRC)

模块将以命令中指定的名字 INCALC 建立在库 MYLIB 中。模块的源语句是在库 MYLIB 文件 QRPGLESRC 中的源成员 INCALC。

该模块目标可以使用语句视图进行调试，并产生模块的编译清单。

2.进入以下的 CL 命令之一来看编译清单

DSPJOB 并选择选项 4(显示假脱机文件)

WRKJOB

WRKOUTQ 队列名

WRKSPLF

```
*============================================================*
* 模 块 名：      INCALC                                      *
* 有关文件：      N/A                                          *
* 有关程序：      TRNSRPT                                      *
* 说     明：它用参数列表中的字段计算交易的收入。在所有计算完成  *
*           之后返回到 TRNSRPT。                              *
*============================================================*
C     *ENTRY        PLIST
C                   PARM                    Prod           10
C                   PARM                    Qty             5 0
C                   PARM                    Disc            3 2
C                   PARM                    Inc            11 1

C                   SELECT
C                   WHEN      Prod = 'Model 1'
C                   EVAL      Inc = 1500 * Qty * Disc
C                   WHEN      Prod = 'Model 2'
C                   EVAL      Inc = 3500 * Qty * Disc
C                   WHEN      Prod = 'Model 8'
C                   EVAL      Inc = 32000 * Qty * Disc
C                   WHEN      Prod = 'Model 12'
C                   EVAL      Inc = 28000 * Qty * Disc
C                   OTHER
C                   EVAL      Inc = 0
C                   ENDSL

*  Return to the  caller TRNSRPT.                           *
C                   RETURN
```

图 22　成员 INCALC 的源语句

6.1.2 建立一个有源语句调试的模块

在此例中，建立了一个可以用源语句调试的程序 ILE RPG/400 模块目标。此模块的源语句如图 23 所示。

为建立模块目标，键入：

```
CRTRPGMOD   MODULE(MYLIB/TRNSRPT)SRCFILE(MYLIB/QRPGLESRC)
              DBGVIEW(*SOURCE)
```

该模块建立在库 MYLIB 中，名字与源成员名相同，即 TRNSRPT。该模块目标可使用源语句视图进行调试。关于其它有效视图的信息，参见 3.1.2 节中的“ 准备一个可以 Debug 的程序”。

命令也将产生 TRNSRPT 模块的编译清单。

```
 *=============================================================*
 * 模 块 名：     TRNSRPT                                      *
 * 有关文件：     TRNSDTA（物理文件）                          *
 * 有关程序：     INCALC  （计算程序）                             *
 * 说     明：这个程序从物理文件 TRNSDTA 中读每个交易记录，它调用  *
 *           做计算的 INCALC 并返回一个值，然后打出交易记录。      *
 *=============================================================*
FTRNSDTA   IP   E            DISK
FQSYSPRT   O    F   80       PRINTER
 *
ITRNREC       01

 *  Call the calculation procedure INCALC.                     *
C   01             CALLB     'INCALC'
C                  PARM                  PROD
C                  PARM                  QTY
C                  PARM                  DISCOUNT
C                  PARM                  INCOME          11 1
C
OQSYSPRT   D    01                  1
O                                        12 'PRODUCT: '
O                      PROD              25
O                                        40 'QUANTITY: '
O                      QTY               45
```

```
0                                          60 'INCOME: '
0                        INCOME        2   75
```

图 23　模块 TRNSRPT 的源程序

模块 TRNSDTA 的 DDS 如图 24 所示。

```
A*****************************************************************
A* 有关文件：TRNSRPT                                              *
A* 说     明：这是一个物理文件 TRNSDTA，它有一个记录格式 TRNREC。 *
A*****************************************************************
A* PARTS TRANSACTION FILE -- TRNSDTA
A          R TRNREC
A            PROD          10S 0       TEXT('Product')
A            QTY            5S 0       TEXT('Quantity')
A            DISCOUNT       3S 2       TEXT('Discount')
```

图 24　TRNSDTA 的 DDS

6.1.3 其他例子

关于建立模块的更多的例子，参见：

2.4.4 节中的“样本服务程序”是为服务程序建立模块的例子。

2.4.4.2 节中的“连接一个程序”是建立供服务程序使用的模块的例子。

2.5.7.1.1 节中的“为运行时数组动态分配存储空间”是为运行时数组动态分配存储空间而建立模块的例子。

3.1.13 节中的“调试源程序样本的例子”，是建立在样本调试程序中使用的 RPG 和 C 模块的例子。

6.1.4 连接 ILE RPG/400 模块

在 ILE RPG/400 中，过程的各项边界是：

LR 语义的作用域

打开文件的作用域

周期的作用域

既然每个　ILE RPG/400 模块有且仅有一个过程，那么过程的作用域与模块的作用域是相同的，所以当把两个模块连接在一起时，程序有多个周期。每个过程一个周期。

注意：在其它的 ILE 语言中过程作用域也许与模块作用域不相同，因为他们

的概念及处理的方法是不同的。

6.1.5 相关的 CL 命令

模块可以使用的 CL 命令：

显示模块(DSPMOD)
修改模块(CHGMOD)
删除模块(DELMOD)
处理模块(WRKMOD)

有关这些命令的详细信息见 CL 参考手册。

6.2 把模块连接成程序

连接是把一个或多个模块以及可选的服务程序，连起来建立一个可运行的 ILE 程序的过程，并解释它们之间传递的符号。做这些连接和解释的系统代码叫做 AS/400 系统连接程序。

作为连接处理的一部分，必须有一个过程被标识为启动过程，或叫程序入口过程。当程序被调用时，程序入口过程接收从命令行传来的参数，并得到程序初始控制权。与程序入口过程相关的用户代码是用户的入口过程。

每个 ILE RPG/400 模块都隐含地包括一个程序入口过程。但是对其它的 ILE 语言来说可能不是这样的。例如，ILE C/400 模块只有当包含 main()函数时才有程序入口过程。

图 25 给出了程序目标内部结构的概念。它显示了通过连接 TRNSRT 和 INCALC 两个模块而建立的程序目标 TRPT。TRNSRPT 为入口模块。

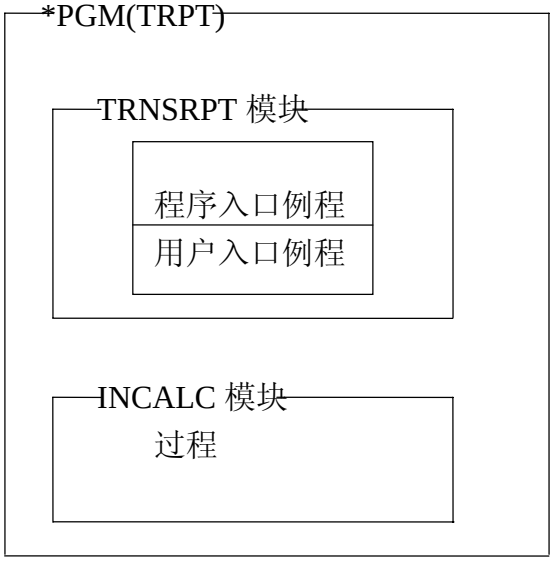


图 25　程序 TRPT 的结构

在一个连接的目标内，过程可以通过静态过程调用相互联系，这些连接调用要比外部调用快。因此，由一个有多个连接调用的连接程序组成的应用程序，要比与之相似但是包含有之间是外部调用的若干独立程序组成的应用程序性能要快。

为了把模块连接在一起，可把它们连接成为服务程序（类型为*SRVPGM）。服务程序允许在程序模块外对模块进行编码和维护。公共子例程可以建立为服务程序，如果对子例程修改的话，那么这些修改可以通过再次连接服务程序来完成，而使用这些公共子例程的程序就不必重新建立。关于服务程序的详细信息参见第七章 2.4 节中的“建立服务程序”。

关于连接处理过程和连接信息，参见 ILE 概念。

6.2.1 使用 CRTPGM 命令

建立程序(CRTPGM)命令由一个或多个以前建立的模块来建立程序目标。如果有要求的话，还包括一个或多个服务程序。还可以连接由任何 ILE 建立模块命令建立的模块，这些命令如 CRTRPGMOD，CRTCMOD，CRTCBLMOD 及 CRTCLMOD。

注意：需要的模块和服务程序必须事先建立。

在使用 CRTPGM 命令建立模块前，应该：

1.为程序命名

2.指定一个或多个要连接成程序目标的模块，如果需要的话，还应包括要连接的服务程序。

3.标识出哪个模块中有程序入口过程。

通过 CRTPGM 命令的 ENTMOD 参数指出哪个模块包含程序入口过程。默认值为 ENTMOD(*FIRST)，表示在 MODULE 参数列表中的第一个模块是入口过程模块。

如果把多于一个的 ILE RPG/400 模块连接在一起，那么应该指定*FIRST 或者某个模块名作为程序入口过程的模块名。当只连接一个模块或者连接几个模块，但只有一个模块包含程序入口过程时，可以指定 ENTMOD(*ONLY)。例如，如果连接一个 RPG 模块和一个没有 main()函数的 C 模块，那么可以指定 ENTMOD(*ONLY)。

4.指定程序要使用的活动组

如果程序没有特别要求或者你不能确定应使用哪一活动组，可以指定名为 QILE 的命名活动组。让应用程序在自己的活动组中运行是很好的想法。因此，可以在应用之后命名活动组。

注意：CRTPGM 的默认活动组是*NEW。这表示应用程序会在自己的活动组中运行，并且活动组会在程序结束时终止。也就是说无论是否置 LR 为 ON，程序会在下一次调用到它时有其自身数据的新的拷贝。关于活动组的详细信息参见 2.5.6.1 节中的“指定一个活动组。”

要用 CRTPGM 命令建立程序，执行以下步骤：

1.输入 CRTPGM 命令。

2.输入命令参数的适当值。

表 4 列出了 CRTPGM 命令的参数及其默认值。关于 CRTPGM 命令及参数的完整描述，参见 CL 参考手册。

表 4　CRTPGM 命令的参数及其默认值

参数组	参数（默认值）
标识	PGM(库名/程序名)
	MODULE(*PGM)
程序访问	ENTMOD(*FIRST)
连接	BNDSRVPGM(*NONE)
	BNDDIR(*NONE)
运行时	ACTGRP(*NEW)
其它	OPTION(*GEN *NODUPPROC *NODUPVAR *WARN *RSLVREF)
	DETAIL(*NONE)
	ALWUPD(*YES)
	ALWRINZ(*NO)
	REPLACE(*YES)
	AUT(*LIBCRTAUT)
	TEXT(*ENTMODTXT)
	TGTRLS(*CURRENT)
	USRPRF(*USER)

一旦你发出了 CRTPGM 命令，系统应执行以下操作：

1.把列出的模块拷贝到程序目标中，并把服务程序连接到程序目标上。

2.标识包含程序入口过程的模块并且定位在此模块中的第一个入口。

3.按所列出的次序检查模块，并检查第一个入口参数与模块出口参数是否相匹配。

4.返回到第一个模块并定位在下一个入口参数。

5.解析第一个模块中的所有入口参数。

6.继续下一个模块并解析所有入口参数。

7.解析其后的每一个模块中的所有入口参数，直到所有的入口参数都被解析。

8.如果某一个入口参数不能与一个出口参数相一致，则连接过程终止且不产生程序目标。

9.一旦所有的入口解析完毕，则连接过程终止，并建立程序目标。

注意：如果指定一个变量作为出口（使用 EXPROT 键字），有可能变量名会是连接程序目标内另外过程中某个变量名。

在这种情况下，结果可能是不可预测的，关于如何处理这种情况请参考 ILE 概念。

6.2.1.1 连接多个模块

本例显示了如何用 CRTPGM 命令连接两个 ILE　RPG/400 模块成为程序 TRPT。这个程序执行以下操作：

模块 TRNSRPT 从文件 TRNSDTA 中读每个交易记录，然后用连接过程调用 CALLB 调用模块 INCALC。

INCALC 计算每笔交易的收入，INCALC 返回到 TRNSRPT。然后 TRNSRPT 打印交易记录。

TRNSRPT，INCALC 和 TRNSDTA 的源语句如 2.3.1.2 节中的图 23 和 2.3.1.1.1 节中的图 22，和 2.3.1.2 节中的图 24 所示。

1.首先建立模块 TRNSRPT，键入：

```
CRTRPGMOD　MODULER(MYLIB/TRNSRPT)
```

2.然后建立模块 INCALC，键入：

```
CRTRPGMOD MODULE(MYLIB/INCALC)
```

3.建立程序目标，键入：

```
CRTPGM PGM(MYLIB/TRPT)MODULE(TRNSRPT　INCALC)
　　ENTMOD(*FIRST)　ACTGRP(TRPT)
```

CRTPGM 命名在库 MYLIB 中建立程序目标 TRPT。

注意 TRNSRPT 在 MODULE 的参数中被第一个列出；这表示它包含程序入口过程。如果 INCALC 被第一个列出那么它将包含程序入口过程；但是 TRNSRPT 没有被 INCALC 调用将不会运行。

程序将在命名活动组 TRPT 中运行。这就确保了其它的程序不能影响到它的资源。

6.2.2 附加的例子

关于建立程序的其他的例子，参见：

2.4.4.2 节中的“连接到一个程序”，是连接一个模块和服务程序的例子。

3.1.13 节中的“调试例子的样本源程序”是建立包含 RPG 和 C 模块的程序的例子。

6.2.3 有关的 CL 命令

对程序操作可使用以下的 CL 命令：

修改程序(CHGPGM)
删除程序(DLTPGM)
显示程序(DSPPGM)
显示程序参考(DSPPGMREF)
更新程序(UPDPGM)
处理程序(WRKPGM)

关于这些命令的详细资料，参见 CL 参考手册。

6.3 使用连接清单

连接过程可以产生一个清单，它描述了使用的资源，遇到的目标和符号，以及在连接过程中解决或未解决的问题。该清单是作为键入 CRTPGM 命令后作业产生的假脱机文件。该命令在默认情况下是不产生这些信息的，但是可以选择 DETAIL 参数的值来产生三种不同明细的清单：

*BASIC
*EXTENDED
*FULL

根据参数 DETAIL 值的不同，连接清单会包含以下内容：

表 5.基于 DETAIL 参数的连接清单单元

单元名	*BASIC	*EXTENDED	*FULL
命令选项总计	x	x	x
简要总计表	x	x	x
扩展总计表		x	x
连接信息清单		x	x
交叉引用清单			x
连接统计			x

如果连接不成功的话，这个清单中的信息可以帮助你诊断问题所在，或对连接过程中遇到的情况给出反馈信息。如果希望把这个清单做为文件保存，那么可以使用拷贝假脱机文件(CPYSPLF)命令。把这个清单复制成一个数据库文件。

注意：CRTBNDRPG 命令不会产生连接清单，然而，如果在连接阶段发生了连接错误，那么会把错误记录在工作日志中，并且编译清单会包含与此有关的信息。

关于基本连接清单的例子，参见 2.4.4.4 节中的“连接清单样本”。

关于连接清单的详细资料，参见 ILE 概念。

6.4 修改模块或程序

由于改进或维护的原因，可能要修改 ILE 目标。可以使用调试信息或由 CRTPGM 命令产生的连接清单来找出哪些需要修改。此信息可以确定哪些模块需要修改，哪些过程或字段需要经常修改。另外，可能希望修改模块或程序的优化级和可视性。这在调试程序或模块，或当把一个程序做为产品时经常遇到的。与重新建立目标相比，这种修改可以更快的进行，并占用较少的系统资源。

最后，当你做完应用程序时，可能希望缩减程序的尺寸。ILE 程序由于包括了一些附加数据，这使它们与类似的 OPM 程序相比要大一些。

以上每项都需要修改不同的数据。你需要的资源对 ILE 程序来讲可能是无效的。

以下的单元告诉你如何：

更新程序
修改优化级
修改可视性
缩减目标尺寸

注意：在本单元里目标是用来表示 ILE 模块或 ILE 程序的。

6.4.1 使用 UPDPGM 命令

通常，可以根据需要替换模块来修改程序。不必重新建立程序。当你为其它地方提供应用程序时，这是很有帮助的。只需发出修订过的模块，然后接收地就可以用 UPDPGM 或 UPDSRVPGM 命令更新应用程序。

UPDPGM 命令处理程序和模块。这条命令的参数与 CRTPGM 命令的参数非常相似。例如，要替换程序中的模块，键入 MODULE 参数的模块名和库名。

UPDPGM 命令要求被替换的模块是程序建立时库中的那个模块。你可以指定替换所有的模块，或只替换某些模块。

UPDPGM 命令要求模块目标是已经存在的。当你已经用编译和连接二步建立程序后，就很容易地使用这条命令。因为模块目标已经存在，只需在发出命令时指定模块名和所在库就可以了。

要更新用 CRTBNDRPG 命令建立的程序，必需保证修改过的模块在 QTEMP 库中。这是因为当 CRTBNDRPG 命令发出时使用的临时模块都是在 QTEMP 库中的。只要模块已经在 QTEMP 中了，就可以使用 UPDPGM 命令来替换该模块。

6.4.2 修改优化级

目标优化即根据编译的代码，为了使运行时的性能更好而要做的一些处理，也会做出必要的修改。通常要求的优化级越高，建立目标的时间越长。在运行时，优化过的程序或服务程序比相应的未优化的程序或服务程序运行的速度要快。

但是，对于较高级别的优化，当在调试显示或从异常中恢复时字段的值可能是不准确的。另外，优化了的代码可能会把源调试用的断点的位置作改动，因为优化做了修改可能要重新组成或删除一些语句。

为了确保字段的内容反映它们的最近当前值，特别是在异常恢复后，可以在相应的定义规范表中使用 NDOPT 键字。有关的信息，请参见 3.2.2.4 节中的“优化考虑”。

在调试时要防止发生这个问题，可以在调试程序时不选优化就可以在程序调试时准确地显示模块中的字段等信息，以后把程序投入运行时，再提高优化级以提高程序的效率。

要确定程序目标目前的优化级别，使用 DSPPGM 命令。这个命令显示了当前优化级。要修改程序的优化级，使用 CHGPGM 命令。优化程序参数可以指定以下值：*FULL，*BASIC，*NONE。对建立命令的 OPTIMIZE 参数可以指定相同的值。当该命令运行时程序自动地按指定的级别重新建立。

类似地，确定模块当前的优化级别，使用 DSPMOD 命令。这条命令显示的第 2 页指出了当前的优化级。要修改优化级别，使用 CHGMOD 命令。然后用 UPDPGM 或 CRTPGM 命令重新建立程序。

6.4.3 删去可视性

可视性包括两类数据，存储在一个目标内，允许修改目标而不必重新编译源程序。增加了某些数据使目标的尺寸变大。你可以删除这些数据从而缩减目标的大小。但是一旦这些数据删除了，可视性也就同时被删除。必须重新编译源语句并重建程序来替换这些数据。这两种数据是：

建立数据：由*CRTDTA 表示。这种数据用于把代码翻译成机器指令是必要的。你要自己修改优化级，目标必须包含这种数据。

调试数据：由*DBGDTA 表示。这种数据对于调试目标是必要的。

可以使用 CHGPGM 命令或 CHGMOD 命令分别从程序或模块中删除这类数据。也可以两种都删除，也可以都不删除。删除所有的可视性使目标尺寸减至最小（不包括压缩）。如果不重新建立的话，目标就没有可能被修改。因此，必须确保拥有建立程序所需要的所有源码或者拥有带 CRTDATA 的类似的程序目标。要重新建立程序，必须有访问源代码的权利。

6.4.4 缩减目标的大小

ILE 程序或模块的建立数据(*CRTDTA)参数可能会占目标大小的一半以上。通过删除或压缩这些数据，可以明显地减少程序需要的二级存储空间。

如果删除这些数据，那么要保证有建立程序所需的所有源程序，或有 CRTDATA 参数的程序目标，否则的话不能修改目标。

还有一种方法压缩目标，即使用压缩目标命令(CPROBJ)。压缩的目标与非

压缩目标相比占有较少的系统存储空间。如果调用压缩的目标，那么包含可运行代码的那部分目标自动地解压缩。也可以使用解压缩目标(DCPOBJ)命令。

关于这些 CL 命令的详细资料，参见 CL 参考手册。

第七章　建立服务程序

本章给出下面的内容：

服务程序概念
建立服务程序的策略
CRTSRVPGM 命令简介
服务程序的例子

7.1 服务程序概述

服务程序是一个连接程序(类型为*SRVPGM)，它由一系列过程组成，这些过程可以由其他连接程序中的过程调用。

服务程序是典型的公用函数，它在应用程序内和应用程序之间频繁调用。例如，ILE 编译程序使用服务程序来提供运行期间服务，象数学函数和输入/输出子例程。使用服务程序的好处是，可重复使用，维护简单，减少存储空间。

服务程序与程序的区别，表现在两个方面：

它不包括程序入口过程。这意味着不能用 CALL 操作调用服务程序。

服务程序通过引用，连接到其它程序或服务程序中。

当把服务程序连接到程序时，服务程序的内容并没有拷贝到程序中，而是服务程序的连接信息被连接到程序中。和连接模块到程序中的静态连接相对应，这种连接叫做“引用连接”。

因为服务程序是通过引用连接到程序上，所以可以使用 CALLB 操作调用服务程序的出口过程。由于是通过引用连接，所以调用有一定数量的开销，但是，随后对它的过程的任何调用都比程序调用快。

服务程序中的一系列出口是，它所提供的服务的接口。可以使用显示服务程序 CDSPSRVPGM 命令或服务程序清单，来查看哪些变量用于过程调用。

7.2 建立服务程序的策略

当建立服务程序时，应当注意：

1.是否希望以后更新该程序。

2.是否某些更新会引起接口的改变（即使用的进口和出口）。

如果服务程序的接口改变了，那么就必须重新连接那些与原服务程序连接的程序。如果所做的修改是向上兼容的，并且是用连接语言建立的服务程序，那么也许可以减少重新连接的数量。在这种情况下，当更新连接语言源程序，来标识新的出口后，只需重新连接那些使用他们的程序。

连接语言让你控制服务程序的出口，当你要做以下的事情时这种控制是很有用的：

给服务程序用户包装某些服务程序产品
改正问题
增强功能
减小因为修改对应用程序用户造成的影响

关于使用连接语言建立服务程序的例子，参见 2.4.4 节中的“样本服务程序”。

关于连接语言，出口处理及其它服务程序概念，参见 ILE 概念。

7.3 使用 CRTSRVPGM 命令建立服务程序

使用建立服务程序命令(CRTSRVPGM)建立服务程序。任何 ILE 模块都可以被连接到服务程序中。在你用它来建立服务程序前，模块必须是已经存在的。

表 6 列出了 CRTSRVPGM 命令的参数及默认值。关于 CRTSRVPGM 命令及其参数的完整描述，参见 CL 参考手册。

表 6. CRTSRVPGM 命令的参数以及他们的默认值

参数组	参数(默认值)
标识	SRVPGM(库名/服务程序名)
	MODULE(*SRVPGM)
程序访问	EXPORT(*SRCFILE)
	SRCFILE(*LIBL/QSRVSRC)
	SRCMBR(*SRVPGM)
连接	BNDSRVPGM(*NONE)
	BNDDIR(*NONE)
运行时	ACTGRP(*CALLER)
其它	OPTION(*GEN *NODUPPRC *NODUPVAR *WARN *RSLVREF)
	DETAIL(*NONE)
	ALWUPD(*YES)
	ALWRINZ(*NO)
	REPLACE(*YES)
	AUT(*LIBCRTART)
	TEXT(*ENTMODTXT)

TGTRLS(*CURRENT)
USRPRF(*USER)

关于使用 CRTSRVPGM 命令的例子，参见 2.4.4.1 节中的建立服务程序。

7.3.1 修改服务程序

可以用程序目标相同方法更新或修改服务程序。也就是说，你可以：

更新服务程序(使用 UPDSRVPGM)
修改优化级(使用 CHGSRVPGM)
删除可视性(使用 CHGSRVPGM)
缩减尺寸(使用 CPROBJ)

关于以上几点的详细资料，参见 2.3.4 节中的“修改模块或程序”。

7.3.2 相关的 CL 命令

以下的 CL 命令也可以对服务程序使用：

修改服务程序(CHGSRVPGM)
显示服务程序(DSPSRVPGM)
删除服务程序(DLTSRVPGM)
更新服务程序(UPDSRVPGM)
处理服务程序(WRKSRVPGM)

7.4 样本服务程序

以下的例子显示了如何建立服务程序 CNTTOHEX，该程序把字符串转换为与之等价的十六进制形式。有两个参数需要传递给服务程序：

1.要转换的字符型字段(Instring)

2.包含两个字节的十六进制数的字符型字段(Hexstring)

Hexstring 字段用来存储转换结果，并且也表示要转换的字符串的长度。例如，如果被传递的是 30 个字符长的字符串，但是你只想转换前 10 个，那么你只需传递 20 个字节长(2 乘 10)的串作为第二个参数。根据所传递的字段的长度，服务程序决定要处理的长度。

图 26 给出了服务程序的源语句。

服务程序中过程的基本逻辑如下所示：

1.使用操作描述来确定传递参数的长度。

2.确定要转换的长度：该长度值是字符串长度的最小值或 16 进制字段的二分之一。

3.使用子例程 GetHex 把字符串中的每个字符转换成与之相等价的两字节十六进制值。

4.过程返回到调用者。

该服务程序使用了操作描述，它是 ILE 的构成部分，当事先不知道所传递的参数的确切属性时使用它，在本例中是长度。当在 CALLB 操作上指定了操作扩展符(D)，那么在调用过程时就建立操作描述。

要使用操作描述，服务程序必须调用 ILE 的可连接 API，CEEDOD(接收操作描述)。这个 API 要求必须定义 CALLB 操作的某些参数。但它是提供所需信息的最后一个参数，即长度。关于操作描述的详细信息，参见 2.6.3.3 节中的“使用操作描述”。

```
 *===============================================================*
 * CVTTOHEX—把输入串转换成十六进制的输出串。                     *
*                                                                 *
* 注：必须传递操作说明符。                                       *
*                                                                 *
*===============================================================*
 *---------------------------------------------------------------*
* 程序参数：                                                     *
* 1.输入：字符串（N）                                           *
* 2.输入：十六进制（2n）                                        *
*---------------------------------------------------------------*
D InString        S          16383
D HexString       S          32766
*---------------------------------------------------------------*
* CEEDOD 的二进制参数（从操作说明符得来）                              *
*---------------------------------------------------------------*
D ParmNum         S             9B 0
D DescType        S             9B 0
D DataType        S             9B 0
D DescInfo1       S             9B 0
D DescInfo2       S             9B 0
D InLen           S             9B 0
D HexLen          S             9B 0

*---------------------------------------------------------------*
* 程序所用的其它字段                                           *
*---------------------------------------------------------------*
```

```
D HexDigits       C                   CONST('0123456789ABCDEF')
D BinDs           DS
D   BinNum                      4B 0 INZ(0)
D   BinChar                     1    OVERLAY(BinNum:2)
D HexDs           DS
D   HexC1                       1
D   HexC2                       1
D InChar          S             1
D Pos             S             5P 0
D HexPos          S             5P 0

C     *ENTRY        PLIST
C                   PARM                    InString
C                   PARM                    HexString
  *---------------------------------------------------------------*
 * 用操作说明符来确定传递的参数长度                                *
 *---------------------------------------------------------------*
C                   CALLB     'CEEDOD'
C                   PARM      1             ParmNum
C                   PARM                    DescType
C                   PARM                    DataType
C                   PARM                    DescInfo1
C                   PARM                    DescInfo2
C                   PARM                    InLen
C                   PARM                    *OMIT
C                   CALLB     'CEEDOD'
C                   PARM      2             ParmNum
C                   PARM                    DescType
C                   PARM                    DataType
C                   PARM                    DescInfo1
C                   PARM                    DescInfo2
C                   PARM                    HexLen
C                   PARM                    *OMIT
  *---------------------------------------------------------------*
 * 确定处理的长度（最少为输入长度和十六进制长度一半的和）         *
 *---------------------------------------------------------------*
C                   IF        InLen > HexLen / 2
C                   EVAL      InLen = HexLen / 2
```

```
C                   ENDIF

 *-------------------------------------------------------------*
 * 对输入字串的每个字符，转换为 2 字节的 16 进制表示。          *
 * (例如'5'转换成'F5')                                         *
 *-------------------------------------------------------------*
C                   EVAL      HexPos = 1
C                   DO        InLen         Pos
C                   EVAL      InChar = %SUBST(InString:Pos:1)
C                   EXSR      GetHex
C                   EVAL      %SUBST(HexString:HexPos:2) = HexDs
C                   EVAL      HexPos = HexPos + 2
C                   ENDDO

 *-------------------------------------------------------------*
 * 做好后，返回调用程序。                                      *
 *-------------------------------------------------------------*
C                   RETURN
  *============================================================*
 * GetHex—把 InChar 转换成 HexDs 的子程序                      *
 *                                                             *
 * 用 16 去除，分开二个 16 进制数字，商为第一个数字，          *
 * 余数为第二个数字。                                          *
 *============================================================*
C     GetHex        BEGSR
C                   EVAL      BinChar = InChar
C     BinNum        DIV       16            X1              5 0
C                   MVR                     X2              5 0
 *-------------------------------------------------------------*
 * 用 16 进制数字（加 1）来取 16 进制字符（012…CDEF）的子串。  *
 *-------------------------------------------------------------*
C                   EVAL      HexC1 = %SUBST(HexDigits:X1+1:1)
C                   EVAL      HexC2 = %SUBST(HexDigits:X2+1:1)
C                   ENDSR
```

图 26.服务程序 CVTTOHEX 的源语句

当设计这个服务程序时，用连接语言来确定接口，以便日后很容易地就能更

新。图 27 显示了定义服务程序 CVTTOHEX 出口所需的连接语言。

```
STRPGMEXP SIGNATURE(CVTHEX)
   EXPORT SYMBOL('CVTTOHEX')
ENDPGMEXP
```

图 27.CVTTOHEX 的连接程序语言的源语句

STRPGMEXP 中的参数 SIGNATURE 指出了服务程序提供的接口。这种情况下，在连接语言标识的出口是做为接口使用，任何与 CVTTOHEX 连接的程序都要使用这个标识。

连接语言的 EXPORT 语句标识了服务程序的出口。每一个可以调用的过程，都需要有一个出口。在本例中，程序所含模块只有一个过程，因此，只需要一个 EXPORT 语句。

关于连接程序及标识的详细的资料，参见 ILE 概念。

7.4.1 建立服务程序

要建立服务程序 DVTTOHEX，按以下步骤进行：

1.由 2.4.4 节中图 26 中的源语句建立模块 CVTTOHEX :

```
CRTRPGMOD MODULE(MYLIB/CVTTOHEX) SRCFILE(MYLIB/QRPGLFSRC)
```

2.由模块 CVTTOHEX 和 2.4.4 节中图 27 所示的连接语言建立服务程序。

```
CRTSRVPGM SRVPGM(MYLIB/CVTTOHEX) MODULE(*SRVPGM)
                EXPORT(*SRCFILE) SRCFILE(MYLIB/QSRVSRC)
                SRCMBR(*SRVPGM)
```

上面命令的最后三个参数标识了服务程序的有效出口。在这里，它依据库 MYLIB 文件 QSRVSRC 中的成员 CVTTOHEX 里的源语句。注意在这里不需要连接目录，因为建立服务程序所需的所有模块已经在 MODULE 参数中指定了。

服务程序 CVTTOHEX 建立在库 MYLIB 中。可以使用语句视图来调试它，这是由 CRTRPGMOD 命令中 DBGVIEW 参数的默认值决定的，不产生连接清单。

7.4.2 连接到程序中

要完成本例，我们要建立一个“应用程序”，它由程序 CVTHEXPGM 与服务程序相连接组成。它把一个七个字符长的字符串两次传给 CVTTOHEX，一次十六进制串长为 10(也就是转换 5 个字符），另一次十六进制串长为 14，也就是实际的长度。

注意，程序 CVTHEXPGM 的目的只是显示一下服务程序 CVTTOHEX 的用法。在实际的应用中， CVTTOHEX 的调用者与测试相比，有更重要的目的。另

外服务程序是供多个其它程序使用的，或被某些程序多次使用；否则，也就不把它设计为服务程序。

1.由图 28 所示的源程序建立模块，键入：

CRTRPGMOD MODULE(MYLIB/CVTHEXPGM)
SRCFILE(MYLIB/QRPGLESRC)

2.建立程序，键入：

```
CRTPGM PGM(MYLIB/CVTHEXPGM)
       BNDSRVPGM(MYLIB/CVTTOHEX)
       DETAIL(*BASIC)
```

当 CVTHEXPGM 建立后，它包括了识别与服务程序交互作用的接口信息，这与 CVTTOHEX 的连接语言中所反映的相同。

3.调用程序，键入：

```
CALL    CVTHEXPGM
```

在 CVTTOHEX 运行的过程中，系统作以下检查：

在 MYLIB 库中找到服务程序 CVTTOHEX

在 CVTHEXPGM 建立时，使用的公共接口在运行时仍然有效。

如果以上任何一条没有满足，则产生错误信息。

CVTHEXPGM 的输出如下所示。（输入字符串为'ABC123*'）

```
Result14++++++
Result10++
C1C2C3F1F2                10Character Output
C1C2C3F1F2F35C            14Character Output
```

```
*----------------------------------------------------------------*
* 这个程序测试服务程序 CVTTOHEX。                                 *
*                                                                *
* 1.用 7 个字符的输入串。                                         *
* 2.转换成 10 个字符的 16 进制串(由于结果字段对整个输入串          *
*   来说太小，所以仅用输入的头 5 个字符）。                       *
* 3.转换成 14 个字符的 16 进制串(由于结果字段够长，所以输入的      *
*   7 个字符都被转换）。                                         *
*----------------------------------------------------------------*

FQSYSPRT   O    F   80        PRINTER

D ResultDS        DS
D   Result14              1     14
```

```
D   Result10              1      10
D InString        S               7
D Comment         S              25

C                   EVAL       InString = 'ABC123*'

 *----------------------------------------------------------------*
 * 用 CALLB(D)传递字符串和 10 个字符的结果字段，操作扩展(D)        *
 * 每个传递的参数生成操作说明符，这是用 CVTTDHEX 调用的例程        *
 * 所必须的。                                                      *
 *----------------------------------------------------------------*
C                   EVAL       Comment = '10 character output'
C                   CLEAR                  ResultDS
C                   CALLB(D)   'CVTTOHEX'
C                   PARM                   InString
C                   PARM                   Result10
C                   EXCEPT

 *----------------------------------------------------------------*
 * 用 CALLB(D)传递字符串和 14 个字符的结果字段。                   *
 *----------------------------------------------------------------*
C                   EVAL       Comment = '14 character output'
C                   CLEAR                  ResultDS
C                   CALLB(D)   'CVTTOHEX'
C                   PARM                   InString
C                   PARM                   Result14
C                   EXCEPT
C                   EVAL       *INLR = *ON

OQSYSPRT   H    1P
O                                          'Result14++++++'
OQSYSPRT   H    1P
O                                          'Result10++'
OQSYSPRT   E
O                       ResultDS
O                       Comment            +5
```

图 28.测试程序 CVTHEXPGM 的源语句

7.4.3 更新服务程序

由于有连接语言，可以更新服务程序而不必重新编译程序 CVTHEXPGM。例如，可以在 CVTTOHEX 中加一个新的过程。

1.为新过程建立模块目标。

2.修改连接语言源语句以处理与新过程相关联的接口。这样要在已经存在的出口语句后添加新的出口语句。

3.重新建立服务程序把新模块连接到服务程序 CVTTOHEX 上。新的程序可以使用新的功能。因为旧的入口仍按原来的次序排列，所以仍然可以使用原有的程序。除非要改原有的程序，否则就不必重新编译它。

关于更新服务程序的详细资料，参见 ILE 概念。

7.4.4 样本连接清单

图 29 显示了 CVTHEXPGM 的样本连接清单，这是一个基本清单。关于连接清单的详细资料，参见 2.3.3 节中的“使用连接清单”和 ILE 概念。

```
                                            Create Program                                        Page      1
5763SS1 V3R1M0  940909                                                     MYLIB/CVTHEXPGM    AS400S01  09/22/94  23:24:00
 Program  . . . . . . . . . . . . . . . . . . . . . :   CVTHEXPGM
   Library  . . . . . . . . . . . . . . . . . . . . :     MYLIB
 Program entry procedure module . . . . . . . . . . :   *FIRST
   Library  . . . . . . . . . . . . . . . . . . . . :
 Activation group . . . . . . . . . . . . . . . . . :   *NEW
 Creation options . . . . . . . . . . . . . . . . . :   *GEN        *NODUPPROC  *NODUPVAR   *WARN       *RSLVREF
 Listing detail . . . . . . . . . . . . . . . . . . :   *BASIC
 Allow Update . . . . . . . . . . . . . . . . . . . :   *YES
 User profile . . . . . . . . . . . . . . . . . . . :   *USER
 Replace existing program . . . . . . . . . . . . . :   *YES
 Authority  . . . . . . . . . . . . . . . . . . . . :   *LIBCRTAUT
 Target release . . . . . . . . . . . . . . . . . . :   *CURRENT
 Allow reinitialization . . . . . . . . . . . . . . :   *NO
 Text . . . . . . . . . . . . . . . . . . . . . . . :   *ENTMODTXT
 Module      Library              Module      Library              Module      Library              Module      Library
 CVTHEXPGM   MYLIB
 Service                          Service                          Service                          Service
 Program     Library              Program     Library              Program     Library              Program     Library
```

```
 CVTTOHEX    MYLIB

 Binding                         Binding                         Binding                         Binding

 Directory   Library             Directory   Library             Directory   Library             Directory   Library

 *NONE

                                              Create Program                                                 Page     2

5763SS1 V3R1M0  940909                                                        MYLIB/CVTHEXPGM    AS400S01  09/22/94  23:24:00

                                           Brief Summary Table

 Program entry procedures . . . . . :   1

   Symbol    Type      Library     Object      Identifier

             *MODULE   MYLIB       CVTHEXPGM   _QRNP_PEP_CVTHEXPGM

 Multiple strong definitions  . . . . . :   0

 Unresolved references  . . . . . :   0


                          * * * * *  E N D  O F  B R I E F  S U M M A R Y  T A B L E  * * * * *

                                              Create Program                                                 Page     3

5763SS1 V3R1M0  940909                                                        MYLIB/CVTHEXPGM    AS400S01  09/22/94  23:24:00

                                           Binding Statistics

 Symbol collection CPU time . . . . . :              .016

 Symbol resolution CPU time . . . . . :              .004

 Binding directory resolution CPU time  . . . . . :              .175

 Binder language compilation CPU time . . . . . :              .000

 Listing creation CPU time  . . . . . :              .068

 Program/service program creation CPU time  . . . . . :              .234

 Total CPU time . . . . . :              .995

 Total elapsed time . . . . . :             3.531

                          * * * * *  E N D  O F  B I N D I N G  S T A T I S T I C S  * * * * *

*CPC5D07 - Program CVTHEXPGM created in library MYLIB.

                          * * * * *  E N D  O F  C R E A T E  P R O G R A M  L I S T I N G  * * * * *
```

图 29.CVTHEXPGM 的基本连接清单

第八章　运行程序

本章讲述了以下内容：

使用 CL CALL 命令运行程序和传递参数

从菜单驱动的应用程序运行程序

使用用户建立的命令运行程序

管理活动组

管理运行时存储空间

另外，可以用以下方法运行程序

程序员菜单。CL 程序设计包含此菜单的信息，启动应用开发管理工具(STRPDM）命令。ADTS/400：应用开发管理工具包含这条命令的信息。

QCMDEXC 程序。CL 程序设计包含这个程序的信息。

高级语言。第九章 2.6 节中的“程序和过程调用”。提供了从其它高级语言运行程序及调用程序和过程的资料。

8.1 使用 CL CALL 命令运行程序

可以使用 CL CALL 命令运行程序(类型为*PGM)。可以交互地或作为批处理作业的一部分，或在 CL 程序中使用该命令。如果需要提示，键入 CALL 然后按 F4(提示)键。如果需要帮助，键入 CALL 然后按 F1(帮助)键。

例如：要从命令行调用程序 EMPRPT，键入：

```
CALL    EMPRPT
```

指定的程序目标必须在某个库中，并且这个库要包含在库列表*LIBL 中。也可以象下面那样在 CL　CALL 命令中确切地指出库名：

```
CALL    MYLIB/EMPRPT
```

关于 CL CALL 命令的详细的资料参见 CL 参考手册。

一旦调用程序，OS/400 系统就执行程序中的指令。

8.1.1 使用 CL CALL 命令传递参数

当运行程序时，使用 CL CALL 命令的 PARM 选项来向 ILE 程序传递参数。

```
CALL    PGM(程序名)
          PARM(参数 1，参数 2，……，参数 n)
```

也可以只键入参数而不指定键字。

```
CALL        库/程序名(参数 1，参数 2，……，参数 n)
```

每个参数的值可以由 CL 程序变量或由以下之一来指定：

字符串常量

数字常量

逻辑常量

如果把参数传递给一个 ILE RPG/400 过程，这个过程作为入口过程，那么这个程序必须有且仅有一个*ENTRY PLIST。紧随其后的参数（在 PARM 语句中）必须与 CALL 命令所传递的参数一一对应。

关于如何处理参数的完整性描述参见 CL 参考手册中的 CALL 命令或 CL 程序设计中的“在程序间传递参数”。

例如，程序 EMPRPT2 在首次起动时要求有正确的口令传递给它，否则将不运行。图 30 显示了源程序。

1.要建立程序，键入：

CRTBNDRPG PGM(MYLIB/EMPRPT2)

2.要运行程序，键入：

CALL　MYLIB/EMPRPT2(HELLO)

当调用命令发出时，由命令传递的参数值储存起来，程序的参数 PSWORD 指向它的位置。程序检查 PSWORD 的内容与程序中存储的(’HELLO’)是否匹配。在这里，两个值是一样的，所以程序继续运行。

```
*============================================================*
* 程 序 名：     EMPRPT2                                      *
* 有关文件：     EMPMST（物理文件）                           *
* PRINT      （打印文件）                                     *
* 说    明：这个程序在输入的口令正确时，打印存储在 EMPMST     *
*           文件中的雇员信息。                               *
*           在命令行进入“CALL  库名/EMPRPT2（PSWORD）”，   *
*           这里 PSWORD 是这个程序的口令，口令是‘HELLO’。   *
*============================================================*
FPRINT     O    F   80        PRINTER
FEMPMST    IP   E           K DISK

IEMPREC        01
*------------------------------------------------------------*
* 参数列表项是在这个程序中规定，有一个 5 个字符长的参数 PSWORD。   *
*------------------------------------------------------------*
C     *ENTRY        PLIST
C                   PARM                    PSWORD         5
*------------------------------------------------------------*
```

```
 * 检查字段 PSWORD 中的内容是否为'HELLO'。如果不是, LR 和*IN99 为 ON。*
 * *IN99 控制打印信息。                                              *
 *-----------------------------------------------------------------*
C     PSWORD        IFNE      'HELLO'
C                   SETON                                          LR99
C                   ENDIF

OPRINT     H    1P                    2  6
O                                         50 'EMPLOYEE INFORMATION'
O          H    1P
O                                         12 'NAME'
O                                         34 'SERIAL #'
O                                         45 'DEPT'
O                                         56 'TYPE'
O          D    01N99
O                       ENAME             20
O                       ENUM              32
O                       EDEPT             45
O                       ETYPE             55
O          D    99
O                                         16 '***'
O                                         40 'Invalid Password Entered'
O                                         43 '***'
```

图 30　ILE RPG/400 程序

```
A*****************************************************************
A* 说    明：这是物理文件 EMPMST 的 DDS，它有一个记录格式 EMPREC。*
A*           包括公司中每个雇员的记录。                          *
A*****************************************************************
A*
A          R EMPREC
A            ENUM           5  0       TEXT('EMPLOYEE NUMBER')
A            ENAME         20          TEXT('EMPLOYEE NAME')
A            ETYPE          1          TEXT('EMPLOYEE TYPE')
A            EDEPT          3  0       TEXT('EMPLOYEE DEPARTMENT')
A            ENHRS          3  1       TEXT('EMPLOYEE NORMAL WEEK HOURS')
```

```
     A            K ENUM
```

图 31.EMPRPT 的 DDS

8.2 从一个菜单驱动的应用程序中运行程序

另一种运行 ILE 程序的方法是用菜单驱动的应用程序。工作站用户从菜单中选择选项，该菜单依次调用指定的程序。图 32 给出了一个应用程序菜单的例子。

```
*                                                                  *
*                          PAYROLL DEPARTMENT MENU                 *
*                                                                  *
*  Select one of the following:                                    *
*                                                                  *
*       1.  Inquire into employee master                           *
*       2.  Change employee master                                 *
*       3.  Add new employee                                       *
*                                                                  *
*                                                                  *
*                                                                  *
*                                                                  *
*                                                                  *
*                                                                  *
*                                                                  *
*                                                                  *
*                                                                  *
*                                                                  *
*                                                                  *
*                                                                  *
*  Selection or command                                            *
*  ===>                                                            *
*                                                                  *
*  F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel               *
*  F13=Information Assistant  F16=AS/400 main menu                 *
*                                                                  *
*                                                                  *
```

图 32.应用程序菜单的例子

图 32 中的菜单上每个选项调用一个独立的 ILE 程序。可以通过 STRSDA 命

令，然后选择选项 2（’设计菜单’）来建立一个菜单。

图 33 给出了上面 PAYROLL DEPARTMEMT MENE 显示文件的 DDS。源成员叫做 PAYROL，类型为 MNUDDS。这个文件是使用 SDA 建立的。

```
A* Free Form Menu: PAYROL
A*
A                                      DSPSIZ(24 80 *DS3                -
A                                             27 132 *DS4)
A                                      CHGINPDFT
A                                      INDARA
A                                      PRINT(*LIBL/QSYSPRT)
A          R PAYROL
A                                      DSPMOD(*DS3)
A                                      LOCK
A                                      SLNO(01)
A                                      CLRL(*ALL)
A                                      ALWROL
A                                      CF03
A                                      HELP
A                                      HOME
A                                      HLPRTN
A                                    1 34'PAYROLL DEPARTMENT MENU'
A                                      DSPATR(HI)
A                                    3  2'Select one of the following:'
A                                      COLOR(BLU)
A                                    5  7'1.'
A                                    6  7'2.'
A                                    7  7'3.'
A* CMDPROMPT  Do not delete this DDS spec.
A                                  019  2'Selection or command            -
A                                           '
A                                    5 11'Inquire'
A                                    5 19'into'
A                                    5 24'employee'
A                                    5 33'master'
A                                    6 11'Change'
A                                    6 18'employee'
A                                    6 27'master'
```

```
A                                   7  11'Add'
A                                   7  15'new'
A                                   7  19'employee'
```

图 33.应用程序菜单的数据描述

图 34 显示了在图 32 中给出的菜单程序的源语句。
源成员叫做 PAYROLQQ，类型为 MNUCMD。它也是用 SDA 建立的。

```
PAYROLQQ,1
0001 call RPGINQ
0002 call RPGCHG
0003 call RPGADD
```

图 34.菜单程序的源语句

可以键入以下命令来运行该菜单：

GO 库名/PAYROL

如果用户在菜单中输入 1,2 或 3，那么图 34 中的程序就相应地调用 RPGINQ，RPGCHG 或 RPGADD。

8.3 使用用户建立的命令运行程序

可以使用命令定义建立一个命令，用此命令运行程序。命令定义是一个目标(类型为*CMD)，该目标包括命令的定义（包括命令名，参数描述，和有效性检查等信息），并指出执行命令功能要用的程序。

例如可以建立命令 PAY，该命令调用程序 PAYROL，程序 PAYROL 是要运行的程序名，可以交互地或在批处理作业中使用该命令。并于使用命令定义的详细资料参见 CL 程序设计。

8.4 回答运行时的询问信息

当运行有 ILE RPG/400 过程的程序时，可能会产生运行时询问信息，没有处理异常的错误指示器或错误子例程时(*PSSR 或 INFSR)，就会出现这些信息。需要对询问信息作出应答程序才能继续运行下去。

可以把询问信息加到系统应答列表里从而自动地对这些信息作出应答。这种应答询问信息的方法比较适合于批处理程序，否则就需操作员来作出回答。

可以把以下的 ILE RPG/400 询问信息加到系统应答列表中。

注意：ILE RPG/400 的询问信息，编码以 RNQ 为前缀。要把询问信息添加到回答列表中应使用添加应答列表入口命令，键入：

ADDRPYLE　　顺序号　　信息编码

顺序号从 1 到 9999，反映了加到列表中入口的位置，信息编码是要添加的信息号码。对每条要添加的信息，重复这条命令。

使用修改作业（CHGJOB）命令（或其它 CL 作业命令）来显示作业对询问信息回答列表的使用。这样，应把询问信息回答（INQMSGRPY）属性指定为*SYSRPYL。

只用于由作业送出的询问信息，此作业的 INQMSGRPY 指定为*SYSRPYL。以下的 CL 命令中有 INQMSGRPY 参数。

修改作业（CHGJOB）

修改作业描述（CHGJOBD）

建立作业描述（CRTJOBD）

提交作业（SBMJOB）

也可以使用处理回答列表入口（WRKRPYLE）命令来修改或删除系统回答列表中的入口。关于 ADDRPYLE 和 WRKRPYLE 命令的详细资料参考 CL 参考手册。

8.5　结束 ILE 程序

当 ILE 程序正常结束，系统把控制返回给调用者。调用者可以是工作站用户或其它程序（例如菜单处理程序）。

如果 ILE 程序异常结束，并且该程序与它的调用者运行在不同的活动组中，那么就会产生逃逸信息 CEE9901，发出以下信息：Error message-id Caused program to end。

即错误信息使程序结束，并把控制返回给调用者。

使用监控信息命令(MONMSG)的 CL 程序可以监视这种异常。也可以用其它 ILE 语言程序来监视异常。

如果 ILE 程序与其调用者在同一活动组中运行而异常结束，那么是否产生信息取决于程序结束的原因。如果以功能检测结束，那么送出 CPF9999。如果由 RPG 过程产生异常，那么会产生以 RNX 为前缀的信息。

关于异常信息的详细资料，参见 3.2.1 节中的“ 异常情况处理概述”。

8.6　管理活动组

活动组是作业的子结构，它由分配来运行一个或多个 ILE 或 OPM 程序的系统资源组成(例如，存储空间，落实定义，及打开文件）。活动组可以使几个 ILE 程序在同一个作业中运行而彼此互不影响（例如，落实控制及复盖）。它的基本思想是在同一活动组内活动的所有程序组成一个协作的应用程序。

应该在程序建立时指定它在哪个活动中运行。活动组是在程序目标建立时由参数 ACTGRP 的指定值决定的。（OPM 程序总是在默认活动组中运行的。不可以修改他们的活动

组） 。一旦 ILE 程序（目标类型*PGM）被激活，他们就一直保持活动状态直到活动组被删除。

本单元的其余部分会告诉你如何指定一个活动组及怎样删除它。关于活动组的详细资料参见 ILE 概念。

8.6.1 指定活动组

要控制 ILE 程序在哪个活动组中运行是在建立程序（使用 CRTPGM 或 CRTBNDRPG）或服务程序（使用 CRTSRVPGM）时可以通过指定 ACTGRP 参数的值来实现的。

注意：如果使用 CRTBNDRPG 命令，而 DFTACTGRP 值为*NO 时，那么只能为 ACTGRP 指定一个值。

可以选择以下的值：

命名的活动组

命名的活动组允许把 ILE 程序和服务程序做为一个应用程序来管理。当调用第一个在建立时指定了活动组名字的程序时活动组建立。然后它被所有指定了相同活动组名的程序和服务程序使用。

当使用 CL 命令 RCLACTGRP 删除命名的活动组时该活动组结束。这条命令只有当活动组不再使用后才能使用。当它结束时，所有与活动组中的程序和服务程序相关联的资源返回给系统。

在 CRTBNDRPG 命令中，命名的活动组 QILE 是 ACTGRP 参数的默认值。但是，因为活动组要对很多作业响应，所以建议为这个参数指定不同的值。例如，你也许希望在应用程序命名后为活动组命名。

*NEW

当指定*NEW 时，只要程序被调用就建立新的活动组。系统为该活动组建立一个名字，在你的作业中，该名字是唯一的。

由*NEW 建立的活动组在与之相关联的程序结束时结束。因此，如果以 LR 为 OFF 来保持程序的活动状态而从程序中返回，那么就不能指定 ACTGRP 参数为*NEW。

注意：对于服务程序这个值是无效的。服务程序只能在命名的活动组或调用者的活动组中运行。

对于 CRTPGM 命令的 ACTGRP 参数*NEW 是其默认值。

如果使用 ACTGRP(*NEW)建立 ILE RPG/400 程序，那么就可任意多次地调用该程序而不必从较早的调用中返回。对每一次调用都会有该程序新的拷贝。每个新的拷贝会有它自己的数据，打开它自己的文件，等等。但是，必须保证有使其调用自身结束的办法。否则，就会不断的建立新的活动组并且该程序将永远不会返回。

*CALLER

程序或服务程序将在调用程序所在的活动组中被激活。如果以 ACTGRP(*CALLER)建立的 ILE 程序被 OPM 程序所调用，那么它将在 OPM 默认活动组(*DFTACTGRP)中被激活。

8.6.2 在 OPM 默认活动组中运行

当 OS/400 作业启动时，系统建立一个供 OPM 程序使用的活动组。活动组的符号为 *DFTACTGRP。不能删除 OPM 默认活动组。只在作业结束时由系统删除。

OPM 程序自动地在 OPM 默认活动组中运行。当出现以下情况时 ILE 程序也可以在 OPM 默认活动组中运行：

程序以 CRTBNDRPG 命令且 DFTACTGRP(*YES)建立。

程序以 ACTGRP(*CALLER)建立，并且调用程序是在默认活动组中运行。注意对于 CRTBNDRPG 命令如果指定了 DFTACTGRP(*NO)，那么只能指定 ACTGRP(*CALLER)。

注意：由*CALLER 决定在默认活动组中运行的程序，它所使用的资源直到作业结后才会被删除。

8.6.3 维护 OPM RPG/400 和 ILE RPG/400 程序的兼容性

如果你有由几个 RPG 程序组成的 OPM 应用程序，并且用下面的方法建立 ILE 应用程序，那么就可以确保转换后的应用程序象 OPM 应用程序那样工作：

1、使用 CVTRPGSRC 命令转换每个 OPM 源成员，注意要转换/COPY 成员。

有关的详细资料，参见附录 1.2.2 节中的“转换你的源程序”。

2、使用 CRTBNDRPG 命令，独立地编译和连接每个转换过的源成员成为程序目标，指定 DFTACTGRP(*YES)。

关于 OPM 兼容程序的详细资料，参见 1.3.1 节中的“策略 1：OPM 兼容 应用程序”。

8.6.4 删除活动组

当活动组被删除时，它的资源就被收回了。资源包括静态，存储空间及打开文件。当与一个*NEW 活动组相关联的程序返回到调用者时，该活动组被删除。

命名活动组（例如 ILE）是稳定的活动组。因为只有明确地删除或作业结束时，它们才被删除。与运行在命名活动组中程序相关联的存储空间，要到活动组被删除时才被释放。

OPM 默认活动组也是稳定活动组。在默认活动组中运行的 ILE 程序相关联的存储空间在 sign off（对于交互作业）或作业结束（对于批处理作业）时被释放。

如果有许多 ILE RPG/400 程序处于活动状况（即至少调用了一次），那么系统的存储空间可能被占满。因此，你应当避免让占用大量静态存储域的 ILE 程序在 OPM 默认的活动组中运行。因为只有当作业结束时，这些存储空间才会被收回。

注意：以 DFTACTGRP(*YES)建立的 ILE RPG/400 程序在它结束时，它的存储空间会被释放。

与服务程序相联系的存储空间只有当与之相关联的活动组结束时才会被收回。如果服务程序被调用而在默认活动组中运行，那么在作业结束时，它的资源被收回。

可以使用 RCLACTGRP 命令删除命名活动组。使用这个命令来删除那些不用的非默认活动组。该命令提供了删除所有符合条件的活动组或按名字删除一个活动组的选项。

关于 RCLACTGRP 的详细资料参见 CL 参考手册。关于 RCLACTGRP 与活动组的详细资料，参考 ILE 概念。

8.6.5 回收资源命令

回收资源(RCLRSC)命令用来回收不再活动的程序所用的资源。该命令根据程序建立方式的不同而有不同的工作方式。如果程序是 OPM 程序或是以 DFTACTGRP(*YES)建立的，那么 RCLRSC 命令将关闭打开的文件并释放静态存储空间。

对于以*CALLER 建立在 OPM 默认活动组中被激活的 ILE 程序，当发出 RCLRSC 命令时文件被关闭，并且存储空间重新初始化。但不释放存储空间。

对于与命名的活动组相关联的 ILE 程序，RCLRSC 命令不起作用。必须使用 RCLACTGRP 命令来释放命名活动组中的资源。

关于 RCLRSC 命令的详细资料，参见 CL 参考手册。关于 RCLRSC 与活动组的详细资料，参见 ILE 概念。

8.7 管理动态分配的存储空间

ILE 允许你在程序中通过堆管理而直接管理运行时的存储空间。堆是用于动态存储分配的存储区域。

应用程序所要求的动态存储空间数量取决于使用堆的程序和过程所处理的数据。可以通过使用 ILE 可连接的 API 来管理堆。

不须明确地管理运行时存储空间。但是，你可能希望使用动态分配的运行时存储空间。例如，如果不知道确切的数组或多次出现的数据结构有多大，可能就希望这样。可以定义数组或数据结构为 BASED，并且当程序确定了它们的大小时，为它申请实际的存储空间。

在系统中有两种有效的堆：默认的和用户建立的。本单元的其余部分解释了如何使用默认的堆来管理 ILE RPG/400 程序中的运行时存储空间。关于用户建立的堆和其它 ILE 存储空间管理概念的详细介绍，参见 ILE 概念。

8.7.1 管理默认的堆式空间

在一个活动组中对动态存储空间的首次请求导致了默认堆的建立，在这个堆里分配给它存储空间。其后的动态空间请求都从这个堆中分配。如果堆中的存储空间不能满足当前对动态存储的申请，那么堆空间被扩展，分配更多的存储空间。

已分配的存储空间直到明确地指出释放或直到堆被丢弃才会被释放。当拥有默认堆的活动组结时，该堆才会被丢弃。

在同一活动组中的程序都使用同一个默认的堆。如果一个程序存取的空间超出为它分配的存储空间，那么就会对其它程序产生问题。例如，假定有两个程序，程序 A 和程序 B 在同一个活动组中运行。有 10 个字节分配给程序 A，但是程序 A 修改了 11 个字节。如果多出的字节实际是分配给 B 的，那么程序 B 就会发生问题。

可以在一个活动组中分隔程序和过程的动态空间。即建立一个或多个用户建立的堆。关于建立这种用户建立的堆的资料，参见 ILE 概念。

对于默认堆可以使用以下的 ILE 可连接的 AP1：

释放存储空间(CEEFRST)可连接的 API 释放一个以前分配的堆存储空间。

取得堆存储空间(CEEGTST)可连接的 API 在一个堆中分配存储空间。

重分配存储空间(CEECZST)可连接的 API 修改以前分配的存储空间的大小。

注意：不能在用 DFTACTGRP(*YES)建立的程序中使用以上或任何其它的 ILE 可连接的 API。这是因为在这种程序中不允许使用静态连接（通过 CALLB 操作）。

关于存储空间管理可连接的 API 的其他资料参见系统 API 参考手册。

8.7.1.1 为一个运行时数组动态分配存储空间

下面的例子显示了怎样从一个 ILE RPG/400 过程中管理属于默认堆的动态存储空间。在这个例子中，过程 DYNARRAY 为一个实用非连接的压缩型数组动态地分配存储空间。

该过程的调用者可以从 DYNARRAY 中申请以下之一：

把一个元素添加到数组中

由数组中返回一个元素

释放数组的存储空间

DYNARRAY 使用三个 ILE 可以连接存储空间的 API：CEEGTST（获取存储空间），CEECZST（重分配存储空间）和 CEEFRST（释放存储空间）来完成这些操作。

图 35 显示了 DYNARRAY 的定义规范表。把过程定义为使用 a(15,0)压缩十进制数组。只要简单地把元素定义改为字符型字段，就可以很容易转换成处理字符型数组。

```
*================================================================*
          * DYNARRAY：分配一个无边界的压缩型（15,0）运行时数组，这个      *
          *           过程分配这个数组，返回一个数组值并重分配数组。      *

*================================================================*

*----------------------------------------------------------------*
          * 过程参数：                                             *
          * 1.输入：功能选择，可能的值是：                         *
          *         1 往数组中加元素                               *
          *         2 从数组中返回元素                             *
          *         3 释放数组中的存储                             *
          * 2.输入/输出：往数组中加或返回数组的元素
```

```
*
     * 3.输入：加或返回数组元素的下标                              *

*--------------------------------------------------------------*
     D Operation       S              1P 0
     D Element         S             15P 0
     D Index           S              5P 0

*--------------------------------------------------------------*
     * 这个过程支持操作的命名常量定义。                            *

*--------------------------------------------------------------*
     D AddToArr        C                    1
     D ReturnElem      C                    2
     D Terminate       C                    3

*--------------------------------------------------------------*
     * 定义存储管理 API's 的接口所用变量。                        *
     *  1)Heapld=堆的标识，从缺省的堆中分配，值设为 0。          *
     *  2)Size=分配或释放的字节数                                *
     *  3)RetAddr=堆的存储地址                                   *

*--------------------------------------------------------------*
     D HeapId          S              9B 0 INZ(0)
     D Size            S              9B 0
     D RetAddr         S               *

*--------------------------------------------------------------*
     * 定义动态数组，元素数定为允许的最大值。定义中不必说明所用存储    *
     * 空间。（由于它是 BASED）                                  *

*--------------------------------------------------------------*
     D DynArr          S                   DIM(32767) BASED(DynArr@)
     D                                     LIKE(Element)
     D DynArr@         S               *
```

```
*------------------------------------------------------------------*
          * 完整地保存动态数组中当前元素的号码。                    *
*------------------------------------------------------------------*
          D NumElems        S               9B 0 INZ(0)

*------------------------------------------------------------------*
          * 分配数组的初始元素号码以及其后加到数组中去的最少元素数。  *
*------------------------------------------------------------------*
          D InitAlloc       C                   100
          D SubsAlloc       C                   100
```

图 35 DYNARRAY 的定义规范表

图 36 显示了 DYNARRAY 计算规范表，在这里定义了调 APIs 的入口参数列表

```
*------------------------------------------------------------------*
          * 过程参数                                               *
*------------------------------------------------------------------*
          C     *ENTRY        PLIST
          C                   PARM                    Operation
          C                   PARM                    Element
          C                   PARM                    Index

*------------------------------------------------------------------*
          * CEEGTST API 的接口（取得堆存储位置）                    *
          *  1)Heapld=堆的标识，将从缺省的堆中分配。这样，设这个变量为 0。  *
          *  2)Size=分配的字节数                                    *
```

```
*  3)RetAddr=分配存储的返回地址                                    *
*  4)*OMIT=反馈参数。这里规定*OMIT 意思是如果没有与请求匹配的值，  *
*          要收到一个来自 API 的例外。由于没有监控这种情况，调用   *
*          例程将接收这个例外。                                    *

*-------------------------------------------------------------------*
     C     CEEGTST_PL    PLIST
     C                   PARM                    HeapId
     C                   PARM                    Size
     C                   PARM                    RetAddr
     C                   PARM                    *OMIT


*-------------------------------------------------------------------*
* CEECZST API 的接口（分配存储）                                     *
*  1)RetAddr=要重分配的存储地址                                     *
*  2)Size=新存储空间大小                                            *
*  3)*OMIT=反馈参数                                                 *

*-------------------------------------------------------------------*
     C     CEECZST_PL    PLIST
     C                   PARM                    RetAddr
     C                   PARM                    Size
     C                   PARM                    *OMIT


*-------------------------------------------------------------------*
* CEEFRST API 接口（释放存储）                                       *
*  1)RetAddr=要释放的存储地址                                       *
*  2) *OMIT=反馈参数。这里规定*OMIT 的意思是如果没有与请求匹配，    *
*          接收来自 API 的例外。                                    *

*-------------------------------------------------------------------*
     C     CEEFRST_PL    PLIST
     C                   PARM                    RetAddr
     C                   PARM                    *OMIT
```

图 36 DYNARRAY 的参数列表

过程的基本逻辑如下：

1、运行初始化子例程，它调用子例程 ALLOCATE。

该子例程通过调用 ILE 可连接的 API CEEGTST（取得堆存储空间）分配基于数组初值（在这里是 100）的堆存储空间。

2、确定执行哪种操作（添加或返回数组元素，或释放存储空间）。

在向数组添加元素之前，过程检查是否有足够的堆存储空间。如果不足的话，它调用子例程 REALLOC，使用 ILE 可连接的 API CEECZST（重分配存储空间）来申请更多空间。如果有返回要求，那么不管是申请到了，还是为零，过程都返回到调用者。如果需要的元素没有在数组中，那么就会出现后一种情况。如果有释放存储空间的请求，那么过程调用子例程 DEALLOC，该子例程调用 ILE 可连接的 API CEEFRST（释放堆栈空间），然后以 LR 为 ON 返回到调用者。

3、返回

图 37 显示了定义主逻辑的计算表，图 38 显示了包含子例程的计算规范表。

```
*----------------------------------------------------------------*
            * 选择要执行的操作                                    *
*----------------------------------------------------------------*
          C                   SELECT


*----------------------------------------------------------------*
            * 如果用户选择往数组中加，假如数组足够大，做第一个检查，如果不
*
            * 够大，要增加大小，把元素加进去。                    *
*----------------------------------------------------------------*

          C     Operation     WHENEQ    AddToArr
          C     Index         IFGT      NumElems
          C                   EXSR      REALLOC
          C                   ENDIF
          C                   EVAL      DynArr(Index) = Element


*----------------------------------------------------------------*
```

```
            * 如果用户选择从数组返回一个元素。假如数据足够大，做第一个检查，   *
            * 如果不够大，清除这个元素，如果数组足够大，返回这个元素。         *
 *-------------------------------------------------------------------*
          C     Operation     WHENEQ    ReturnElem
          C     Index         IFGT      NumElems
          C                   CLEAR                   Element
          C                   ELSE
          C                   EVAL      Element = DynArr(Index)
          C                   ENDIF
 *-------------------------------------------------------------------*
            * 如果用户选择结束，要重分配数组，设 LR 为 ON，中止这个例程。       *

*-------------------------------------------------------------------*
          C     Operation     WHENEQ    Terminate
          C                   EXSR      DEALLOC
          C                   MOVE      *ON           *INLR
          C                   ENDSL
*-------------------------------------------------------------------*
*   Done.                                                           *
*-------------------------------------------------------------------*
          C                   RETURN
```

图 37 DYNARRAY 的主逻辑部分

```
*-------------------------------------------------------------------*
*                S U B R O U T I N E S                              *
*-------------------------------------------------------------------*

*===================================================================*
* *INZSR：初始化子例程                                               *
*                                                                   *
*         功能：分配一个运行时数组的初始存储空间                    *
*===================================================================*
          C     *INZSR        BEGSR
          C                   Z-ADD     InitAlloc     NumElems
```

```
           C                   EXSR       ALLOCATE

           C                   ENDSR
*============================================================*
* ALLOCATE：分配动态数组的存储                                *
*                                                            *
*         功能：用 CEEGTST API 分配 DynArr 的存储空间         *
*============================================================*
           C     ALLOCATE      BEGSR
*                                                            *
* Determine the number of bytes needed for the array.        *
*                                                            *
           C                   EVAL       Size = NumElems * %SIZE(DynArr)
*                                                            *
* Allocate the storage and set the array basing pointer      *
* to the pointer returned from the API.                      *
*                                                            *
           C                   CALLB      'CEEGTST'      CEEGTST_PL
           C                   MOVE       RetAddr        DynArr@
*                                                            *
* Initialize the storage for the array.                      *
*                                                            *
           C     1             DO         NumElems       i                 5 0
           C                   CLEAR                     DynArr(i)
           C                   ENDDO
           C                   ENDSR


*============================================================*
* REALLOC：重分配存储                                        *
*                                                            *
*         功能：增加动态数组的大小，并初始新数组元素。        *
*============================================================*
           C     REALLOC       BEGSR
*
*  Remember the old number of elements
*
           C                   Z-ADD      NumElems       OldElems          5 0
*
```

```
*  Calculate the new number of elements.  If the Index is
*  greater than the current number of elements in the array
*  plus the new allocation, then allocate up to the index,
*  otherwise, add a new allocation amount onto the array.
*
          C                   IF        Index > NumElems + SubsAlloc
          C                   Z-ADD     Index         NumElems
          C                   ELSE
          C                   ADD       SubsAlloc     NumElems
          C                   ENDIF
*                                                               *
* Calculate the new size of the array                           *
*                                                               *
          C                   EVAL      Size =  NumElems * %size(DynArr)
*                                                               *
* Reallocate the storage and set the array basing pointer       *
* to the pointer returned from the API.                         *
*                                                               *
          C                   CALLB     'CEECZST'     CEECZST_PL
          C                   MOVE      RetAddr       DynArr@
* Initialize the new elements for the array.                    *
*                                                               *
          C     1             ADD       OldElems      i
          C     i             DO        NumElems      i
          C                   CLEAR                   DynArr(i)
          C                   ENDDO
          C                   ENDSR
*===============================================================*
* DEALLOC：解除分配的存储                                       *
*                                                               *
*         功能：释放数组的存储空间                              *
*===============================================================*
          C     DEALLOC       BEGSR
          C                   CALLB     'CEEFRST'     CEEFRST_PL
          C                   ENDSR
```

图 38 DYNARRAY 的子例程

要建立过程 DYNARRAY，键入：

CRTRPGMOD MODULE(MYLIB/DYNARRAY)SRCFILE(MYLIB/QRPGLESRC)，然后使用 CRTPGM 或 CRTSRVPGM 命令把该过程与其它模块相连接。

第九章　程序和过程调用

在 ILE 中，可以调用程序或过程。调用者必须指出调用语句的目标是程序还是过程。ILE RPG/400 提供以下操作码支持程序或过程调用以及参数传递。

操作码	功能
CALL	调用程序目标
CALLB	调用连接的过程
RETURN	返回到调用的程序或过程
PLIST	定义参数列表
PARM	定义参数

本章描述了如何：

调用程序或过程

在程序和过程间传递参数

从程序或过程中返回

使用 ILE 可连接的 API

调用图形子例程

调用特殊子例程

9.1　程序/过程调用概述

在 ILE 内的程序处理是基于过程级的（ILE 程序包括一个或多个模块，模块包括一个或多个过程。一个 ILE RPG/400 模块只包含一个过程。其它的 ILE 语言允许有多个过程）。“程序调用”是过程调用的特殊形式；它是对程序入口过程的调用。程序入口过程是在建立程序时设计，在程序被调用时接收控制权。

本单元比较了程序调用和过程调用。为了帮助你理解调用间的相互关系，也介绍了调用堆栈的概念。

9.1.1　程序调用

可以使用程序调用来调用 OPM 或 ILE 程序。程序调用是对程序目标(*PGM)的调用。调用程序把控制传递给被调用程序之前，被调用程序的名字转换成运行时的地址。因此程序调用通常称为动态调用。

调用 ILE 程序或 EPM 程序或 OPM 程序都是程序调用的例子。对于不可连接的 API 的调用也是程序调用的例子。

你可以使用 CALL 操作进行程序调用。

当 ILE 程序被调用时，程序入口过程接收程序的参数并且得到对该程序的初始控制。另

外，在该程序内的所有过程都成为有效的过程调用。

9.1.2 过程调用

与 OPM 程序不同，ILE 程序不只限于程序调用。ILE 程序也可以用静态过程调用或过程指针调用其它过程。过程调用也叫做调用。

静态调用是对一个 ILE 过程的调用，由该过程的名字在连接时转为地址，因此，以静态命名。使用静态过程调用的运行性能比使用程序调用要快。静态调用允许有操作描述，省略参数，并且扩展了传递的参数个数的限制（到 399）。

过程指针调用提供了对过程的动态调用方法。例如，你可以把过程指针作为参数传递给另一个过程，然后，运行在传递的 PARM 操作中指定的过程，也可以用一个过程名或地址数组向不同的过程发送过程调用。如果被调用的过程在相同的活动组中，那么过程指针调用的开销几乎与静态过程调用的开销完全相同。

使用任何一种调用，可以调用同一个 ILE 程序或服务程序中某个模块内的过程，或是某个 ILE 服务程序内的过程。任何能用静态过程调用的过程都可以用过程指针调用。

要使用 CALLB 操作进行过程调用。

9.1.3 调用堆栈

调用堆栈是按照后进先出顺序(IFO)排列的调用堆栈入口列表。调用堆栈入口是对程序或过程的调用(CALL 或 CALLB)。每个作业有一个调用堆栈。

当 ILE 程序被调用时，程序入口过程首先加到调用堆栈中。然后系统自动地执行过程调用，并且相关的用户过程也被添加。当过程被调用时，只有用户过程被添加；对于程序入口过程没有动作。

图 39 显示一个调用堆栈，应用程序有 OPM 程序组成的，调用一个有两个模块的 ILE 程序，一个是 RPG 模块，包含程序入口程序和相关的用户入口过程；一个是 C 模块包含普通的过程。注意在本书的图解中，当前入口是在堆栈底部的。

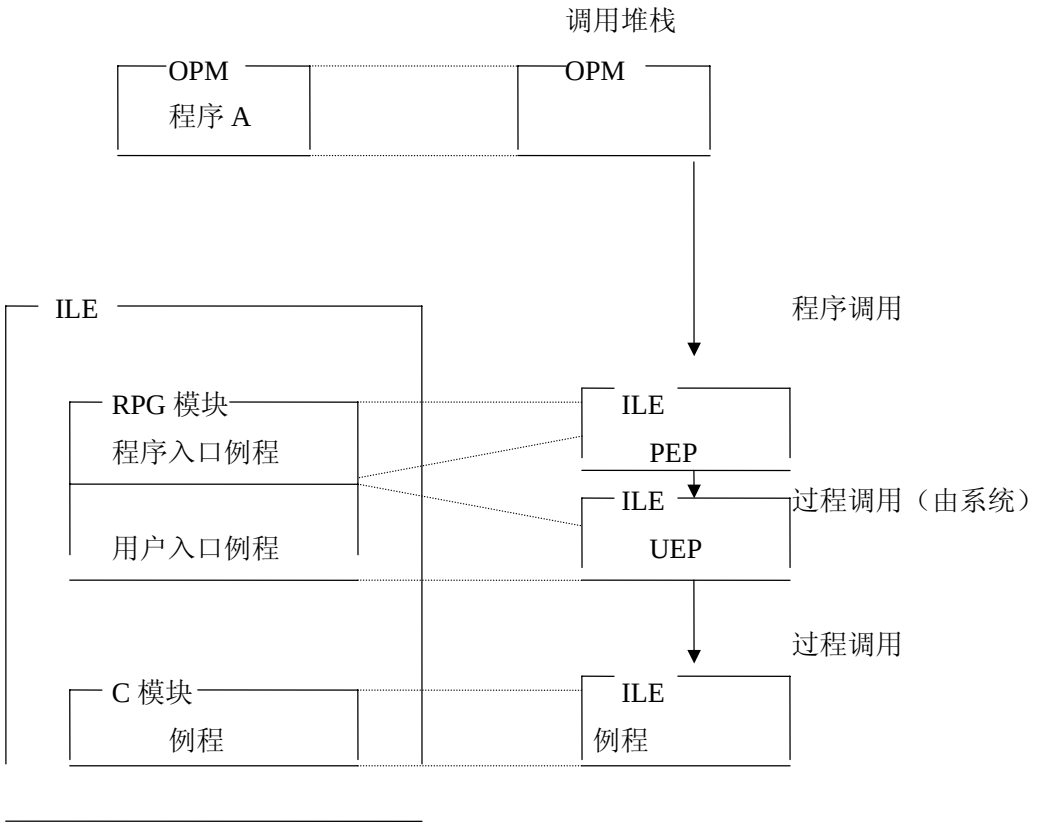


图 39 调用堆栈中的程序和过程调用

注意：用程序调用，对程序入口过程和用户入口过程(UEP)的调用是同时发生的，因为对用户入口过程(UEP)的调用是自动的。因此，今后在本章和其它章节中有关调用堆栈的图解中，把对程序调用的两个步骤合在一起。

关于调用堆栈需要理解的很重要的一点是，在堆栈上的 ILE RPG/400 过程。如果没有返回到调用者就不能被调用。因此，在调用其它过程时要小心，不能调用这样的过程，该过程已经调用一个处于活动状态的 ILE RPG/400 过程。

例如，假设在同一程序中有过程 A，B 和 C。如果过程 A 调用过程 B，那么过程 B 就不可以调用过程 A 或 B。如果 B 返回了(LR 为 ON 或不为 ON 均可)，并且过程 A 接着调用过程 C，那么 C 可以调用过程 B，但是不能调用过程 A 或 C。参见图 40。

过程 B 不能调用　　过程 C 不能调用
过程 A 或过程 B　　过程 A 或过程 C

图 40 过程不能调用其它活动的 RPG 过程

类似的，也不能调用已经在调用堆栈上的 OPM RPG 程序。

9.2 使用 CALL 或 CALLB 操作

你可以使用 CALL(调用程序)操作来调用程序，使用 CALLB(调用连接的过程)操作来调用过程。两个操作在语法及使用上都非常相似。要调用程序或过程，按以下一般步骤进行：

1、在因子 2 上指定要调用的目标。

2、可以选择指定一个错误指示器（73 和 74 列）和/或一个 LR 指示器（75 和 76 列）。

如果指定了错误指示器，那么当目标调用因错误结束时，错误指示器被置为 ON。类似的，如果指定了 LR 指示器，那么当被调用目标以 LR 为 ON 返回，则 LR 指示器被置为 ON。

3、向被调用目标传递参数，可以在调用操作的结果字段指定一个 PLIST(参数列表)或紧跟着调用操作使用 PARM 操作。

两种操作都把控制由调用者传递给被调用目标。在被调用目标运行之后，控制返回到调用程序或过程中调用操作后的第一条语句。

以下是关于两个调用操作的一些细节：

因子 2 可以是变量、文字、或命名的常量。注意该项分大小写。

只适合 CALL 操作：因子 2 可以是库名/程序名的形式，例如：MYLIB/PGM1。如果没有指定库名，那么使用库列表来寻找该程序。被调用程序的名字可以在运行时通过在因子 2 中指定一个字符型变量给出。

只适合 CALLB 操作：要进行过程指针调用，你应该指定过程指针的名字，它包含调用过程的地址。

一个过程可以包含用相同或不同的 PLIST 规定对同一个目标的多次调用，该目标可以定义相同或不同的 PLIST。

当 ILE RPG/400 过程（包括程序入口过程）被第一次调用时，字段被初始化，并且被赋予控制权。在其它对同一个过程的一般调用中，如果前一次调用没有结束，那么被调用过程中的所有字段，指示器和文件都与上一次调用返回时相同。

系统记录了从一个 RPG 过程中调用的所有程序的名字，在 RPG 过程连接到程序(*PGM)中时，即使你不能指出哪个过程做了调用，你也可以使用 DSPPGMREF 来查询这些名字。

对于模块，可以使用 DSPMOD DETAIL(*IMPORT)来查询被调用的过程名。列表中的一些过程会是系统过程，它们的名字通常包含下划线，对于这些你不必考虑。

如果使用变量调用一个目标，那会看到一个名为*VARIABLE 的入口（设有库名）。

仅适合 CALLB：编译程序建立一个操作描述来指示 CALLB 操作上传递的参数个数，并把这个值放在被调用过程的程序状态数据结构的*PARMS 字段中。这个数字包括任何设计为省略(PARM 操作上的*OMIT)的参数。

如果在 CALLB 操作中使用了操作扩展符(D)，那么编译程序也为每个字段及子字段建立一个操作描述。

关于操作描述的详细介绍，参见 2.6.3.3 当中的“使用操作描述”。

使用 CALL 或 CALLB 操作码时还有其他一些限制。关于这些内容的详细描述，参见 ILE RPG/400 参考手册。

9.2.1 调用操作的例子

关于使用 CALL 操作的例子，参见：

3.1.13 节中的“调试例子的样本源程序”是调用 RPG 程序的例子。

2.6.3.5.1 节中的“在 PSDS 中使用*PARMS 字段”是调用 PRG 程序的例子。

关于使用 CALLB 操作的例子，参见：

2.4.4 节中的“样本服务程序”是调用服务程序中过程的例子。

2.5.7.1.1 节中的“为运行时数组动态分配存储空间”是调用可连接 API 的例子。

3.1.13 节中的“调试例子的样本源程序”是调用 C 过程的例子。

4.6.2.2 节中的“CUSMAIN：RPG 源程序”是主查询程序调用多个 RPG 过程的例子。

9.3 参数传递

ILE RPG/400 在 CALL 操作中提供了 PLIST 和 PARM 操作码来指定要传递和接收的字段。

在与 ILE RPG/400 不同的 HLL 传递参数时，要注意以下方面：

HLL 的参数传递方式。

每一 HLL 有其自己的参数传递方法，可以是数据参数值的指针（就象在 ILE RPG/400 中），值的拷贝或值本身。关于这些方式的详细介绍，参见 2.6.3.6 节中的“ILE 交互语言调用”。

交互语言数据兼容性

不同的 HLL 支持不同的数据表示方法。通常应该只传递那些调用和被调的程序或过程的公共数据类型的数据。但是，有时可能不能确定传递给你的数据的确切格式。在这种情况下，应告知过程的用户，传递一个操作描述来提供有关识别所传递的参数格式信息。有关描述参见 2.6.3.3 节中的“使用操作描述”。

参数个数

通常，应传递与被调用的程序或过程个数相同的参数，如果传送的比所需要的少，被调用目标引用了一个没有被传递数据的参数，那么就会发生错误。关于调用过程时省略参数的介绍，参见 2.6.3.4 节中的“省略参数”。

ILE RPG/400 编译程序总是传递一个操作描述来描述传递给过程的参数个数。可以使用这个描述来检验调用与被调用的过程是否使用了相同个数的参数，有关详细的描述，参见 2.6.3.3 节中的“使用操作描述”。

注意：其它 ILE 语言，可能不传递上面那样的操作描述。

9.3.1 使用 PLIST 操作

PLIST 操作：

定义一个名字，使用该名字可以引用参数列表。参数的列表紧跟在 PLIST 操作后用 PARM 操作指定。

定义参数的因子 1 必须是 PLIST 名。这个名字可以在一个或多个调用操作的结果字段中使用。如果参数列表是被调用过程的入口参数，那么因子 1 必须是*ENTRY。

在一个过程中可以有多个 PLIST 操作。但是，只能指定一个*ENTRY PLIST。

关于 PLIST 操作的例子，参见 2.5.1.1 节中的图 30 和 2.6.3.5.1 节中的图 42。

9.3.2 使用 PARM 操作

PARM 操作用来定义由过程传出或接收的单个参数。每个参数在一个独立的 PARM 操作中定义。可以在结果字段中指定参数名，该名字不必与调用/被调用的过程中的名字相同因子 1 和因子 2 是可选的，根据结果字段处于调用还是被调用的程序/过程中来指出变量或文字的值是由这些字段接收还是由这些字段传出。

表 7　PARM 操作中因子 1 和因子 2 的意义

状态	因子 1	因子 2
在调用程序中	返回时从结果字段传来的值	调用时放在结果字段的值
在被调用的过程中	调用时，从结果字段传来的值。	返回时放在结果字段的值。

注意:只有当被调用的过程正常返回调用者时，才会发生向因子 1 或结果字段的传送。如果向任何一种入口传送数据时发生错误，那么传送就不会完成。

如果在调用过程时参数不足，那么当被调用的过程使用没有指定的参数时就会发生错误。要避免这样的错误，可以用 PARM 操作的结果字段指定*OMIT 来定义没有传递的参数。这样，被调用的程序。通过检查参数是否有*NULL 值来检验该参数是否被省略了。详细的介绍，参见 2.6.3.4 节中的“省略参数”。

在指定 PARM 操作时要注意以下方面：

一个或多个 PARM 操作必须紧跟在 PLIST 操作之后作使用。

一个或多个 PARM 操作可以紧跟 CALL 或 CALLB 之后使用。

当在 PARM 的结果字段中指定了多次出现的数据结构时，数据结构的所有出现将用一个字段传送。

PARM 操作的因子 1 和结果字段不能是文字，look－ahead 先行字段，命名常量，或用户日期保留字。

如果结果字段为多次出现的数据结构或*OMIT，那么因子 1 和因子 2 必须为空。

使用 PARM 操作码时还有其它的限制。关于这些内容的详细介绍参见 ILE RPG/400 参考手册。

关于 PARM 操作的例子，参见：

2.5.1.1 节中的图 30

2.3.1.2 节中的图 23

2.6.3.5.1 节中的图 43

2.4.4 节中的图 26

9.3.3 使用操作描述

有时即使作不能确切地知道被调用过程的数据类型，也必须向它传递参数，（例如：不同类型的串）。在这种情况下，可以使用操作描述来为被调用过程提供参数形式的描述信息。这些信息能使程序正确地解释串，只在被调用程序需要时才使用操作描述，通常是 ILE 可连接的 API。

注意：最初 ILE RPG/400 编译程序只支持对字段及子字段的操作描述。换言之，对于数据结构，数组或表，操作描述是无效的。

要使用操作描述，在 CALLB 操作上指定操作码扩展符(D)。然后，由调用过程建立操作扩展符，并作为隐藏参数传递给程序。即使 CALLB 操作引用了省略参数，也可以指定(D)。对于省略参数不建立操作描述。

操作描述对于被传递的参数及其传递方式是没有影响的。当把没有必要的操作描述传递给过程时，这些操作描述忽略。

使用 ILE 可连接 API 获得操作描述信息命令(CEEDOD)和取得关于串参数的描述信息命令(CEESGI)你可以从操作描述中获取信息。

注意操作描述只允许对过程级的调用使用。如果在 CALL 操作上指定了操作码扩展符‘D’，那么编译程序将产生错误消息。

关于使用操作描述的例子，参见 2.4.4 节中的“样本服务程序”。该例子包括一个把字符串转换成为等价的十六进制形式的服务程序。该服务程序使用操作描述来确定字符串的长度及要转换的长度。

9.3.4 省略参数

在调用过程时，有时传送的参数个数比被调用过程所要求的参数要少几个。例如，当调用 ILE 可连接 API 时可能就会出现这种情况。

要表示省略的参数，应在 PARM 操作中指定*OMIT。当指定了*OMIT 时，编译程序就会产生必要的编码以向被调用的过程指示该参数已被省略了。

注意：被调用过程的语言必须支持引用传递。否则结果是不可预测的。

省略的参数不能为被调用的过程引用。如果引用的话，就会发生错误。指定*OMIT 值，就是让你在编写被调用过程时，该过程能够检查参数是否被省略，如果是的话，就不去引用它。

要检查在 ILE RPG/400 过程中参数是否被省略了，应使用内部函数%ADDR 来检查参数的地址。如果地址为*NULL，那么参数已被省略了。也可以使用可连接 API CEEST（检查省略参数）。

当指定*OMIT 时要注意以下方面：

只允许在紧跟在 CALLB 操作后的 PARM 操作中或被 CALLB 所使用的 PLIST 中指定*OMIT。 如果指定了*OMIT，那么 PARM 操作的因子 1 和因子 2 必须为空。

在作为*ENTRY PLIST 一部分的 PARM 操作中允许指定*OMIT。

下面是如何使用*OMIT 的一个简单例子。在此例子中，一个过程调用 ILE 可连接 API CEEDOD 来分解操作描述。CEEDOD API 需要接收七个参数；但是在调用过程中只定义了六个参数。CEEDOD（及大多数的可连接 API）的最后一个参数是确定 API 如何结束的反馈码。但是，调用过程已设计成由异常而不是由反馈码接收错误消息。因而，在对 CEEDOD 的调用中，该过程必须指出对应反馈码的参数已被省略了。

图 41 显示了这种情况的编码。

```
*=========================================================================*
            * 下面的例程调用 ILE API 来分解一个操作描述符。希望通过例外来接收   *
            * 错误信息。是由省略反馈码做到这点的(CEEDOD API 的最后一个参数)。   *

*=========================================================================*

*-------------------------------------------------------------------------*
            * CEEDOD parameters                                            *

*-------------------------------------------------------------------------*

DName++++++++++ETDsFrom+++To/L+++IDc.Functions++++++++++++++++++++
D posn            S               9B 0
D desctype        S               9B 0
D datatype        S               9B 0
D descinf1        S               9B 0
D descinf2        S               9B 0
D datalen         S               9B 0

*-------------------------------------------------------------------------*
            * 调用 CEEDOD 来分解操作说明符，省略反馈码。                      *
```

```
*----------------------------------------------------------------------*
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C                   EVAL       posn=1

C                   CALLB      CEEDOD
C                   PARM                    posn
C                   PARM                    desctype
C                   PARM                    datatype
C                   PARM                    descinf1
C                   PARM                    descinf2
C                   PARM                    datalen
C                   PARM                    *OMIT

C                   MOVE       '1'          *INLR
```

图 41　使用*OMIT 键字指定省略参数

在本书的前部分关于建立服务程序的例子中提供了一个类似的使用*OMIT 的例子。关于它的源码参见 2.4.4 节中的“样本服务程序”。

9.3.5　检查所传递的参数的个数

可以使用程序状态数据结构(PSDS)中的*PARM 字段来确定传递给被调用过程的参数个数。这个数包括省略的参数个数（在 PARM 操作中的*OMIT）。根据所写的过程。这个数使你能避免引用没有传递的参数。例如，假设想编写一个有时传递三个参数，有时传递四个参数的过程。可以让该过程根据*PARMS 字段的值处理两种个数的参数。

当需要新字段时，参数的个数要增多。使用该字段的新过程会为它传递一个值。旧的过程保持不变。

因为*PARMS 字段中的参数个数包括使用*OMIT 传递的参数，所以建议你在使用 CALLB 操作时检查传递参数是否有*NULL 值。关于使用*OMIT 的详细介绍，参见 2.6.3.4 节中的“省略参数”。

9.3.5.1　使用 PSDS 中的*PARMS 字段

在本例中，程序 FMTADDR 已被修改了若干次以允许修改公司雇员的地址，FMTADDR 由几个不同的程序调用。这些程序的不同之处只在于它们用来处理雇员信息的参数的个数。即对于 FMTADDR 的新需求，那么，就要添加新的参数，但仍支持旧的调用 FMTADDR 的程序，并且不用修改或重新编译。

对雇员地址的修改可以概括为以下方面：

最初只需要街道名及号码，因为所有的雇员都住在同一城市中。这样，城市名和省名可采用默认值。后来，公司扩展了，因此由于公司范围的应用需要，有多种城市信息。

更进一步，公司的扩展也需要多种省名信息。程序根据传递的参数个数来处理信息。参数个数会在 3 到 5 之间变化。该数字告诉程序是否提供城市名和/或省名的缺省值。图 42 显示了此程序的源语句。

程序 FMTADDR 的主逻辑如下：

1、通过检测 Numof Parms 的值来判断传递了多少个参数。

   这个字段是依据该程序 PSDS 中的*PARMS 定义的。如果该数大于 4，那么默认者被由第五个参数 P_Province 提供的实际省名代替。

   如果该数大于 3，那么默认的城市名被由第四个参数 P_City 提供的实际城市名代替。

2、使用子例程 GetStreet#来修改供打印的街道号。

3、连接生成完整的地址。

4、返回。

```
*==================================================================*
          * FMTADDR—地址格式                                        *
          *                                                               *
          * 参数项                                                     *
          * 1.地址      字符（70）                                    *
          * 2.街码      压缩（5,0）                                      *
          * 3.街名      字符（20）                                    *
          * 4.城市      字符（15）                                    *
          * 5.省        字符（15）                                    *

*==================================================================*

          D Address          S              70
          D Street#          S               5  0
          D Street           S              20
          D P_City           S              15
          D P_Province       S              15



*------------------------------------------------------------------*
          * 可以不传递的参数的缺省值                                     *
```

```
*-------------------------------------------------------------------*
          D City            S               15    INZ('Toronto')
          D Province        S               15    INZ('Ontario')


*-------------------------------------------------------------------*
           * 用*PARMS 来确定传递多少参数                              *

*-------------------------------------------------------------------*
          D Psds            SDS
          D  NumOfParms       *PARMS

          C     *ENTRY        PLIST
          C                   PARM                    Address
          C                   PARM                    Street#
          C                   PARM                    Street
          C                   PARM                    P_City
          C                   PARM                    P_Province


*-------------------------------------------------------------------*
           * 检查省参数是否传递。如果传递了，用参数值来代替缺省值。       *

*-------------------------------------------------------------------*
          C                   IF        NumOfParms > 4
          C                   EVAL      Province = P_Province
          C                   ENDIF

*-------------------------------------------------------------------*
           * 检查城市参数是否传递。如果传递了，用参数值来代替缺省值。       *

*-------------------------------------------------------------------*
          C                   IF        NumOfParms > 3
          C                   EVAL      City = P_City
          C                   ENDIF
*-------------------------------------------------------------------*
           * 把‘CStreet#’转换成字符格式‘Street#’                      *
*-------------------------------------------------------------------*
```

```
C                   EXSR      GetStreet#
*------------------------------------------------------------*
 * 地址格式    街号，城市，省                                  *
*------------------------------------------------------------*
C     CStreet#      CAT(P)    Street:1      Address
C                   CAT(P)    ',':0         Address
C                   CAT(P)    City:1        Address
C                   CAT(P)    ',':0         Address
C                   CAT(P)    Province:1    Address
C                   SETON                                   LR
*============================================================*
 * 子例程：GetStreet#                                       *
 *  取街号的字符格式，左对齐，右边用空格填充。                 *

*============================================================*
C     GetStreet#    BEGSR
C                   MOVEL     Street#       CStreet#       10
*------------------------------------------------------------*
* 找第一个非零字符                                           *
*------------------------------------------------------------*
C     '0'           CHECK     CStreet#      Non0            5 0
*------------------------------------------------------------*
* 如果没有非零字符，截取非零字符的起始值。                   *
*------------------------------------------------------------*
C                   IF        Non0 > 0
C                   SUBST(P)  CStreet#:Non0 CStreet#
*------------------------------------------------------------*
* 如果没有非零字符，用 0 做为街号的起始值。                  *
*------------------------------------------------------------*
C                   ELSE
C                   MOVEL(P)  '0'           CStreet#
C                   ENDIF
C                   ENDSR
```

图 42 程序 FMTADDR 的源程序。检测*PARMS 以确定传递的参数个数。

图 43 显示了程序 FMTADDR 的源程序。该程序说明了 FMTADDR 的使用。为方便起见，调用 FMTADDR 的三个程序连接为一个程序。因为，是举例子，所以数据是程序描述的。因为 PRTADDR 是“3 合 1”程序，因此，它必须定义三个不同地址的数据结构。类似的，在此计

算规范表中有三个部分。每个部分对应于程序的一个阶段。打印地址之后，程序 PATADDR 返回。

```
*=================================================================*
* PRTADDR—打印地址                                                 *
*                                                                 *
* 参数项                                                          *
* 1.地址     字符（70）                                           *
* 2.街号     压缩（5,0）                                          *
* 3.街名     字符（20）                                           *
* 4.城市     字符（15）                                           *
* 5.省       字符（15）                                           *

*=================================================================*
          FQSYSPRT   O    F   80        PRINTER
*-----------------------------------------------------------------*
* 第一步：原地址数据结构，仅街和号是变量信息。                    *
*-----------------------------------------------------------------*
          D Stage1          DS
          D   Street#1                     5P 0 DIM(2) CTDATA
          D   StreetNam1                  20    DIM(2) ALT(Street#1)


*-----------------------------------------------------------------*
* 第二步：改变地址数据结构为城市信息当前变量的形式。              *
*-----------------------------------------------------------------*
          D Stage2          DS
          D   Street#2                     5P 0 DIM(2) CTDATA
          D   Addr2                       35    DIM(2) ALT(Street#2)
          D     StreetNam2                20    OVERLAY(Addr2:1)
          D     City2                     15    OVERLAY(Addr2:21)

*-----------------------------------------------------------------*
* 第三步：改变地址数据结构为省信息当前变量的形式。                *
*-----------------------------------------------------------------*
          D Stage3          DS
          D   Street#3                     5P 0 DIM(2) CTDATA
```

```
D   Addr3                    50    DIM(2) ALT(Street#3)
D     StreetNam3             20    OVERLAY(Addr3:1)
D     City3                  15    OVERLAY(Addr3:21)
D     Province3              15    OVERLAY(Addr3:36)
*----------------------------------------------------------------*
* 程序 1：在城市参数加上之前使用 FMTADDR。                       *
*----------------------------------------------------------------*
C                   DO      2             X                5 0
C                   CALL    'FMTADDR'
C                   PARM                  Address          70
C                   PARM                  Street#1(X)
C                   PARM                  StreetNam1(X)
C                   EXCEPT
C                   ENDDO
*----------------------------------------------------------------*
* 程序 2：在省参数加上之前使用 FMTADDR。                         *
*----------------------------------------------------------------*
C                   DO      2             X                5 0
C                   CALL    'FMTADDR'
C                   PARM                  Address          70
C                   PARM                  Street#2(X)
C                   PARM                  StreetNam2(X)
C                   PARM                  City2(X)
C                   EXCEPT
C                   ENDDO

*----------------------------------------------------------------*
* 程序 3：在省参数加上之后使用 FMTADDR。                         *
*----------------------------------------------------------------*
C                   DO      2             X                5 0
C                   CALL    'FMTADDR'
C                   PARM                  Address          70
C                   PARM                  Street#3(X)
C                   PARM                  StreetNam3(X)
C                   PARM                  City3(X)
C                   PARM                  Province3(X)

C                   EXCEPT
```

```
            C                   ENDDO
            C                   SETON                                         LR
*---------------------------------------------------------------------*
* 打印地址                                                            *
*---------------------------------------------------------------------*
            OQSYSPRT    E
            O                       Address
      **
      00123Bumble Bee Drive
      01243Hummingbird Lane
      **
      00003Cowslip Street       Toronto
      01150Eglinton Avenue      North York
      **
      00012Jasper Avenue        Edmonton        Alberta
      00027Avenue Road          Sudbury         Ontario
```

图 43 程序 PRTADDR 的源程序，调用程序 FMTADDR。

要建立这些程序，按以下步骤进行：

1、用图 42 中的源程序建立 FMTADDR，键入：

CRTBNDRPG PGM(MYLIB/FMTADDR)

2、用图 43 中的源程序建立 PRTADDR，键入：

CRTBNDRPG PGM(MYLIB/PRTADDR)

3、调用 PRTADDR.输出如下所示:

```
123Bumble BeePrive, Toronto, Ontatrio
                •
                •
                •
                •
                •
                •
```

9.3.6 ILE 交互语言调用

ILE RPG/400 使用同样的调用方式来调用 ILE HLL 程序或过程：CALL 和 CALLB。类似的，ILE RPG/400 使用一种传递方式—引用来传递和接收参数。换句话说，ILE RPG/400 通过实际数据目标的指针来传递和接收参数。其它 ILE 语言可能有不同的传递数据的方法。表 8 显示了 ILE 语言的一般传递参数的方法。

表 8　　ILE 语言的默认参数传递方式

ILE HLL	传送数据目标	接收数据目标
ILE RPG/400	通过引用	通过引用
ILE C/400	直接通过值，或通过引用	直接通过值或通过引用
ILE COBOL/400	通过引用或通过 CONTENT	通过引用或间接地通过值
ILE CL	通过引用	通过引用

简而言之，在涉及与其他 ILE 语言特别是 C 或 COBOL 过程传递参数时，必须确保它是通过引用接收数据的。

9.4　　从被调用的程序/过程中返回

从过程（包括程序入口过程）中返回涉及到把它的调用堆栈入口由调用堆栈中除掉，关闭或删除操作。

ILE RPG/400 过程用以下的方式之一把控制返回给调用过程：

正常结束

非正常结束

末结束

下面的内容介绍由被调用过程返回的方式。

关于在 RPG 程序周期中在何处检测 LR，H1 到 H9 及 RT 指示器的详细描述，参见 ILE RPG/400 参考手册中关于 RPG 程序周期的单元。

9.4.1　　正常结束

当 LR 指示器为 ON 而 H1 到 H9 指示器不为 ON 时，过程正常结束并且控制返回到调用过程。

LR 指示器可按以下方式置为 ON：

隐含地，在程序周期中从主文件或次文件处理完最后一个记录。

明确地，把 LR 置为 ON 时。

如果是下面情况，过程也可以正常结束。

执行了 RETURN 操作，H1 到 H9 不为 ON，LR 指示器为 ON。

RT 指示器为 ON，H1 到 H9 指示器不为 ON，并且 LR 指示器为 ON。

当过程正常结束时，发生以下事件：

执行由 PARM 的因子 2 向结果字段传送。

所有在定义规范表中指定了“到文名”的数组和表，及所有加锁的数据区数据结构被写出。

所有被该过程加锁的数据区被解锁。

所有打开的文件被关闭。

设置一个返回码以向调用过程指出该过程已正常结束，并且控制返回到调用程序。

下一次调用时，该过程会有一个新的供处理的有效拷贝。

注意：如果你习惯于用 LR 为 ON 结束，这时，存储空间被释放。

如果正在一个命名的(稳定的)活动组中运行，那么你可能要考虑以不结束返回。原因是：

如果活动组不结束，存储空间就不会被释放。所以对于以 LR 为 ON 结束来说，没有存储空间的回收。如果每次调用不必重新初始化程序那么调用性能会有提高。

9.4.2 非正常结束

当出现以下情况时，过程非正常结束，并且控制返回到调用程序：

当产生 ILE RPG/400 错误信息时采用了取消选项。

执行了*PSSR 中的 ENDSR*CANCL 或 INFSR 错误子例程。（关于*PSSR 的*CANCL 返回指针及 INFSR 错误子例程的详细资料，参见 3.2.3.3 节中的“在 ENDSR 操作中定义返回指针”。）

当执行 RETURN 操作时，H1 到 H9 某个指示器为 ON。

在 RPG 周期中当最后记录(LR)处理时，H1 到 H9 指示器之一为 ON。

受过程外的某些事件结束的影响，过程也会非正常结束。例如：如果一个 ILE RPG/400 过程 X 调用其它的过程，它直接向调用过程 X 发出逃逸信息[（如 CL 过程）]就会出现这种情况。

当过程非常结束时，会出现以下情况：

所有被打开的文件被关闭。

任何被该程序上锁的数据区解锁。

如果有功能检查，那么会渗透给调用者。如果没有，那么直接向 X 的调用者发出逃逸信息 RNX9001（如果没有其它替代的逃逸信息的话）。

在对该过程的下一次调用中，会有一个供处理的新拷贝。关于过程状态数据结构的详细资料，参见 3.2.3 节中的“使用 RPG 规范处理程序”。

9.4.3 以未结束返回

当 LR 或 H1 到 H9 指示器中没有设为 ON，并且发生下列某件之一，过程就可以把控制返回给调用过程而不结束：

执行了 RETURN 操作

RT 为 ON 并且控制到达了 RPG 周期的*GETIN 部分，在这种情况下控制立即返回到调用过程。（关于 RT 指示器的详细介绍，参见 ILE RPG/400 参考手册。

如果调用一个过程并且它以未结束返回，那么当你再次调用该过程时，所有的字段，指示器和文件将保持离开该过程时的值。

注意：

假如程序在*NEW 活动组中运行时不是这样的，因为该活动组在程序返回时就被删除了。

在这种情况下，下一次调用程序就象以 LR 为 ON 结束一样。如果正在共享文件，那么文件的状态可能与离开该过程时不同。

可以使用 RETURN 操作码或将 RT 指示器与 LR 指示器及 H1 到 H9 指示器联用。要注意到 RPG 程序周期中关于 REUTRN 操作，RT 指示器，LR 指示器和 H1 到 H9 指示器的检测序列。

9.4.4 使用 ILE 可连接的 API 返回

可以用 ILE 可连接 PAI CEETREC 来正常结束一个程序，该 API 会终止在同一个活动组中某个控制边界的全部调用堆栈入口。当过程用 CEETREC 结束时它遵循正常的 LR 处理，就象在 2.6.4.1 节中的“正常结束”中描述的那样。在对该过程进行下一次调用时，会有一个供处理的新的有效拷贝。

同样，可以使用 ILE 可连接 API CEE4ABN 来非正常地结束一个过程。该过程会象在 2.6.4.2 节中描述的那样结束。

注意：不能在以 DFTACTGRP(*YES)建立的过程中使用这两个 API，因为在这些过程中不允许有过程调用。

关于 CEETREC 和 CEE4ABN 的详细情况，参见系统 API 参考手册。

9.5 使用可连接的 API

可连接的应用程序编程接口(API)对所有的 ILE 语言都是有效的。在某些情况下它们提供的附加功能超过了由某种特定 ILE 语言所提供的功能。对于多种语言应用程序它们也是很有用的，因为它们是独立于 HLL 的。

可连接 API 提供的广泛功能包括：

- 活动组和控制流程管理
- 存储空间管理
- 条件管理
- 信息服务
- 源语句调试
- 调用数学函数管理
- 操作描述访问

可以使用 ILE RPG/400 所使用的相同调用方式来调用过程，即 CALLB 操作。图 44 显示了对一个可连接 API 调用的样本。

```
C                   CALLB   'CEExxxx'
C                   PARM                    parm1
C                   PARM                    parm2
                    ...
C                   PARM                    parmn
```

C PARM feedback

图 44 ILE 可连接 API 的调用语样本

在这里：

CEEXXXX 是可连接 API 的名字

Parm1，Parm2，…Parmn 是需要或可省略的 API 参数。

feed back 是用来指示可连接 API 结果的可省略反馈码。

以下是一些限制：

要求返回值的 API 不能被调用。

要求值参数的 API 不能被使用。

如果在 CRTBNDRPG 中指定了 DFTACTGRP(*YES)那么不能使用 API。

关于可连接 API 的详细资料，参见系统 API 参考手册。

9.5.1 使用可连接 API 的例子

关于使用可连接 API 的例子，参见 ：

2.4.4 节中的“样本服务程序”是使用 CEE DOD 的例子。

2.5.7.1.1 节中的“为运行时数组动态人配存储空间”。

是使用 CEEGTST，CEEFRST 和 CEECZST 的例子。

3.2.5 节中的“使用取消处理程序”是使用 CEEHDLR 和 CEEHDLL 的例子。

9.6 调用绘图例程

ILE RPG/400 可以使用 CALL 操作调用 OS/400 绘图功能。它包括绘图数据显示管理(GDDM，有一系列画图的工具）和介绍绘图例程（一组商用图例程）。因子 2 必须是文字或‘GDDM’，用 PLIST 和 PARM 操作来传递下列参数：

要运行的绘图例程名

绘图例程所需要的参数必须是图形例程所需要的数据类型。

处理这个 call 的过程不隐含开始或结束 OS/400 图形例程。

关于 OS/400 图形。图形例程及参数的更多资料，参见《GDDM 程序设计》和《GDDM 参考》。

注：你能用“CALL”操作来调用 OS/400 图形，不能用 CALLB 操作，不能把 Date，Time，Timestamp 或 Graphic 字段传给 GDDM，也不能把指针传递给它。

9.7 调用特殊例程

ILE RPG/400 支持使用操作“CALL”和“PARM”来调用下面几个特殊例程：

信息检索例程(SUBR23R3)

移动括号内的双字节数据和删除控制字符(SUBR40R3)。

移动括号内的双字节数据和增加控制字符(SUBR41R3)。

注意：不能用操作码“CALLB”来调用这些特殊例程。

虽然支持信息检索例程，仍建议使用“QMHRTVM　messape API”，这样会更有效。

类似的，例程“SUBR40R3”和“SUBR41R3”执行时也存在类似的情况，他们不能被更新，来反映由 RPGⅣ通过新的图形数据类型支持的各级图形。

第三部分

调试和异常处理

这部分包括以下内容：

- 使用 ILE 源码调试器调试一个 ILE 应用程序
- 写处理异常的程序
- 获得转储数据

第十章　调试程序

调试时允许检测、诊断和消除程序中的“运行时”错误。可以使用 ILE 源码调试器来调试 ILE 程序。

本章叙述怎样使用 ILE 源码调试器来做：

为 ILE RPG/400 程序调试做准备

开始调试

在调试中增加和移出程序

在调试中查看程序源码设置、清除条件断点和无条件断点

单步执行程序

显示字段的值

修改字段的值

显示字段的属性

给出字段、表达式或调试命令的速记名。当调试和检测程序时，一定要确保此程序用的测试数据所在的测试库在库列表的最前面，以便调试时不影响任何的实际数据。

可以使用下面的命令来保护产品库中的数据库文件不被无意中修改：

使用开始调试(STRDBG)命令，规定 UPDPROD 参数的值为“*NO”。

用改变调试(CHGDBG)命令

有关防止产品库文件被无意修改的更多内容请参看《控制语言参考》中有关调试的附录。

有关 ILE 源码调试器的更多内容（包括调试一个程序或服务程序所需要的授权和优先级的作用），请参看《ILE 概念》中关于调试的章节。

ILE 源码调试

ILE 源码调试用来检测和清除程序目标和服务程序中的错误，在任何 ILE 程序中使用调试命令，能做以下事情：

查看程序源码或修改调试视图；

设置。清除条件断点和非条件断点；

单步执行指定数目的语句；

显示或修改字段、数据结构和数组的值给出字段、表达式或调试命令的速记名。

在使用源码调试之前，用“CRTRPGMOD”或“CRTBNDRPG”生成一个模块目标或程序目标时，你必须选择一个调试视图。

启动调试程序之后，可以设置断点，然后调用这个程序、

当程序由于一个断点或一个单步命令而停止运行时，在屏幕上显示程序停止点上相关模块目标的视图。此刻你可以执行其他动作，例如显示或修改字段的值。

注意：如果程序已被优化，虽然仍能显示字段，但它们的值可能是不可靠的，为保证数据项或数据结构的值是正确（最新）的，请在相应的定义规范表上指定“NOOPT”键字，要更

改优化级，参见 2.3.4.2 中的“更改优化级”。

10.1 调试命令

在 ILE 源码调试程序中可用许多调试命令。这些调试命令和它们的参数，在显示模块源码及计算表达式屏幕底部的调试命令行输入。输入时可以大写、小写或大小写混合。
注意：调试命令行上输入的调试命令不是控制语言命令。

ILE 源码调试的联机帮助描述了调试命令。解释了允许的缩写，并为每一个命令提供了语法图。它也对每种 ILE 语言提供了用源码调试程序的显示或修改变量的例子。在调试会话时，你可以按 F1 键取得帮助。

这些调试命令列表如下：

命令	描述
ATTR	显示一个变量的属性。这些属性是记录在调试符号表中的变量的大小和类型。
BREAK	在被测试程序的某个位置上输入一个条件断点或非条件断点。使用“BREAK 行号 WHEN 表达式 ”来输入一个条件断点。
CLEAR	清除条件断点和非条件断点。
DISPLAY	显示用 EQUATE 命令分配的名字和字义，也允许显示与当前显示在显示模块原码屏幕上的源程序不同的模块，这个部分模块目标必须存在于当前程序目标中。
EQUATE	允许分配一个表达式、变量或调试命令的速记名。
EVAL	显示或修改变量的值，或者显示表达式、记录、数据结构或数组的值。
QUAL	定义出现在“EVAL”命令后面变量的范围。这个命令只用于带局部变量的语言。例如 ILE C/400，ILE RPG/400 不用它。
STEP	运行被调试程序的一个或多个语句。
FIND	在当前显示的模块向前或向后检索一个指定的行号或字符或文本。
UP	把源码显示窗口往前移到以输入数为开始行的那一屏。
DOWN	把源码显示窗口往前移到以输入数为末尾行的那一屏显示。
LEFT	使源码显示窗口左移。
RIGHT	向右移动源码显示窗口，移动的列数是你输入的字符数。
TOP	窗口定位在从第一行显示。
BOTTOM	窗口定位在最后一行。
NEXT	窗口定位到（以当前显示为基准的）下一个断点那一屏显示。
PREVIOUS	定位窗口到（以当前显示为基准的）前一个断点那一屏显示。
HELP	对可用的源码调试命令显示联机帮助信息。

10.2 准备一个要调试的程序

要调试一个程序或模块，必须有调试数据。由于调试数据是在编译期间生成的，那么在用“CRTBNDRPG”或“CRTRPGMOD”命令生成时，就要指定它是否要包含调试数据，你可以在任一命令中使用“DBGVIEW”参数，来指定在编译时生成什么类型的数据。

与某个模块有关的调试数据的类型叫做一个调试视图，对每一个要调试的模块，能生成下列视图：

源语句视图

拷贝源码视图

列表视图

语句视图

命令“CRTRPGMOD”和“CRTRPGMOD”的缺省值是生成一个语句视图，这个视图提供了最接近于前一个版本的调试支持。

如果在模块中不想包括调试数据或者想缩短编译时间，那么在生成模块时，指定“DBGVIEW”(*NONE)。但是，没有调试数据不能用 DUMP（转储功能）。

注意：一个程序或模块的存储需求，在某种程度上依赖于包含在其中的调试数据的类型。下表中列出的是“DBGVIEW”参数的值是依它们对辅存需求影响程度的升序排列的。

1、*NONE

2、*STMT

3、*SOURCE

4、*COPY

5、*LIST

6、*ALL

一旦生成了一个带调试数据的模块并且已把它连到一个程序目标中了，就可以开始调试程序了。

10.2.1 生成一个源语句视图

一个源语句视图包括源语句成员的文本描述，这个视图不包含任何“/COPY”成员。另外，它不适用于源成员是 DDM 文件的情况。在用“CRTRPGMOD”或“CRTBNDRPG”命令生成一个模块时参数“DBGVIEW”是使用*SOURCE、*COPY 或*ALL 选项。能生成一个源语句视图来调试模块。

当模块目标(*MODULE)被编译时，编译程序生成源语句视图。它是引用源语句成员的文本中的某些部分来生成，而不是把成员文本复制到模块中。基于这个原因，你不能在生成的模块和这些成员的调试模块之间来修改、重命名或移动源语句成员，如果这样做，这些源成员的视图可能不能使用：

例如，用“CRTBNDRPG”生成程序“DEBUGEX”的一个源语句视图时，键入：

```
CRTBNDRPG    PGM(MYLIB/DEBUGEX) SRCFILE(MXLIB/GRPGLESRC)
                    TEXT('ILE RPG/400 Program DEBUGEX')
```

DBGVIEW(*SOURCE)

使用“CRTRPGMOD”生成模块“DBGEX”的一个源语句视图，键入：

```
CRTRPG    MODVLE(MYLIB/DBGEX)    SRCFILE(MYLIB/QRPGLESRC)
          TEXT('Entry module for program    DEBUGEX')
          DBGVIEW(*SOURCE)
```

这两个生成命令都指定 DBGVIEW(*SOURCE)生成了调试模块 DBGEX 的源语句视图，由于缺省值规定将产生一个带/COPY 成员和扩展 DDS，以及其它信息的编译列表。

10.2.2 生成拷贝源码视图

拷贝源码视图包含了源成员的文本，以及所有扩展到源码文本中的/COPY 成员的文本。当使用 COPY 视图时，可以用源语句视图的程序来调试源成员，拷贝源码视图的程序来调试/COPY 成员。

由 DBGVIEW(*COPY)生成的源成员视图与由“DBGVIEW(*SOURCE)”生成的相同。如果源文件是一个 DDM 文件，可用源语句视图，不能用 COPY 源码视图。

通过在 DBGVIEW 参数中指定*COPY 或*ALL，可生成 COPY 源码视图来调试一个模块。

当模块目标(*MODULE)被编译时，编译程序生成“COPY”视图，COPY 视图引用源成员中（包括源成员和/COPY 成员）的一部分来生成，而不是把成员复制到视图中，基于这个原因，在模块目标被生成时及调试由这些成员生成的模块时，不能修改、重命名或移动源成员。如果这样做，这些源成员的视图，也许是不可用的。

例如，为生成包含/COPY 成员的程序 TEST1 一个源码视图，键入：

```
CRTBNDRPG    PGM(MYLIB/TEST1)    SRCFILE(MYLIB/QRPGLESRC)
             TEXT('ILE RPG/400 program TEST1')
             DBGVIEW(*COPY)
```

在生成命令中指定 DBGVIEW(*COPY)就生成一个带有/COPY 成员的一个源语句视图用来调试模块 TEST1，缺省规定产生一个编译列表，由于 OPTION(*SHOWCPY)是缺省值，这个编译列表将包含/COPY 成员。

10.2.3 生成清单视图

清单视图包含的文本类似于由 ILE RPG/400 编译程序产生的编译清单中的文本。这个信息是否包含在清单视图中依赖于在生成命令中指定 OPTION(*SHOWCPY)或 OPTION(*EXPDDS)。(*SHOWCPY)使列表中包括/COPY 成员。OPTION(*EXPDDS)使编译列表包括外部描述文件。

注意：如果在编译列表中指定标识（通过指定 INDENT 参数），在清单视图中就不会出现。

在命令“CRTRPGMOD”或“CRTBNDRPG”中生成模块时，规定参数“DBGVIEW”*LIST 或*ALL 选项，就生成一个清单视图来调试模块。

当模块目标(*MODULE)被生成时，编译程序就生成列表视图。这个列表视图把相应的

源成员的文本复制到模块目标来生成。一旦列表视图生成，就与建立它的源成员没关系了。

例如，要为程序 TEST1 生成一个包含扩展 DDS 的列表视图，键入：

```
CRTBNDRPG    PGM(MYLIB/TEST1)    SRCFILE(MYLIB/QRPGLESRC)
                SRCMBR(TEST1)    OUTPUT(PRINT)
                TEXT('ILE RPG/400 Program TEST1')
                OPTION(*EXPDDS)    DBGVIEW(LIST)
```

在生成命令中指定 DBGVIEW(*LIST)和 OPTION(*EXPDDS)就生成了一个带有扩展 DDS 的列表视图来调试 TEST1 的源码。注意：OUTPUT(*PRINT)和 OPTION(*EXPDDS)是缺省值。

10.2.4 生成一个语句视图

语句视图允许使用调试命令调试模块目标。由于不显示源程序，所以必须用编译列表中的源语句最左边列显示的语句号，换句话说，要有效的使用这个视图，你需要一个编译清单。

当生成一个模块时，在 CRTRPGMOD 或 CRTBNDRPG 中 DBGVIEW 参数指定*STMT，就生成了一个语句视图来调试模块。

有下列情况时，使用这个视图。

你需要调试程序而内存有限制时，不想重新编译模块或程序。

你正把编译过的目标发送给其他用户，并且希望使用调试器能诊断编码问题，而又不想让用户看到实际的编码。

例如，使用 CRTBNDRPG 为程序 DEBUGEX 生成一个语句视图，键入：

```
CRTBNDRPG    PGM(MYLIB/DEBUGEX)    SRCFILE(MYLIB/QRPGLESRC)
                TEXT('ILE RPG/400 Program DEBUGEX')
```

使用“CRTRPGMOD”生成一个语句视图，键入：

```
CRTRPGMOD MODULE(MYLIB/DBGEX)    SRCFILE(MYLIB/QRPGLESRC)
                TEXT('Entry module for program DEBUGEX')
```

缺省值将产生一个编译清单和一个语句视图，根据编译清单得到语句号，你使用调试命令来调试这个程序。

如果修改生成命令的缺省值，必须明确的指定 DBGVIEW(*STMT)和 OUTPUT(*PRINT)。

10.3 ILE 启动源语句调试器

一旦生成了调试视图（语句、源语句、COPY 或清单），就能开始调试应用程序了。ILE 启动源码调试器，使用 STRDBG 命令。一旦启动调试器，它将一直活动直到输入结束调试(ENDDBG)命令为止。在 STRDBG 命令中指定 Program(PGM)参数来指定程序在调试环境中。它们可以是任何 OPM 或 ILE 程序。能把一个程序加到调试环境中，必须对这个程序有*CHANGE 权力。

注意：如果使用 COPY 或源语句视图来调试一个程序。源码必须和被调试程序目标在

同一系统中。另外，源码还必须用与编译时相同的名字放在一个库/文件成员中。

如果一个 ILE RPG/400 程序与调试数据放在一个调试环境中，那么显示入口模块，它有一个调试视图，否则，显示连接到带调试数据的 ILE RPG/400 程序的第一个模块。

例如，要启动调试环境来调试程序 DEBUGEX 和一个可调用程序 RPGPGM，键入：

STRDBG　PGM(MYLIB/DEBUGEX　MYLIB/RPGPGM)

出现“Display Module Source”屏。如图 45。DEBUGEX 包含两个模块，一个 RPG 模块 DBGEX 和一个 C 模块 CPNOC，它们的源码参见“3.1.13 中内容”。

如果入口模块有一个源 COPY 或清单视图，则屏幕将显示第一个程序的入口模块的源码，在这种情况下，程序用 DBGVIEW(*ALL)生成，显示的是主模块 DBGEX 的源码。

```
*                          Display Module Source
*
*  Program:   DEBUGEX        Library:   MYLIB          Module:   DBGEX    *
*       1
*=================================================================  *
*       2         * DEBUGEX_这个程序解释 ILE RPG/400 源码调试。给出不同数据类型  *
*       3         *             和数据结构的样本。也可做为产生样本的格式化转储。 *
*       4         *                                                 *
*       5         *
*       6         *                                                 *
*       7         *
*=================================================================  *
*       8                                                         *
*       9
*-----------------------------------------------------------------  *
*      10         * DEBUG 键字可用来做格式化转储。                    *
*      11         *
*-----------------------------------------------------------------  *
*      12        H DEBUG                                          *
*      13                                                         *
*      14
*-----------------------------------------------------------------  *
*      15         * 对不同的 ILE RPG/400 数据类型定义独立字段。       *
*                                                        More...  *
*  Debug . . .                                                    *
*                                                                 *
*  F3=End program   F6=Add/Clear breakpoint   F10=Step   F11=Display variable  *
```

```
*  F12=Resume        F13=Work with module breakpoints      F24=More keys        *
*                                                                               *
*                                                                               *
```

图 45　程序 DEBUG 的“Display Module Source”屏

注意：不能在“STRDBG”命令中指定 ILE 服务程序。

可以在“Work With Module List”显示屏(F14)中用选项 1(增加)来把 ILE 服务程序加到一个调试环境中。

也可让原码调试把它做为 STEP INTO 调试命令的一部分而加进来。

10.3.1 设置调试选项

在开始一个调试环境后，可以设置调试选项了。特别，可以指出当调试程序时，是否更新数据库文件。这个选项是由 STRDBG 命令的 UPDPROD 参数规定的。“UPDPROG”参数规定当作业处于调试方式时，产品库中数据库文件是否可以打开用来更新记录，增加新记录，否则，在试图运行一个使用这些文件的程序之前，必须把这些文件拷贝到测试库中。要设置调试选项，请按如下步骤做：

1、输入“STRDBG”后，如果当前显示不是“Display Module Source”屏，键入：

DSPMODSRC

出现“Display Module Source”屏。

2、按 F16 键（设置调试选项）出现“Set Debug Options”显示。

3、在“Update Production files”上填(Yes)，按执行键返回到“Display Module Source”屏。当作业在调试模式时，产品库中的数据库文件被更新。

10.4　往调试环境中增加/移出程序

当启动一个调试环境后，可以往调试环境中增加程序，从调试环境中移出程序。要做到这些，你必须对程序有*CHANGE 的权力。

对 ILE 程序，你使用 DSPMODSRC 命令后在显示“Work With Module List”中要增加程序用选项 1(Add program)，要移出一个 ILE 程序或服务程序，在同一显示中用选项 4(Remore program)。当一个 ILE 程序或服务程序被清除时，那个程序的所有断点也将被清除，对从一个调试环境中同时加入或移出的 ILE 程序或服务程序的参数没有限制。

对 OPM 程序，使用“增加程序”(ADDPGM)命令或移出除程序(RMVPGM)命令，在一个调试环境中，一次可以有十个程序在这个调试环境中。

10.4.1 增加服务程序例子

在本例中，把服务程序 CVTTOHEX 加到前面已经启动的调试环境中。（对“服务程序”的讨论见 2.4.4 节中的“样本服务程序”部分。

1、如果当前显示不是“Display Module Source”屏，键入：

DSPMODSRC

“Display Module Source”屏出现。

2、按 F14 键(Work With Module List)显示“Work With Module List”屏，如图 46。

3、要加入服务程序 CVTTOHEX，在屏幕的第一行，键入：1(Add program)，在 Program/module 字段项填 CVTTOHEX。在 Library 字段填 MYLIB，修改缺省程序类型*PGM 为*SRVPGM，按执行键。

4、按 F12(CALCEL)返回到“Display Module Source”屏。

```
*
*              Work with Module List
*                          System:   AS400S1
*                    Type          options,          press          enter.
*
*       1=Add  program      4=Remove  program      5=Display  module  source
*    8=Work with module breakpoints
*
*     Opt            Program/module            Library                  Type
*
*      1              cvttohex                  mylib                  *SRVPGM
*
*                     RPGPGM                     MYLIB                  *PGM
*
*                      RPGPGM                                          *MODULE
*
*                     DEBUGEX                    MYLIB                  *PGM
*
*                  DBGEX                                 *MODULE       Selected
*
*                     CPROC                                           *MODULE
*
*                                                                   Bottom  *
*  Command                                                                  *
*  ===>                                                                     *
*  F3=Exit    F4=Prompt    F5=Refresh    F9=Retrieve    F12=Cancel            *
*                                                                           *
```

图 46　往调试环境中增加一个 ILE 服务程序

10.4.2 从调试环境中移出 ILE 程序的例子

在本例中，从一个调试环境中移出 ILE 程序 CVTHEXPGM 和服务程序 CVTTOHEX。

1、如果当前显示不是“Display Module Source”屏，键入：

DSPMODSRC

“Display Module Source”屏出现。

2、按 F14(Work With Module List)显示“Work With Module List”屏，如图 47 显示。

3、在这个显示中键入 4(Remove program)执行键。

4、按 F12(取消)返回到“Display Module Source”屏。(图 47)从一个调试环境中移出一个 ILE 程序。

```
*                                                                          *
*                           Work with Module List                          *
*                                                  System:   AS400S1       *
*  Type options, press enter.                                              *
*    1=Add program    4=Remove program    5=Display module source          *
*    8=Work with module breakpoints                                        *
*                                                                          *
*  Opt      Program/module      Library         Type                       *
*                               *LIBL           *PGM                       *
*   4         CVTHEXPGM         MYLIB           *PGM                       *
*               CVTHEXPG                        *MODULE                    *
*   4         CVTTOHEX          MYLIB           *SRVPGM                    *
*               CVTTOHEX                        *MODULE                    *
*             RPGPGM            MYLIB           *PGM                       *
*               RPGPGM                          *MODULE                    *
*             DEBUGEX           MYLIB           *PGM                       *
*               DBGEX                           *MODULE      Selected      *
*               CPROC                           *MODULE                    *
*                                                                          *
*                                                                  Bottom  *
*  Command                                                                 *
*  ===>                                                                    *
*  F3=Exit    F4=Prompt    F5=Refresh    F9=Retrieve    F12=Cancel         *
*                                                                          *
```

图 47　用 DEBUG 环境传送一个 ILE 程序

10.5　查看程序源码

“Display Module Source”屏显示程序目标的源码，每次显示一个模块目标。如果模块目标编译时使用了如下的调试视图选项，那么可以显示这个模块目标的源码。

DBGVIEW(*SOURCE)

DBGVIEW(*COPY)

DBGVIEW(*LIST)

DBGVIEW(*ALL)

有两种方法可以改变“Display Module Source”屏的显示。

切换到不同的模块

修改模块的视图

当修改一个视图时，ILE 源码调试器把要修改的视图映射到对等位置。当修改模块时，显示视图中的可运行的语句被存储到内存中，当两次显示这个模块时可被查看，有断点的行号显示成高亮度，当一个断点，一个作业号或一个信息引起程序中止时，显示这屏设置断点的语句被高亮度显示出来。

10.5.1 查看一个不同的模块

要改变在“Display Module Source”屏显示中的模块目标，在这屏上用选项 5(Display Module Source)，也可以从 Display Module Source 显示中按 F14 键(Work With Module List)来访问这些模块的列表。

如果对一个程序目标用这个选项，则显示源 COPY 或列表视图的入口模块（如果存在的话），否则显示连接到带调试数据的程序的第一个模块。

查看不同模块目标的另一个方法是用“DISPLAY”调试命令，在调试命令行，键入：

DISPLAY MODULE　模块名

则显示模块目标名。这个模块目标必须存在于加入到这个调试环境中的某个程序中。

例如，用“Display Module Source”选项从 3.1.3 的图 45 中的模块 DBGEX，切换到模块 CPYOC，按如下步骤。

1、为了处理模块，键入 DSPMODSRC，按执行键“Display Module Source”屏显示出来。

2、按 F14(处理模块列表)显示“Work With Module List”屏，图 48 是显示例子。

3、要选择“CPROC”，在它的 OPT 列上键入“5”(Display Module Source)，按“执行键”。则出现图 49 中的显示，出现源码视图。如果没有源码视图，则显　示连接到带调试数据的程序的第一个模块。

```
*                                                                    *
*                          Work with Module List                     *
*                                                  System:   AS400S1 *
*  Type options, press enter.                                        *
```

```
*    1=Add program   4=Remove program   5=Display module source              *
*    8=Work with module breakpoints                                          *
*                                                                            *
*  Opt      Program/module     Library        Type                           *
*                              *LIBL          *PGM                           *
*                                                                            *
*           RPGPGM             MYLIB          *PGM                           *
*             RPGPGM                          *MODULE                        *
*           DEBUGEX            MYLIB          *PGM                           *
*             DBGEX                           *MODULE     Selected           *
*   5         CPROC                           *MODULE                        *
*                                                                            *
*                                                                   Bottom   *
*  Command                                                                   *
*  ===>                                                                      *
*  F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F12=Cancel               *
*                                                                            *
```

图 48　切换到一个不同的模块

```
*                                                                            *
*                          Display Module Source                             *
*                                                                            *
*  Program:   DEBUGEX       Library:   MYLIB          Module:   CPROC        *
*       1       #include <stdlib.h>                                          *
*       2       #include <string.h>                                          *
*       3       #include <stdio.h>                                          *
*       4       extern char EXPORTFLD[6];                                    *
*       5       void c_proc(int *size, char **ptr)                           *
*       6 {                                                                  *
*       7          *ptr = malloc(*size);                                     *
*       8          memset(*ptr, 'P',*size );                                 *
*       9          printf("import string: %6s.\n",EXPORTFLD);                *
*      10       }                                                            *
*                                                                            *
*                                                                   Bottom   *
*  Debug . . .                                                               *
```

```
*                                                                              *
*  F3=End program   F6=Add/Clear breakpoint   F10=Step   F11=Display variable*
*  F12=Resume       F13=Work with module breakpoints     F24=More keys        *
*                                                                              *
```

图 49　ILE C 过程“CPROC”的源码视图

10.5.2 修改一个模块的视图

生成模块时指定的值。根据这可以显示几种不同的视图。这些视图是：

源语句视图

COPY 源码视图

清单视图

可以通过“Select View”屏，改变在“Display Module Source”屏显示的模块的视图。选择视图的显示，可以从“Display Module Source”屏，按 F15(Select View)来做“Select View”屏如图 50 显示。窗口顶端显示当前视图的名字，下端显示其它可用的视图。程序中每一个模块有不同的可用视图，这依赖于用于生成它时所用的调试选项。

例如，源码视图改变到清单视图，按如下步骤：

1、键入 DSPMODSRC，按“执行键”，“Display Module Source”屏出现。

2、按 F15(Select View)，“Select View”窗口显示如图 50。

```
*                                                                              *
*                              Display Module Source                           *
*  ............................................................................*
*  :                           Select View                                  :  *
*  :                                                                        :  *
*  :  Current View . . . :     ILE RPG/400 Copy View                        :  *
*  :                                                                        :  *
*  :  Type option, press Enter.                                             :  *
*  :    Select                                                              :  *
*  :                                                                        :  *
*  :  Opt   View                                                            :  *
*  :   1        ILE RPG/400 Listing View                                    :  *
*  :            ILE RPG/400 Source View                                     :  *
*  :            ILE RPG/400 Copy View                                       :  *
*  :                                                                        :  *
*  :                                                               Bottom   :  *
*  : 12=Cancel                                                              :  *
*  :                                                                        :  *
```

```
*  :.....................................................................: *
*                                                                  More...  *
*  Debug . . .                                                              *
*                                                                           *
*  F3=End program   F6=Add/Clear breakpoint   F10=Step   F11=Display variable*
*  F12=Resume       F13=Work with module breakpoints     F24=More keys       *
*                                                                           *
```

图 50　改变模块的视图

注意：如果生成模块时指定了 DBGVIEW(*ALL)，则“Select View”窗口将显示三种可用视图：源语句、COPY、清单。如果模块没有“/COPY”成员，那么 COPY 视图等同于源语句视图。

3、在列表视图 OPT 行上键入“1”，按执行，“Display Module Source”屏出现，显示清单视图的模块。

10.6　设置和取消断点

你可以使用断点，在程序运行时，强制它在某点上停止，非条件断点在指定语句上停止程序目标的运行，条件断点是指当在某个语句满足时，指定条件停止程序的运行。

要在程序运行前设置断点，当程序停止时，显示“Display Module Source”屏显示，相应的模块目标以及出现断点的行的源码。这行是高亮度显示，这时，可以给字段赋值，设置更多的断点，运行任何调试命令。使用断点前，要了解有关断点的特性。

在一条语句上设置断点时，断点停在这条语句前。

当检索到一个条件断点的语句时，与断点有关的条件表达式在这条语句被处理前被赋值。如果表达式值为真，断点起作用，程序停止在这行上。

如果设置断点的这行不是一个可执行语句，断点被设置到下一个可执行语句上。

如果一个断点被分支，则不处理断点。

断点功能是通过调试命令规定的，这些功能包括：

—往程序目标中增加断点

—从程序目标取消断点

—显示断点信息

—当检索到一个断点继续程序的运行

如果设置断点后修改了模块视图，那么断点行号通过源码调试程序，映射到新视图中。

如果你用语句视图来调试模块或程序，那么可用从编译清单获得的语句号来设置或取消断点，有关的详细内容，见 3.1.6.4 的内容。

10.6.1 设置和取消无条件断点

你可以用下面的方法设置和取消无条件断点。

从“Display Module Source”屏中用 F6(增加/清除断点)。

在 Display Module Source”屏用 F13(处理模块断点)。

调试命令“BREAK”设置一个断点，

“CLEAR”取消。

“Work With Module BreakPoints”显示。

设置和清除一个无条件断点的最简单方式是用 F6(dd/Clear Breakpoint)。这个功能键做为一个触发器，它从光标所在行取消一个断点，（如果在那行已经设置了一个断点的话）。

要清除一个无条件断点用 F13 键(‘Work With Module BreakPoints’)。把光标移到要清除断点所在的行上，按 F13 键。出现允许设置或取消断点的选择列表。如果选择 4(Clear)，就从这行上取消了断点。

设置和取消在非条件断点的另一个方法是使用“BREAK”和“CLEAR”调试命令。用“BREAK”设置一个无条件断点，在调试命令行上键入：

BREAK　行号

变量“行号”是要设置断点的模块当前显示视图中的行号。

要使用“CLEAR”调试命令来取消一个无条件断点，在调试命令行上键入：

CLEAR　行号

变量“行号”是要取消断点的模块当前显示视图中的行号。

10.6.1.1　设置一个无条件断点的例子

在本例中，使用 F6(Add/Clear Breakpoint)设置一个无条件断点。这个断点被设置到第一个可运行的计算规范上，这样可以显示变量字段和数据结构。

1、要处理一个模块，键入“DSPMODSRC”按执行键，显示“Display Module Source”屏。

2、如果在模块显示的设置断点，接第 3 步做，如果想在不同模块中设置断点在调试命令行，键入：DSPLAY MODULE 模块名

模块名是要显示的模块名字。

3、要在第一个计算规范上设置一个无条件断点，把光标置到 76 行上。

4、按 F6 键(Add/Clear Breakpoint)。如果在 76 行无断点，那么一个条件断点就被设置到那行上。如图 51 所示。如果在这行有一个断点，就被取消。

注意：因为我们想把断点设在第一个计算规范上。因此，可以把光标设在计算规范开始前的任一行上。因为 76 行是第 1 个可执行语句，所以可以在这行设。

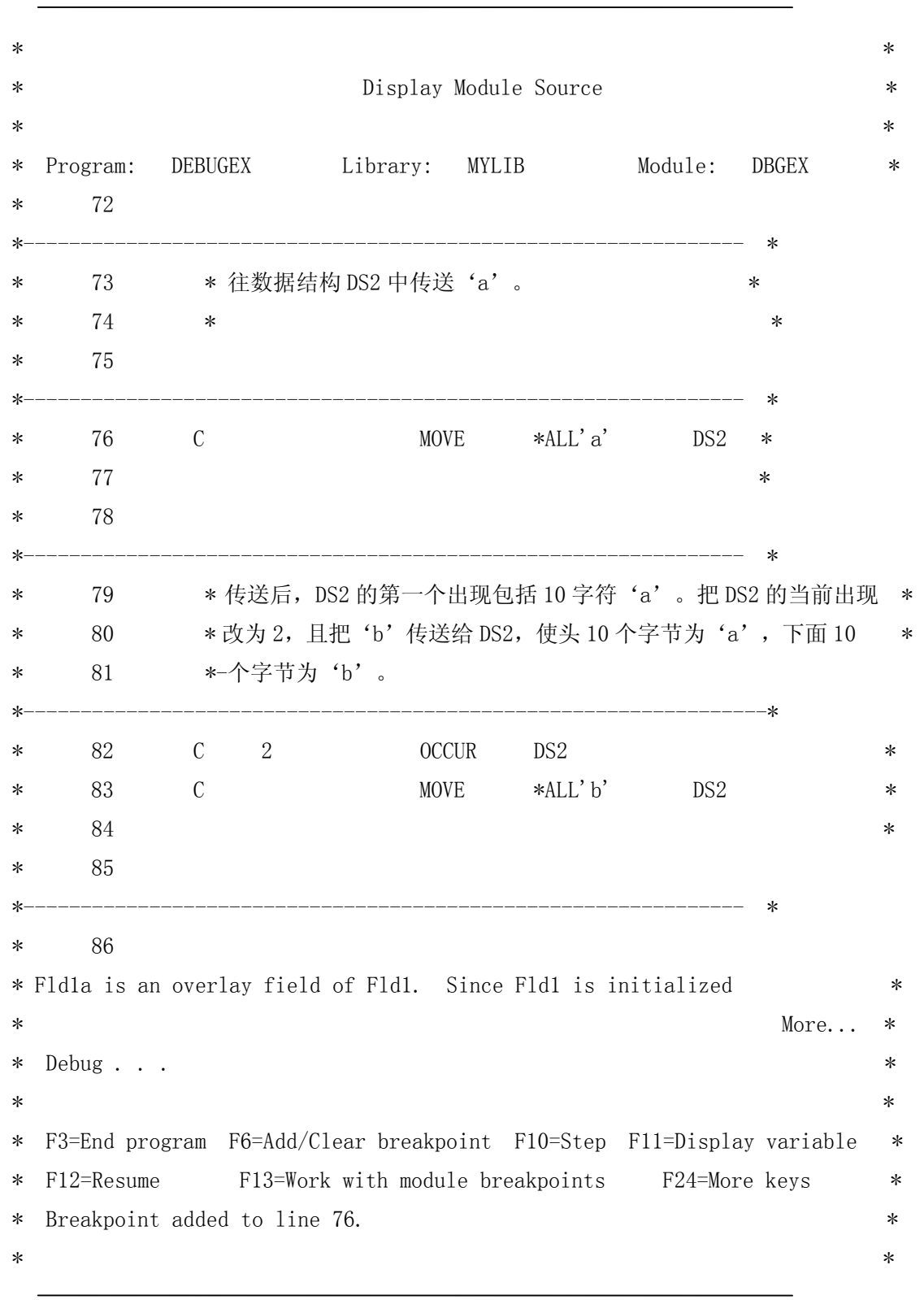


图 51　　设置一个无条件断点

5、断点设置后，按 F3(EXIT)离开“Display Module Source”屏，断点不能被清除。

6、调用这个程序，当到达断点时，程序停止，显示“Display Module Source”屏，有断点的那行高亮设显示。这时，你可以单步执行程序或接着执行。

10.6.2 设置和取消条件断点

可以通过如下方式设置或取消一个条件断点，这要：

“Work With Module BreakPoints”屏

用“BREAK”调试命令设置一个断点。

用“CLEAR”调试命令清除一个断点。

注意：条件断点使用的关系操作符是：

＜，＞，＝，≤，≥，和“＜＞”（不等）

设置或取消条件断点的一种方法是通过“Work With Module BreakPoints”屏。在“Display Module Source”屏幕按 F13(Work With Module BreakPoints，就可访问到这屏。)它提供了一个可以增加或取消条件及无条件断点的选项列表。此屏的一个例子如 3.1.6.2.1 中的图 52 所示。

要产生断点条件，在条件字段指定一个条件表达式，如果你设置断点的那行不是一个可执行语句，断点将被重新设置到下一个可执行语句。

一旦完成规定所有的断点，调用这个程序。在“Display Module Source”屏幕，可使用 F21(Command Line)来从命令行上调用这处程序，或者退出这屏后调用这个程序。

当检索到一个带条件断点的语句时，这个断点的条件表达式是在这个语句被执行前赋值的。如果结果为假，程序继续运行，如果结果为真，程序停止运行，并显示“Display Module Source”屏，此时，可以给字段赋值。设置更多的断点和运行调试命令。

设置和取消条件断点的另一个方法是使用“BREAK”和“CLEAR”调试命令。

要使用“BREAK”调试命令设置一个条件断点，在调试命令行，键入：

BREAK　行号　WHEN　表达式

行号是模块的当前显示视图中你想设置断点的行号，表达式是当遇到断点时被赋值的条件表达式。支持条件断点的相关操作码在本节的开始已给出。

在非数值条件断点表达式中，做比较前，较短的表达式在比较前先空格填充，这种填充是在民族语言排序顺序(NLSS)转换之前做的有关“NLSS”的更多介绍见 3.1.6.3 内容。

要用“CLEAR”调试命令取消一个条件断点，在调试命令行，键入：

CLEAR　行号

变量“行号”是模块的当前显示视图中想取消断点的行号。

10.6.2.1　使用 F13 设置一个条件断点的例子

在本例中，使用 F13(Work With Module BreakPoints)设置一个条件断点。

1、要设置一个条件断点按 F13(Work With Module BreakPoints)，出现“Work With Module BreakPoints”显示。

2、在这屏列表的第一行键入 1(Add)，来加一个条件断点。

3、要在第 111 行上设置一个“*IN02＝’1’”的条件断点，在 Line 字段里键入“111”，条件字段上键入“*IN02＝’1’”，按执行键，图 52 显示了加这个条件断点之后的

“Work With Module BreakPoints”显示。

```
*                                                                      *
*                    Work with Module Breakpoints                       *
*                                                   System:   TORASD80  *
*  Program  . . . :   DEBUGEX              Library  . . . :   MYLIB      *
*    Module . . . :     DBGEX              Type . . . . . :   *PGM       *
*                                                                      *
*  Type options, press Enter.                                           *
*    1=Add   4=Clear                                                    *
*                                                                      *
*  Opt      Line        Condition                                       *
*           111         *in02='1'                                       *
*           76                                                          *
*           90                                                          *
*                                                                      *
*                                                             Bottom   *
*  Command                                                              *
*  ===>                                                                 *
*  F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F12=Cancel          *
*  Breakpoint added to line 111.                                        *
*                                                                      *
```

图 52　　设置一个条件断点

在行 111 上设置了一个断点，这条语句运行前，给表达式赋值。如果结果为真（本例中，如果*IN02＝’1’)，程序停止，显示“Display Module Source”屏。如果结果为假，程序继续运行。原有的断点被在新设的同一位置断点所取代。

4、设置好断点后，按 F12(Cancel)退出“Work With Module BreakPoints”屏，按 F3(End Program)来退出 ILE 源码调试环境，断点不被取消。

5、调用这个程序，当检索到断点，程序中止，且显示“Display Module Source”屏。

这时，你可以单步执行程序或接着执行。

10.6.2.2　使用 BREAK 命令设置一个条件断点的例子

在本例中，当数据字段“Bigdate”有某一值时中止程序。用“BREAK”命令指定这个条件断点：

1、从“Display Module Source”屏，键入：

break　112　When　Bigdate＝’1994－09－30’

在行 112 设置了一个条件断点。

2、当条件断点设定后，按 F3(End program)结束 ILE 源码调试，断点不被清除。

3、调用这个程序，当检索到断点后，程序停止，显示“Display Module Source”屏。

```
*                                                                    *
*                          Display Module Source                     *
*                                                                    *
*  Program:   DEBUGEX       Library:   MYLIB         Module:   DBGEX  *
*     105                                                            *
*     106                                                            *
*--------------------------------------------------------------------*
*     107        * 在 SETON 操作后，*IN02= ‘1’                        *
*     108                                                            *
*--------------------------------------------------------------------*
*     109       C                   SETON                            *
*     110       C                   IF        *IN02                  *
*     111       C                   MOVE      ’1994-09-30’  BigDate  *
*     112       C                   ENDIF                            *
*     113                                                            *
*     114                                                            *
*--------------------------------------------------------------------*
*     115        *  现在开始格式化转储，然后由 LR 为 ON 返回。        *
*     116                                                            *
*--------------------------------------------------------------------*
*     117       C                   DUMP                             *
*     118       C                   SETON                            *
*     119                                                            *
*                                                          More...   *
*  Debug . . .   break 112 when BigDate=’1994-09-30’                 *
*                                                                    *
*  F3=End program  F6=Add/Clear breakpoint  F10=Step   F11=Display variable  *
*  F12=Resume       F13=Work with module breakpoints     F24=More keys  *
*                                                                    *
```

图 53　用“BREAK”命令设置一个条件断点。

10.6.3 民族语言排序顺序（NLSS）

非数值条件断点表达式分为如下两 种类型：

Char－8：每个字符包括 8 位。

这相应于 RPG 的数据类型 Character，date，time，timestamp。

Char－16：每字符包括 16 位(DBCS)。

这相应于 RPG 图形数据类型。

NLSS 只适用于 Char－8 类型的非数值条件断点表达式。条件断点表达式可能的值见表 9。

由源码调试程序使用的，用于 Char－8 类型的排序表，是“CRTRPGMOD”或“CRTBNDRPG”命令中 SRTSEQ 参数指定的排序表。

如果给定的排序表是“*HEX”，则没有可用的排序表。这样，源码调试程序使用字符的内码值决定排列顺序。否则，做比较前，给每个字节分配一个。但不管字节之间的移入移出字符，这与 ILE RPG/400 处理比较的方式有所不同；ILE RPG 的比较方式是所有字符，包括移入移出字符，都被赋权。

注：

1、在控制规范表中，由 ALTSEQ(*SRC)指定的替换顺序对 ILE 源码调试程序无效。它使用保存“*HEX”排序表。

2、编译期间，排序表的名字，调试时，源码调试程序使用这个编译时保存的名字来访问排序表。

如果编译时指定的排序表取决于别的而不是*HEX 或*JOBRUN，要注意的是调试开始前，不替换排序表。如果这个表由于损坏或删除而不能被访问，源码调试程序使用*HEX 排序表。

表 9　非数字条件断点表达式

类型	可能的情况
Char－8	字符字段同字符字段比较
	字符字段同字符文字比较(1)
	字符字段同十六进制文字比较(2)
	字符文字同字符字段比较
	字符文字同字符文字比较
	字符文字同十六进制文字比较
	十六进制文字与字符数据字段比较
	十六进制文字同字符文字比较
	十六进制文字同十六进制文字比较
Char－16	图形字段与图形字段比较
	图形字段与图形文字比较(3)
	图形字段与十六进制文字比较(2)
	图形文字(3)与图形字段比较
	图形文字与图形文字比较
	图形文字与十六进制文字比较

十六进制文字与图形字段比较

十六进制文字与图形文字比较

(1)字符型文字的格式如‘abc’；

(2)十六进制文字格式为 X’十六进制数字’；

(3)图形文字格式为 G’0k1k2ki’。shift－out 由’0’代表，shirt－in 由“i”代表。

10.6.4 使用语句号设置和取消断点

使用模块的编译清单中找到的语句号，可设置和取消条件和无条件断点。如果要调试由 DBGVIEW(*STMT)生成的一个模块。这是必要的。

要用“BREAK”调试命令设置一个无条件断点，在调试命令行，键入：

BREAK　过程名/语句号

过程名是你要设置断点的过程的名字，对 ILE RPG/400 来说是模块名。(ILE RPG/400 只允许每一个模块包括一个过程)。语句号在编译清单中，想在其上设置断点的行号。

要使用“BREAK”调试命令设置一个条件断点，在调试命令行键入：

BREAK　过程名/语句号 When　表达式

过程名和语句号与无条件断点情况相同。表达式是当遇到断点时被赋值的条件表达式。

要用“CLEAR”调试命令取消一个无条件或条件断点，在调试命令行键入：

CLEAR　语句号

10.6.5 取消所有断点

通过使用“CLEAR PGM”调试命令，从在“Display Module Source”屏中显示的程序目标中，清除所有的断点，(条件的和无条件的)，要使用这个调试命令，在调试命令行，键入：

CLEAR　PGM

就清除了组成这个程序的所有模块的断点。

10.7　单步执行程序目标

遇到一个断点后，可运行程序的几条语句，然后再次中止程序运行并返回到“Display Module Source”屏。通过使用 ILE 源码调试程序的“步进”功能。使程序从中止的模块的下一个语句运行。通常，是通过一个断点来中止程序运行。通过如下方式，可以单步执行一个程序。

在“Display Module Source”屏上用 F10(Step)或 F22(Step into)，用“STEP”调试命令。

单步执行一个程序一次一条语句，最简单的方法是在“Display Module Source”屏使用 F10(Step)或 F22(Step into)，按 F10 或 F22 时，运行显示在“Display Module Source” 屏中模块的下一条语句，且程序再次被中止。

注意：当使用 F10(Step)或 F22(Step into)，不能指定单步执行的语句个数，按 F10(Step)

或 F22(Step into)只执行一步。

单步执行程序的另一个方法是使用“STEP”调试命令，“STEP”调试命令允许在一个“单步”中运行多于一个的语句，缺省的语句数是 1。要使用“STEP”命令，在调试命令行键入：

STEP　语句数

语句数是程序目标被再次中止前要运行的语句数，例如，如果键入：

STEP　5

则运行下来的 5 条语句，然后程序目标再次中止，且显示“Display Module Source”屏。

在一个调试环境中，当遇到调用另一个程序或过程的语句。可以：

step over　被调用程序或过程，或

step into　被调用程序或过程。

如果选择“Step over”，则 CALL 语句和被调用程序做为一个“单步”来运行，在调用程序停在下一步前，运行完调用程序，“Step over”是缺省的步进模式。

如果选择“Step into”，则被调用的程序目标中的每一条语句作为一个单步来运行。如果正在运行的程序由于被调用程序中的故障而停止运行，则强行停在被调用的程序的故障点，并显示“Display Module Source”屏。

注意：不能对 RPG 子程序使用“Step over”或“Step into”。

10.7.1“Step over”程序目标

可以通过以下方法来使用“Step over”：

在“Display Module Source”屏中使用 F10(Step)

使用“Step over”调试命令。

在一个调试环境中，可使用“Display Module Source”屏中的 F10(Step)来“Step over”一个被调用程序目标。如果将要运行的下条语句是一条调用另一个程序目标的调用语句。按 F10(Step)使被调用的程序目标在调用程序目标再次停止前运行完。

另一个方法是，在一个调试环境中，使用“STEP OVER”调试命令来“Step over”一个被调用程序目标。要使用“STEP　OVER”调试命令，在命令行上键入：

STEP　语句数　OVER

变量语句数是程序目标再次被中止前，在下一步运行的语句数。如果没用这个变量，缺省值是 1。如果要运行的语句中有调用另一个程序语句，ILE 源码调试程序将“STEP OVER”这个被调用的程序。

10.7.2“STEPPing into”程序目标

可以用下面的方法来“Step into”程序目标：

在“Display Module Source”屏中用 F22(“Step into”)来“Step into”一个被调用的程序。如果即将运行的下一条语句是一个调用语句，按 F22(Step into)将运行被调用程序中的第一个可执行语句。被调用的程序被显示在“Display Module Source”屏中。

注：被调用的程序一定要有相应的调试数据，以便它能显示在“Display Module Source”屏中。

替换另外一个方法是，使用“Step into”调试命令来“Step into”一个被调用的程序，要使用“Step into”调试命令，在调试命令行上键入：

STEP 语句数 INTO

语句数是程序目标再次中止前，下一步要运行的语句数，如果没有这个变量，缺省值是“1”，如果要运行的语句中包括调用另一个程序语句，调试程序目标的每一条语句，都做为一个单步执行，调用程序中的单步结束后，则它将被显示在“Display Module Source”屏中。例如，在命令行键入：

STEP 5 INTO

则程序接下来执行 5 条语句。如果第三条语句是一个调用语句，则运行调用程序两条语句后运行，被调用程序的头三条语句。“STEP INTO”调试命令也处理 CL 的“CALL”命令，你可利用这一点来单步调试程序。启动源码调试程序后，从初始“Display Module Source”屏，键入：

STEP 1 INTO

这将设置步计数为 1 ，用 F12 返回到命令行，然后，调用这个程序，程序将停止在带调试数据的第一条语句上。

10.7.2.1 用 F22“STEP into”一个程序的例子

在本例中，使用 F22(Step into)来从程序 DEBUGEX 中“Step into”程序 RPGPGM。

1、确保“Display Module Source”屏显示“DBGEX”的源码。

2、在第 90 行上设置一个无条件断点，这是“CALL”操作前的最后一个可运行语句，键入“Break”按“Enter”。

3、按 F3(End program)结束“Display Module Source”屏。

4、调用这个程序，程序在断点 90 停止，显示如图 54。

```
*                                                                        *
*                          Display Module Source                         *
*                                                                        *
*  Program:   DEBUGEX       Library:   MYLIB        Module:   DBGEX       *
*      86         *                                               *
*      87         * FLD1a 是 FLD1 的复盖字段。由于 FLD1 初始化为‘ABCDE’，那么 *
*      88         * FLD1a 的值为‘A’。在下面的 MOVE 操作后，FLD1a 的值为‘1’。*
*      89
*--------------------------------------------------------------    *
*      90        C                   MOVE       '1'          Fld1a(1)        *
```

```
*      91                                                                  *
*      92
*------------------------------------------------------------  *
*      93        * 调用程序 RPG PGM，它是另外的程序目标。              *
*      94
*------------------------------------------------------------  *
*      95       C     Plist1        PLIST                                  *
*      96       C                   PARM                   Parm1           *
*      97       C                   CALL      'RPGPGM'     Plist1          *
*      98                                                                  *
*      99
*------------------------------------------------------------  *
*     100        * 调用 C-Proc，它从这个程序输入 ExportFldx.               *
*                                                             More...   *
*  Debug . . .                                                          *
    *
*
*  F3=End program  F6=Add/Clear breakpoint  F10=Step  F11=Display variable    *
*  F12=Resume       F13=Work with module breakpoints     F24=More keys        *
*  Breakpoint at line 90.                                                     *
*                                                                           *
```

图 54"Stepping into"RPGPGM 前的"DBGEX"的"Display Module Source"　屏

5、按 F22(Step into)运行程序的一条，然后出现程序“RPGPGM”的“Display Module Source”屏显示，如图 55。

在这种情况下，执行“RPGPGM”的第一个可运行语句（第 13 行），然后程序停止。

注：当使用 F22 时，不能指定“步进”语句数，按 F22 执行一个单步。

```
*                                                                        *
*                          Display Module Source                         *
*                                                                        *
*  Program:   RPGPGM        Library:   MYLIB         Module:   RPGPGM    *
*        1        *=============================================================  *
*        2        * RPG PGM_由 DEBUGEX 调用的程序，解释 ILE 源码调试的 STEP 功能。   *
*        3        *                                                     *
```

```
*       4        *                                                          *
*       5        *  这个程序从 DEBUGEX 接收一个参数 InputParm。显示它，然后返回。    *
*       6        *                                                    *
*       7        *============================================================ *
*       8                                                                    *
*       9       D InputParm       S              4P 3                      *
*      10                                                                    *
*      11       C     *ENTRY       PLIST                                    *
*      12       C                  PARM                    InputParm       *
*      13       C     InputParm    DSPLY                                    *
*      14       C                  SETON                                    *
*                                                                           *
*                                                                  Bottom   *
*  Debug . . .                                                              *
*                                                                           *
*  F3=End program   F6=Add/Clear breakpoint   F10=Step   F11=Display variable *
*  F12=Resume       F13=Work with module breakpoints     F24=More keys      *
*  Step completed at line 13.                                               *
*                                                                           *
```

图 55 “Step into”RPGPGM

10.7.3“Stepping Over”过程

如果你在“步进”调试命令中指定“Over”，则对程序和函数的调用都看作一条语句，这是缺省步进模式，通过如下方式，启动“Step－over”功能：

用“Step Over”调试命令

用 F10 (Step)键

10.7.4“Stepping into”过程

如果在“STEP”命令中指定“into”，则被调用过程或函数中的每一条语句都做为一条步进语句。如果在四条语句中有一条语句是调用有十六条语句的过程，那么，执行这四条语句就导致执行 20 条语句。通过如下方式，可以启动“stepinto”功能：

用“Step Into”调试命令

用 F22(Step Into)键

被调用的过程必须有相应的调试数据，以便能显示在“Display Module Source” 屏上。

10.8 显示数据和表达式

你可以显示字段、数据结构、数组的内容，且可以给表达式赋值，有两种方式可用来显示或赋值。

F11(Display Variable)

EVAL 调试命令

“EVAL”命令中使用字段的范围是用 QUAL 命令定义，但不必特别定义 ILE RPG/400 模块中的字段范围。因为它们都是全局范围。显示数据或表达式的简单方式是在“Display Module Source”屏上用 F11 键(Display Variable)，把光标移到要显示的字段上并按 F11 键。这个字段的当前值被显示在“Display Module Source”屏底部的信息行上。

当用 F11 键给结构、记录或数组赋值时，返回的信息也许有多行，这些信息被显示在“Evaluate Expression”屏上，显示信息的整个文本。一旦已查看完了“Evalute Expression”屏的信息，按执行键返回到“Display Module Source”屏上。

要用 EVAL 调试命令来显示数据，在调试命令行键入：

EVAL　字段名

字段名是你想要显示或赋值的字段、数据结构或数组的名字。如果 EVAL 调试命令是在“Display Module Source”屏输入且值可以用一行显示完，“值”将被显示在信息行。否则，将被显示在“Evaluate Expression”屏中。

图 56 显示的是使用 EVAL 调试命令来显示子字段 Last Name 内容的例子。

```
*                                                                    *
*                          Display Module Source                     *
*                                                                    *
*  Program:   DEBUGEX       Library:   MYLIB         Module:   DBGEX  *
*      57       D  LastName                    10A   INZ('Jones    ')  *
*      58       D  FirstName                   10A   INZ('Fred     ')  *
*      59                                                            *
*      60        *-------------------------------------------------- *
*      61        * 定义 CALL 和 CALLB 的参数                          *
*      62        * 1、在调用 RPGPROG 程序时用 Parm1。                 *
*      63        * 2、定义 ExportFld 为输出与 ILE C/400 过程 C-Proc 一起使用。  *
*      64        *                                                   *
*      65        *-------------------------------------------------- *
*      66       DParm1           S              4P 3 INZ(6.666)      *
*      67       DExportFld       S              6A   INZ('export') EXPORT  *
```

```
*      68                                                                   *
*      69       *=============================================================  *
*      70       * 当前操作修改值或调用其它目标。                                *
*      71       *=============================================================  *
*                                                                  More...  *
*  Debug . . .   eval LastName                                              *
*                                                                           *
*  F3=End program   F6=Add/Clear breakpoint   F10=Step   F11=Display variable    *
*  F12=Resume       F13=Work with module breakpoints     F24=More keys      *
*  LASTNAME = 'Jones     '                                                  *
*                                                                           *
```

图 56　使用 EVAL 命令显示一个字段

图 57 显示不同类型的 RPG 字段使用 EVAL 命令的情况

这些字段来自 3.1.13 图 65 中的源码，其余的例子在源码调试程序的联机帮助中提供。

```
Scalar Fields                           RPG Definition

> EVAL String                           6A    INZ('ABCDEF')
  STRING = 'ABCDEF'
> EVAL Packed1D0                        5P 2 INZ(-93.4)
  PACKED1D0 = -093.40
> EVAL ZonedD3D2                        3S 2 INZ(-3.21)
  ZONEDD3D2 = -3.21
> EVAL Bin4D3                           4B 3 INZ(-4.321)
  BIN4D3 = -4.321
> EVAL DBCSString                       3G    INZ(G'~BBCCDD~')
  DBCSSTRING = '*BBCCDD*'
> EVAL NullPtr                           *    INZ(*NULL)
  NULLPTR = SYP:*NULL

Based Fields
> EVAL String                           6A    INZ('ABCDEF')
  STRING = 'ABCDEF'
```

```
> EVAL BasePtr                              *    INZ(%ADDR(String))
  BASEPTR = SPP:C01947001218
> EVAL BaseString                          6A    BASED(BasePtr)
  BASESTRING = 'ABCDEF'

Date, Time, Timestamp Fields
> EVAL BigDate                              D    INZ(D'9999-12-31')
  BIGDATE = '9999-12-31'
> EVAL BigTime                              T    INZ(T'12.00.00')
  BIGTIME = '12.00.00'
> EVAL BigTstamp                            Z    INZ(Z'9999-12-31-12.00.00.000000
  BIGTSTAMP = '9999-12-31-12.00.00.000000'
```

图 57　基于模块 DBGEX 的 EVAL 命令的例子

10.8.1 显示一个数组的内容

在 EVAL 中指定一个数组名能显示整个数组，要显示数组的一个元素，在圆括号中指定要显示的元素的下标，也可用%INDEX 调试内部函数。

要显示某个范围的元素，使用如下范围标志。

EVAL　字段名(n・・・m)

字段名是数组的名字。变量 n 是一个代表范围开始的数字，变量 m 是一个代表范围结束的数字。

图 58 是 DBGEX 中带数组的 EVAL 用法。

```
> EVAL Arry                                 3S 2 DIM(2) INZ(1.23)
  ARRY(1) = 1.23    ** Display full array **
  ARRY(2) = 1.23

> EVAL Arry(2)      ** Display second element **
  ARRY(2) = 1.23

> EVAL Arry(1..2)   ** Display range of elements **
  ARRY(1) = 1.23
  ARRY(2) = 1.23
```

图 58　EVAL 命令用于数组的例子

10.8.2 显示一个表的内容

对一个表使用 EVAL 将显示当前表元素。使用范围标识，可显示整个表。例如，要显示一个三元素表，键入：

EVAL　Table A(1・・3)

可使用“%INDEX”内部函数修改当前元素，图 59 是 DBGEX 中表的 EVAL 使用方法。

```
                                    3    DIM(3) CTDATA

                                    Compile-time data:  **
> EVAL TableA            ** Show value at                    aaa
  TABLEA = 'aaa'            current index                    bbb
                                                             ccc
> EVAL TableA(1)         ** Specify index 1 **
  TABLEA(1) = 'aaa'
> EVAL TableA(2)         ** Specify index 2 **
  TABLEA(2) = 'bbb'

> EVAL TableA(1..3)      ** Specify the whole table **
  TABLEA(1) = 'aaa'
  TABLEA(2) = 'bbb'
  TABLEA(3) = 'ccc'

> EVAL TableA=%INDEX(3)  ** Change current index to 3 **
> EVAL TableA
  TABLEA = 'ccc'
```

图 59　EVAL 命令用于表的例子

10.8.3 显示数据结构

要显示一个数据结构或它的子字段的内容。只要在 EVAL 后边使用数据结构名就可看整个内容，或用子字段名看一部分内容。

当显示一个多次出现的数据结构，用数据结构名的 EVAL 将显示使用当前下标的子字段。要指定某一个数据结构。在紧接着数据结构名的圆括号中指定下标。例如，要显示 DS1 的第二个字段的内容，键入：

EVAL　DS1(2)

类似的。要查看一个子字段中某个字段的内容。使用下标标识。

如果一个子字段被定义为复盖另一个子字段的数组。要看复盖子字段的内容。可使用%INDEX 内部函数来指定这个字段，用下标标识来指定数组。显示一个数组复盖某个子字段的另外方法是使用下面的标识：

EVAL　子字段名(出现标志，数组下标)

子字段名是要显示的子字段的名字。出现标志是要显示的数组出现的位数。数组下标是要显示的数组元素的号码。

图 60 是 EVAL 与 DBGEX 中定义的数据结构使用的例子。

```
** Note that you can enter the data structure name or a subfield name. **

> EVAL DS3
  TITLE OF DS3 = 'Mr.  '                5A   INZ('Mr.  ')
  LASTNAME OF DS3 = 'Jones     '        10A  INZ('Jones    ')
  FIRSTNAME OF DS3 = 'Fred      '       10A  INZ('Fred     ')

> EVAL LastName
  LASTNAME = 'Jones     '

> EVAL DS1                                   OCCURS(3)
  FLD1 OF DS1 = 'ABCDE'                 5A   INZ('ABCDE')
  FLD1A OF DS1(1) = 'A'                 1A   DIM(5) OVERLAY(Fld1)
  FLD1A OF DS1(2) = 'B'                 5B 2 INZ(123.45)
  FLD1A OF DS1(3) = 'C'
  FLD1A OF DS1(4) = 'D'
  FLD1A OF DS1(5) = 'E'
  FLD2 OF DS1 = 123.45

> EVAL DS1=%INDEX(2)     ** Change the occurrence of DS1 **
```

```
  DS1=%INDEX(2) = 2

> EVAL Fld1              ** Display a Subfield  **
  FLD1 = 'ABCDE'            (current occurrence)
> EVAL fld1(2)
  FLD1(2) = 'ABCDE'         (second occurrence)

> EVAL Fld1a             ** Display an Array Overlay Subfield **
  FLD1A OF DS1(1) = 'A'     (current occurrence)
  FLD1A OF DS1(2) = 'B'
  FLD1A OF DS1(3) = 'C'
  FLD1A OF DS1(4) = 'D'
  FLD1A OF DS1(5) = 'E'

> EVAL Fld1a(2,1)        ** Display 2nd occurrence, 1st element  **
  FLD1A(2,1) = 'A'

> EVAL Fld1a(2,1..2)     ** Display 2nd occurrence, 1st - 2nd elements **
  FLD1A(2,1) = 'A'
  FLD1A(2,2) = 'B'
```

图 60　对数据结构使用 EVAL

要显示一个没有定义的子字段的数据结构，必须用下面讨论的 EVAL 的字符显示功能。

10.8.4 显示指示器

指示器被定义为一个字节长的字符型字段。几个除了象*INLR 这样的指示器以外，可显示象*INxx 或*IN(xx)这样的指示器。由于系统把指示器做为一个数组存放，使用范围标识来显示它们的全部或某一部分。例如，如果输入 EVAL　*IN，将得到一个从指示器 01 到 99 的列表。要显示指示器 01 到 06，输入：EVAL *IN(1，6)。

图 61 是使用设置在 DBGEX 中的指示器的每一种用法。

```
> EVAL IN02
  Identifier does not exist.
> EVAL *IN02
```

```
  *IN02 = '1'
> EVAL *IN(02)
  *IN(02) = '1'

> EVAL *INLR
  *INLR = '0'
> EVAL *IN(LR)
  Identifier does not exist.

> EVAL *IN(1..6)        ** To display a range of indicators **
  *IN(1) = '0'
  *IN(2) = '1'
  *IN(3) = '0'
  *IN(4) = '1'
  *IN(5) = '0'
  *IN(6) = '1'
```

图 61 数组使用 EVAL 命令的例子

10.8.5 用十六进制值来显示字段

可以使用 EVAL 调试命令用十六进制格式显示字段的值。要以十六进制格式显示一个变量，在调试命令行，键入：

EVAL　字段名：X　字节数

字段名是要以十六进制格式显示的字段名字，‘X’指定了这个字段以十六进制格式来显示。字节数指出要显示的字节数。如果‘X’后没有规定长度，字段的大小用做长度。最少显示十六个字节。如果字段的长度小于十六个字节，则剩余的空间填‘0’，直到 16 个字节长。

例如，字段‘String’被定义做 6 个字符串，要给出前 3 个字符的十六进制表示，输入：

EVAL　String：X 3

结果显示：

00000　C1C2C300　00000000　00000000　00000000　－ABC・・・・

10.8.6 以字符格式显示字段

可以用 EVAL 调试命令来以字符格式显示一个字段。要以字符格式显示一个变量，在调试命令行键入：

EVAL　字段名：C　字符数

字段名是要以字符格式显示的字段名字。‘C’规定要显示的字符数。

例如，在程序 DEBUGEX 中，数据结构 DS2 没有定义任何子字段，用几个“MOVE”操作把值送到这个子字段中。

由于没有定义子字段，不能显示这个数据结构。要看它的内容，可使用 EVAL 的字符显示功能。

EVAL　DS2：C　20

结果显示：

DS2：C　20＝‘aaaaaaaaabbbbbbbbbb’

10.8.7 使用调试内部函数

使用 ILE 源码调试器时，可用如下内部函数：

%SUBSTR　　取一个字符型字段的子串

%ADDR　　取回一个字段的地址

%INDEX　　修改一个表或多次出现的数据结构的下标

%SUBSTR 内部函数能取一个字符串变量的子串，第一个参数必须是一个字符串标识符，第二个参数是起始位置，第三个参数是单字节或双字节字符的数目。第二和第三个参数必须是正的整型文字，参数间用一个或多个空格间隔。

使用%SUBSTR 可以做如下事情：

显示一个字符字段的一部分

分配一个字符字段的一部分

在一个条件断点表达式中使用字符字段的一部分

图 62 是对 3.1.13 图 65 中源码使用%SUBSTR 的例子。

```
 > EVAL String
   STRING = 'ABCDE '
** Display the first two characters of String **
 > EVAL %substr (String 1 2)
   %SUBSTR (STRING 1 2) = 'AB'


 > EVAL TableA
   TABLEA = 'aaa'
** Display the first character in the first table element **
 > EVAL %substr(TableA 1 1)
   %SUBSTR(TABLEA 1 1) = 'a'


 > EVAL BigDate
   BIGDATE = '1994-10-23'
```

```
** Set String equal to the first four characters of BigDate **
 > EVAL String=%substr(BigDate 1 4)
   STRING=%SUBSTR(BIGDATE 1 4) = '1994  '

 > EVAL Fld1           (5 characters)
   FLD1 = 'ABCDE'
 > EVAL String         (6 characters)
   STRING = '123456'
** Set the characters 2-5 of String equal to the
                first four characters of Fld1 **
 > EVAL %substr(String 2 4) = %substr(Fld1 1 4)
   %SUBSTR(STRING 2 4) = %SUBSTR(FLD1 1 4) = 'ABCD'
 > EVAL String
   STRING = '1ABCD6'

** You can only use %SUBSTR on character or graphic strings! **
 > EVAL %substr (Packed1D0 1 2)
   String type error occurred.
```

图 62　DBGEX 使用%SUBSTR 的例子

要修改当前下标，可使用%INDEX 内部函数。在紧跟着函数名的括号内指定下标。在 3.1.8.2 的图 59 和 3.1.8.3 的图 60 中的“表”部分可找到%INDEX 的例子。

注意：%INDEX 将把当前下标修改成新指定的下标。这样，在其后的 EVAL 语句中任何引用这个表或多重数据结构的源语句，将以这个下标操作。要注意其结果是否符合原来想法。

10.9　修改字段的值

通过用 EVAL 命令和一个分配操作符（=），可修改字段的值。

EVAL 命令使用的字段范围由使用 QUAL 命令来定义。但不必特别定义在 ILE　RPG/400 模块中的数据项的范围。因为它们都是全局范围的。

要修改字段的值，在调试命令行键入：

EVAL　字段名＝值

字段名是要修改的变量名字，值是要赋给变量的一个标识值或文字或常量值，例如：

EVAL　COUNTER＝3

把 COUNTER 的值变为 3，并在“Display Module Source”屏的信息行显示：

COUNTER＝3 ＝ 3

使用 EVAL 调试命令把数字的、字母的、字母数字的值分配给字段，也可以在赋值表达式上使用%SUBSTR。

给一个字符字段赋值时，按如下规则做：

如果源表达式的长度小于结果表达式的长度，则在结果表达式中数据左对齐剩余位置填空格。

如果源表达式的长度大于结果表达式的长度，则在结果表达式中数据左对齐并截止到结果表达式的长度。

注：可把如下任何一个值赋给图形字段：

另一图形字段

形如 G’0k1k2i’的一个图形文字

形如 X’her digits’的一个十六进制文字

把文字赋值给字段时，遵照正常的 RPG 规则：

字符应用引号括起来

图形应被指定做 G’0DDDDi’，’0’是移出，’i’是移入

十六进制应被引起来，跟在一个’X’后

数字不用引号

注意：使用 EVAL 调试命令，不能把一个图形常量分配一个字段，EVAL 调试命令不支持图形常量。

图 63 是对 3.1.13 图 65 中源码字段做修改的例子。其他例子请在源码调试程序的联机帮助中查看。

```
** Target Length = Source Length **

 > EVAL String='123456'      (6 characters)
   STRING='123456' = '123456'
 > EVAL ExportFld            (6 characters)
   EXPORTFLD = 'export'
 > EVAL String=ExportFld
   STRING=EXPORTFLD = 'export'

** Target Length < Source Length **

 > EVAL String               (6 characters)
   STRING = 'ABCDEF'
 > EVAL LastName             (10 characters)
   LASTNAME='Williamson' = 'Williamson'
```

```
 > EVAL String=LastName
   STRING=LASTNAME = 'Willia'

** Target Length > Source Length **

 > EVAL String               (6 characters)
   STRING = '123456'
 > EVAL TableA               (3 characters)
   TABLEA = 'aaa'
 > EVAL String=TableA
   STRING=TABLEA = 'aaa   '

** Using %SUBSTR **

 > EVAL BigDate
   BIGDATE = '1994-10-23'
 > EVAL String=%SUBSTR(BigDate 1 4)
   STRING=%SUBSTR(BIGDATE 1 4) = '1994  '

** Substring Target Length > Substring Source Length **
 > EVAL string = '123456'
   STRING = '123456' = '123456'
 > EVAL LastName='Williamson'
   LASTNAME='Williamson' = 'Williamson'
 > EVAL String = %SUBSTR(Lastname 1 8)
   STRING = %SUBSTR(LASTNAME 1 8) = 'Willia'

** Substring Target Length < Substring Source Length **
 > EVAL TableA
   TABLEA = 'aaa'
 > EVAL String
   STRING = '123456'
 > EVAL String=%SUBSTR(TableA 1 4)
   Substring extends beyond end of string.      ** Error **
 > EVAL String
   STRING = '123456'
```

图 63　改变 DBGEX 中字段值的例子

10.10 显示字段的属性

使用属性调试命令(ATTR)显示一个字段的属性，这些属性是记录在符号调试表中的变量大小(以字节计)和变量类型。

图 64 是显示字段属性的一些例子。字段是在 3.1.13 图 65 的源码，其他例子请在源码调试程序的联机帮助查看。

```
> ATTR NullPtr
  TYPE = PTR, LENGTH = 16 BYTES
> ATTR ZonedD3D2
  TYPE = ZONED(3,2), LENGTH = 3 BYTES
> ATTR Bin4D3
  TYPE = BINARY, LENGTH = 2 BYTES
> ATTR Arry
  TYPE = ARRAY, LENGTH = 6 BYTES
> ATTR tablea
  TYPE = FIXED LENGTH STRING, LENGTH = 3 BYTES
> ATTR tablea(2)
  TYPE = FIXED LENGTH STRING, LENGTH = 3 BYTES
> ATTR BigDate
  TYPE = FIXED LENGTH STRING, LENGTH = 10 BYTES
> ATTR DS1
  TYPE = RECORD, LENGTH = 9 BYTES
> ATTR SpcPtr
  TYPE = PTR, LENGTH = 16 BYTES
> ATTR String
  TYPE = FIXED LENGTH STRING, LENGTH = 6 BYTES
> ATTR *IN02
  TYPE = CHAR, LENGTH = 1 BYTES
> ATTR DBCSString
  TYPE = FIXED LENGTH STRING, LENGTH = 6 BYTES
```

图 64　显示基于 DBGEX 的字段属性的例子

10.11 用名字表示一个字段，表达式或命令

可使用 EQUATE 调试命令把一个字段，表达式或调试命令用一个名字表示，

用做简写，然后，可以单独使用名字或在表达式中使用它。如在另一表达式中使用名字，它的值由表达式赋值确定。这些名字在这个调试环境中一直起作用，调试后，取消这个名字，被清除。

要用名字表示一个字段，表达式或调试命令在调试命令行键入：

EQUATE　简写名　定义

简写名是表示一个字段、表达式或调试命令的名字。定义是要用这个名字表示的字段、表达式或调试命令。

例如，要定义一个名叫 DC 的简写名以显示字段 COUNTER 的内容，在命令行键入：

EQUATE　DC　EVAL　COUNTER

那么当在调试命令行键入 DC 时，就执行 EVAL　COUNTER 命令。

在一个 EQUATE 命令中最多可键入 144 个字符，如果先前已用 EQUATE 命令定义了一个名字，现在的定义要起作用，先前的定义被取消。如果使用没有定义的名字，显示一个错误信息。

要查看一个调试环境中被 EQUATE 命令定义的名字，在命令行键入：

DISPLAY　EQUATE

在“Evaluate　Expression”屏中显示一个活动名字的列表。

10.12 ILE RPG/400 支持的源码调试民族语言

当你处理 ILE RPG/400 支持的民族语言时，注意下面的情况。

在“Display Module Source”屏上显示一个视图时，源码调试程序把所有的数据转换成调试作业的编码字符集标识(CCSID)。

给字段赋值文字时，用引号把文字括起来（例如，’abc’），源码调试程序不执行 CCSID 转换。

有关 NLS 规则的详细内容，参见《ILE 概念》的有关章节。

10.13 源码调试的例子

图 65 给出了程序 DEBUGEX 的主过程的源码。本章中大多数的例子和屏幕都是基于这个源码。图 66 和 67 分别是被调用程序 RPGPGM 和过程 CPROC 的源码。

程序 DEBUGEX 的功能是给出 ILE 源码调试器程序和 ILE RPG/400 格式转储的不同方面。下页给出转储的样本。

下面给出如何生成程序 DEBUGEX 的步骤：

1、要用图 65 中的源码生成模块 DBGEX，键入：

CRTRPGMOD　MODULE(MYLIB/DBGEX)
SRCFILE(MYLIB/QRPGLESRC)　　DBGVIEW(*ALL)
TEXT(‘Main module for Sample Debug Program’)

2、要用图 67 中的源码生成“C”模块，键入：

CRTCMOD MODULE(MYLIB/CPROC)
SRCFILE(MYLIB/QCLESRC)
DBGVIEW(*SOURCE)
TEXT(‘C Procedure for Sample Debug Program’)

3、要生成程序 DEBUGEX，键入：

CRTPGM PGM(MYLIB/DEBUGEX) MODULE(MYLIB/DBGEX
MYLIB/CPROC)
TEXT(‘Sample Debug Program’)

第一个模块 DBGEX 是这个程序的输入模块，调用它时，程序在一个新的活动组 中运行。（即：*NEW）。

4、要用图 66 中的源码生成被调用的 RPG 程序，键入：

CRTBNDRPG PGM(MYLIB/RPGPGM) DFTACTGRP(*NO)
DBGVIEW(*SOURCE) ACTGRP(*NEW)
TEXT(‘RPG Program for Sample Debug Program’)

生成 RPGPGM 在 OPM 缺省活动组中运行,但我们决定让它在与 DEBUGEX 相同的活动组中运行。既然 DEBUGEX 只需要一个临时的活动组，所以对这些程序规定 ACTGRP 为*NEW。

```
*================================================================*
* DEBUGEX_程序用来解释 ILE RPG 源码调试。对不同的数据类型和        *
*          数据结构提供样本，也能用来产生样本格式化转储。          *
*================================================================*

*----------------------------------------------------------------*
* DEBUG 键字能用来做格式化转储。                                  *
*----------------------------------------------------------------*
H DEBUG

*----------------------------------------------------------------*
* 对不同的数据类型定义独立字段。                                  *
*----------------------------------------------------------------*
D String          S              6A   INZ('ABCDEF')
D Packed1D0       S              5P 2 INZ(-93.4)
D ZonedD3D2       S              3S 2 INZ(-3.21)
D Bin4D3          S              4B 3 INZ(-4.321)
D Bin9D7          S              9B 7 INZ(98.7654321)
```

```
D DBCSString      S              3G   INZ(G'*BBCCDD*')

 *   Pointers
D NullPtr         S               *   INZ(*NULL)
D BasePtr         S               *   INZ(%ADDR(String))
D ProcPtr         S               *   ProcPtr INZ(%PADDR('c_proc'))
D BaseString      S              6A   BASED(BasePtr)
D BaseOnNull      S             10A   BASED(NullPtr)
 *
D Spcptr          S               *
D SpcSiz          S              9B 0 INZ(8)

 *   Date, Time, Timestamp
D BigDate         S               D   INZ(D'9999-12-31')
D BigTime         S               T   INZ(T'12.00.00')
D BigTstamp       S               Z   INZ(Z'9999-12-31-12.00.00.000000')

 *   Array
D Arry            S              3S 2 DIM(2) INZ(1.23)

 *   Table
D TableA          S              3    DIM(3) CTDATA

 *----------------------------------------------------------------*
 * 定义不同类型的数据结构。                                       *
 *----------------------------------------------------------------*
D DS1             DS                  OCCURS(3)
D  Fld1                          5A   INZ('ABCDE')
D  Fld1a                         1A   DIM(5) OVERLAY(Fld1)
D  Fld2                          5B 2 INZ(123.45)
 *
D DS2             DS             10   OCCURS(2)
 *
D DS3             DS
D  Title                         5A   INZ('Mr.  ')
D  LastName                     10A   INZ('Jones     ')
D  FirstName                    10A   INZ('Fred      ')
```

```
 *----------------------------------------------------------------*
 * 为 CALL 和 CALLB 定义参数：                                    *
 * 1、当调用 RPGPROG 程序时用 Parm1。                             *
 * 2、定义 ExportFld 为输出，与 ILE C/400 过程 C-Proc 一起使用。  *
 *----------------------------------------------------------------*
DParm1            S               4P 3 INZ(6.666)
DExportFld        S               6A   INZ('export') EXPORT

 *================================================================*
 * 此操作的值或调用其它目标。                                     *
 *================================================================*
 *----------------------------------------------------------------*
 * 往数据结构 DS2 中传送 ‘a’。传送之后，DS2 的第一次出现包括 10 个 *
 * 字符的 ‘a’。                                                   *
 *----------------------------------------------------------------*
C                   MOVE      *ALL'a'       DS2

 *----------------------------------------------------------------*
 * 修改 DS2 的出现为 2，并把'b'传送给 DS2，使头 10 个字节是'a'，  *
 * 后面是 10 个字节的 ‘b’。                                       *
 *----------------------------------------------------------------*
C     2             OCCUR     DS2
C                   MOVE      *ALL'b'       DS2

 *----------------------------------------------------------------*
 * Fld1a 是 Fld1 的复盖字段。Fld1 初始化为'ABCDE'。所以 Fld1a 中是'A'。*
 * 在下面的传送后，Fld1a 中是'1'。                                *
 *----------------------------------------------------------------*
C                   MOVE      '1'           Fld1a(1)

 *----------------------------------------------------------------*
 * 调用程序 RPG PGM，它是另外的长程素序目标。                     *
 *----------------------------------------------------------------*
C     Plist1        PLIST
C                   PARM                    Parm1
C                   CALL      'RPGPGM'      Plist1

 *----------------------------------------------------------------*
```

```
     * 调用 c_proc, 它从这个程序输入 ExportFld。                          *
     *----------------------------------------------------------------*
    C                   CALLB     'c_proc'
    C                   PARM                    SpcSiz
    C                   PARM                    SpcPtr

     *----------------------------------------------------------------*
     * 在下面的 SETON 之后,  *IN02 = '1'。                              *
     *----------------------------------------------------------------*
    C                   SETON                                        020406
    C                   IF        *IN02
    C                   MOVE      '1994-09-30'  BigDate

     *----------------------------------------------------------------*
     *  现在开始格式化转储，在 LR 为 ON 时返回。                          *
     *----------------------------------------------------------------*
    C                   DUMP
    C                   SETON                                        LR

     *================================================================*
     * 表的编译时数据段。                                               *
     *================================================================*
**
aaa
bbb
ccc
```

图 65　模块 DBGEX 的源码，DBGEX 是程序 DEBUGEX 的主模块。

```
     *================================================================*
     *  RPGPGM - DEBUGEX 调用的程序，用来解释 ILE 源码调试的 STEP 功能,   *
     *  程序从 DEBUGEX 接收参数 InputParm，显示它，然后返回。             *
     *================================================================*

    D InputParm       S              4P 3

    C     *ENTRY        PLIST
```

```
C                   PARM                    InputParm
C     InputParm     DSPLY
C                   SETON                                         LR
```

图 66　程序 RPGPGM 的源码，它被 DBGEX 调用。

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
extern char EXPORTFLD[6];
void c_proc(int *size, char **ptr)
{
   *ptr = malloc(*size);
   memset(*ptr, 'P',*size );
   printf("import string: %6s.\n",EXPORTFLD);
}
```

图 67　C 过程 Cproc 的源码，它被 DBGEX 调用。

第十一章　　处理异常情况

这一章讲述 ILE RPG/400 异常处理以及如何使用：

异常处理例程

ILE RPG/400 特定处理例程

ILE 条件处理例程

取消处理例程

ILE RPG/400 支持下面类型的异常处理例程：

RPG－特定处理例程，例如，使用错误指示器或*PSSR 或*INFSR 错误子例程。

ILE 条件处理例程：用户写的异常处理程序，在运行时，使用可连接 API CEEHDLR 的 ILE 条件处理例程。ILE 取消例程，可用于过程非正常结束时使用。

由于程序中存了对异常的处理设计，所以很少出现程序的非正常结束。一般都因功能检查而出现异常。ILE 条件处理允许用一种连续的方式处理不同语言的应用程序的异常。

可使用 RPG 异常处理程序来处理一个 RPG 程序中可能遇到的大多数情况。RPG 提供的异常处理的最小级别是在操作中的错误指示器，要学会怎样使用它们，请阅读本章中下面的部分：

3.2.1.1 的“ILE RPG/400 异常处理”

3.2.3.1 的“指定错误指示器”

另外，要知道 ILE 异常处理程序如何工作，请阅读：

3.2.1 中的“异常处理概述”（基本的概念）

3.2.3 节中的“使用 RPG－特别处理”

《ILE 概念》中关于错误处理的部分。

关于异常处理和 RPG 逻辑周期的资料，见《ILE RPG/400 参考手册》

注：在本书中，术语‘异常处理’通常指“异常处理”和“错误处理”，然而，为严密起见，术语“错误”用于“错误指示器”和“错误子例程”。

11.1 异常处理概述

异常处理是处理：

解释运行时错误结果发送的异常信息。

有选择的修改已收到的异常信息（即处理）。

通过异常信息传送给部分代码以便采取必要的动作来恢复异常。

当出现一个运行时错误时，生成一个异常信息。根据出现的错误，异常信息有如下类型：

*ESCAPE	指示已检测到严重错误
*STATUS	描述由程序做的作业状态
*NOTIFY	描述一个需要更正的动作或回答来自调用程序的问题
Function Check	指示出现以上三种异常之一，并且没有处理异常

异常信息与调用堆栈入口在一起。每个调用堆栈入口依次连着为那个入口定义的异常处理的清单。（关于“调用堆栈”的详细讨论，参见 2.6.1.3“调用堆栈”）。

图 68 是一个 OPM 程序调用一个由几个模块组成的 ILE 程序的调用堆栈。在下面的讨论中请参照此图。

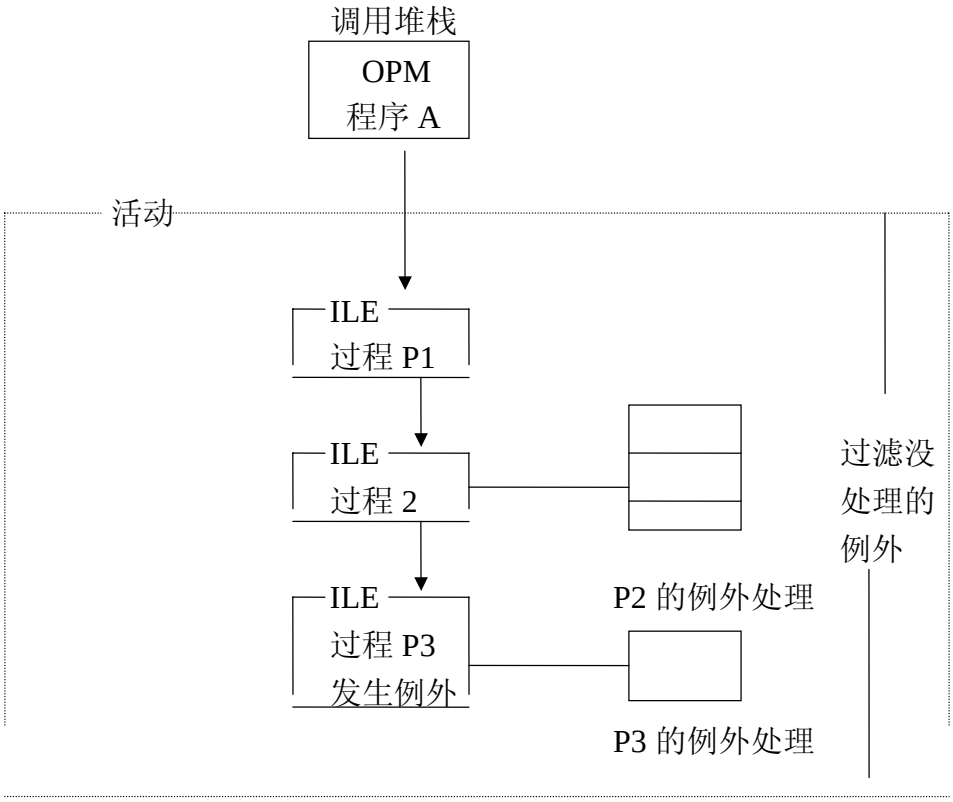
通常，当出现一个异常时，对调用堆栈入口的处理也会处理这个异常。如果这个异常没有被列表中某项处理，那么它被认为是不可处理的，对这个不可处理的异常，将执行下面的缺省动作。

1、如果异常是一个功能检查，调用堆栈入口从堆栈中清除。

2、异常被移到先前的调用堆栈入口

3、对这个调用堆栈入口重新启动异常处理。允许先前的调用堆新启动异常处理例程来　　处理一个异常的动作被看做是“渗透”。“渗透”一直持续到这个异常被处理或到　　达控制边界。

一个控制边界是一个调用堆栈入口，它紧接着前面不同的活动组的调用堆栈入口或是一个 OPM 程序。图 68 中，过程 P1 是控制边界。



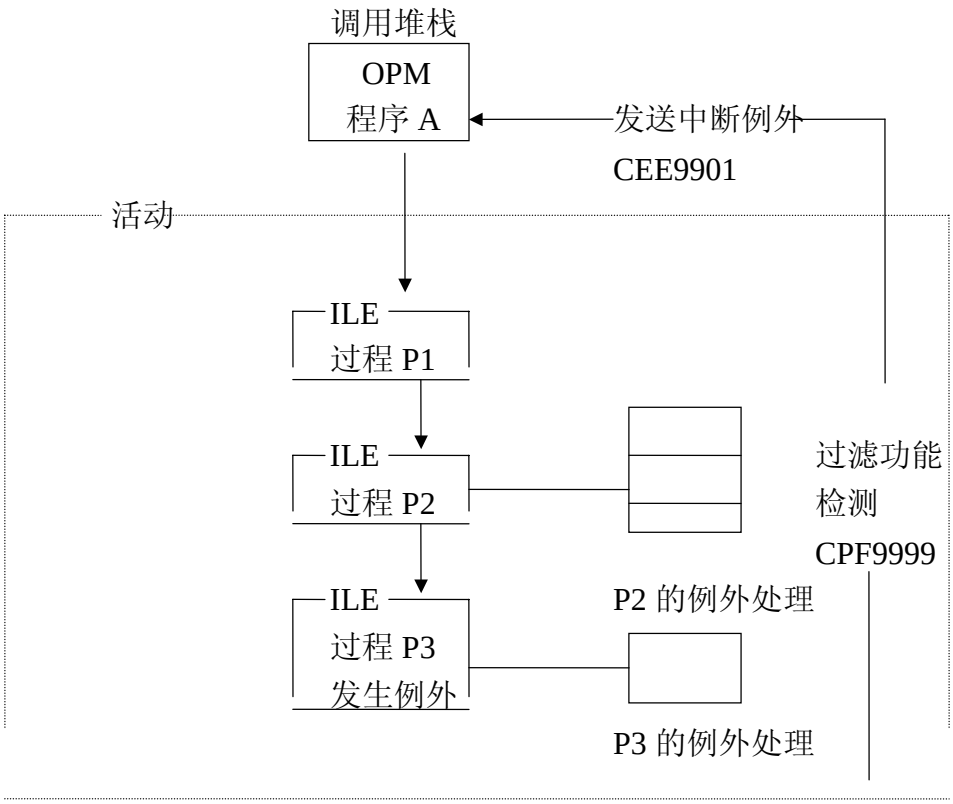


图 68　调用堆栈和异常信息过滤

在 OPM 中，异常信息与调用堆栈中活动的程序相连系。如果一个异常没被相关的异常处理例程所处理。要往接收异常的相同调用堆栈入口发送的一个功能检查。如果它仍旧没被处理，则入口被清除且功能检查被“渗透”，这个过程重复直到异常被处理为止。

在 ILE 中，一个异常信息与调用堆栈中活动的过程相连系。当异常被“渗透”时，不转化成功能检查。每个调用堆栈入口有机会处理原始异常，直到到达控制边界，这时异常才被转化成一个功能检查。那么这类的异常处理，从第一次收到异常的过程开始，每次调用堆栈入口都给一个机会来处理功能检查。如果到达控制边界并且异常仍没被处理，则发送一个一般失败信息 CEE9901 发送给控制边界过程的调用者。此外，取消不处理信息的调用堆栈入口。

11.1.1 ILE RPG/400 异常处理

ILE RPG/400 提供了三种类型的异常处理途径

错误指示器处理

错误子例程处理

缺省异常处理

RPG 把异常分为两类：程序类和文件（类）。这取决于调用什么类型的错误子例程。程序异常的例子象零做除数，数组下标越界，或负数的排序。文件异常的一些例子象未定义的记录类型或设备错误。

有三种方法指示 RPG 处理异常，它们是：

1、在相应操作码的计算规范表中 73－74 列指定一个错误指示器。

2、编一个文件错误子例程。这个子例程在文件描述规范表上由 INFSR 键字定义。

3、编一个程序错误子例程，名叫*PSSR。

无论什么时候出现异常，ILE RPG/400 做如下处理：

1、如果在计算规范设置错误指示器，且异常是那个操作码可检测到的。

a、这个指示器被置“on”

b、异常被处理

c、控制转到下一个 ILE RPG/400 操作

2、如果没有设置错误指示器且编了一个错误子例程并且异常是一个程序异常。或已经　　为文件编了一个 INFSR 错误子例程并且异常是一个 I/O 异常。

那么异常将被处理并且控制将转给错误子例程的第一条语句上。

3、如果没有错误指示器或错误子例程，则 RPG 将执行缺省的错误处理：

如果异常不是一个功能检查，则异常将被“渗透”。

如果异常是一个功能检查，则将显示一个询问信息。

如果选择’G’或’R’，就处理功能检查且控制将转给在过程中相应的点上。（’G’

即’*GETIN’，’R’即做相同的规范规定），否则，功能检查将被渗透且过程被　　非正常结束。

有关 RPG 缺省处理内容，请看 3.2.2.3 中的“未处理的异常”。

11.1.1.1　OPM 和 ILE RPG400 异常处理的不同

大部分情况下，OPM　RPG/400 和 ILE　RPG/400 异常处理行为相同。关键的不同是在未处理异常方面。

在 OPM 中，如果出现异常且不是 PRG 指定的处理，则将发送一个询问信息，在 ILE 中，这种情况只发生在异常是功能检查上。如果异常不是功能检查，则它将被传送到程序或过程的调用者，并且任何适宜的高阶调用堆栈入口都有可能处理这个异常，考虑如下的例子：

PGM A 调用 PGM B，PGM B 调用 PGM C

PGM B 有一个针对 CALL 语句的错误指示器

PGM C 没有错误指示器或*PSSR 错误子例程

PGM C 得到一个异常

在 OPM 中，将发送一个针对 PGM C 的询问信息。在 ILE 中，PGM C 不处理它异常被渗透到 PGM B。PGM B 中的错误指示器被置为“on”让 PGM B 来处理这个错误并且在此过程 PGM C 非正常结束，没有询问信息。

如果 PGM C 有一个*PSSR 错误子例程，则在 OPM 和 ILE 中异常都被 PGM C 处理，且运行错误子例程。

注：由 ILE RPG/400 发送的询问信息前缀为’RNQ’，不象 OPM RPG/400 是’RPG’。

对某些特别的错误，有不同的处理，方法详情请看附录 A“OPM RPG/400 和 ILE RPG/400 间的区别”。

11.2　使用异常处理例程

为应用程序设计异常处理包括以下内容：

1、决定是否使用 RPG 特定的处理方式（即错误指示器或子例程）或者写一个异常　　处理例程，它用 ILE API CEEHDLR 登记，也可二者都选用。

2、确定恢复动作，就是如果使用一个独立的异常处理例程，给出程序接着处理的接口。　　此外，在设计异常处理时，要注意处理的优先级。

嵌套异常

未处理异常的缺省动做

优化的影响

11.2.1 异常处理优先级

如果你既使用语言规定的错误处理又使用 ILE 条件处理，异常处理的优先级就变得很重要。对一个 ILE/400 过程，异常处理有如下优先级。

1、错误指示器处理

2、I/O 错误子例程处理

3、ILE 条件处理

4、程序错误子例程处理

5、RPG 缺省处理（对未处理异常）

11.2.2 嵌套异常

异常可以是嵌套的。一个嵌套异常是处理一个异常时又出现另一个异常。发生这种情况。第一个异常被临时挂起，先处理最后发生的那次异常。

11.2.3 未处理的异常

一个未处理的异常是一个不曾被第一个接收异常的调用堆栈入口所处理的异常。当一个异常不被处理时，ILE RPG/400 缺省处理采取如下动作之一：

如果信息类型是一个功能检查(CPF9999)，则 RPG 缺省处理发送一个询问信息描述这个原始情况。

在回答信息时，如果选择 D(ump)或 C(ancel)，则第一个接收异常信息的过程终止且功能检查“渗透”给调用者。

如果选择 R(etry)或 G(et input)，则功能检查被处理，异常处理结束，过程 *GETIN 点开始执行（当选择 G 时）或从发生异常处理 I/O 操作（当选择 R 时）开始执行。例如，如果由于记录锁定而导致读失败时，任何读操作将被重新获取，那么再做读操作。

对其它类型的信息，异常将被渗透到程序的调用者。这个过程有处理异常功能，并有机会处理它。如果没有，则异常将被渗透给调用堆栈，直到达到控制边界。这时，异常被转化成一个功能检查，且异常处理又一次如上面描述的那样开始。

11.2.3.1 未处理“逃逸”信息的例子

下面描述出现一个 escape 逃逸信息时，产生这个信息的过程不能处理它，这个事件有如下的假设：

1、有两个程序，PGM1 和 PGM2 运行在同一活动组。每一个程序分别包含一个过程 PRC1 和 PRC2。

2、PRC1 动态的调用 PGM2 且 PRC2 接收控制。

3、PRC1 中的 CALL 操作码有一个错误指示器。

4、在 PRC2 中没有 RPG 异常处理编码。也就是对 SUBST 操作没设置错误指示器，也没*PSSR 错误子例程。

5、PRC2 有 SUBST 操作，且因子 1 是一负数。

当 PGM1 调用 PGM2 时，执行 SUBST 操作，生成一个异常信息“RNX0100”。图 69 给出说明及出现的事件。

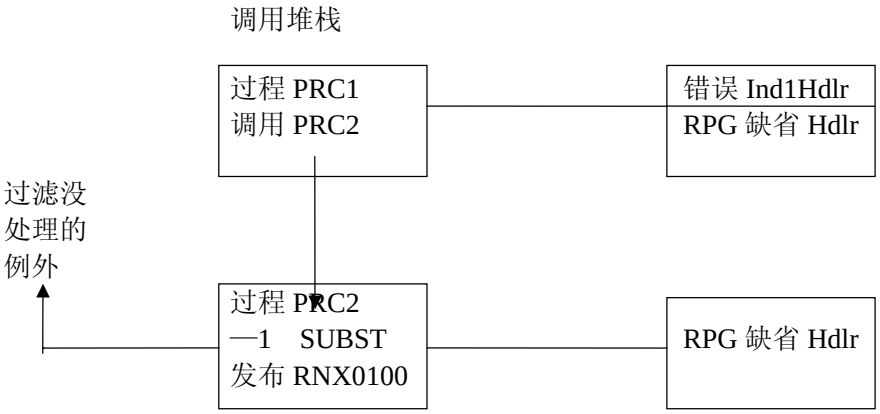


图 69　对未处理逃逸信息的说明

如发生如下情况：

1、在 PRC2 中 SUBST 操作中没有错误指示器或*PSSR 错误子例程，PRC2 就不能处理这个程序错误，所以异常不被处理。

2、因为异常不是一个功能检查，则它被渗透到 PRC1。

3、PRC1 接收（处理）这个异常信息，CALL 操作的错误指示器置为 ON。

4、接着执行 PRC1 中 CALL 操作后边的语句。

注意：同样的异常处理也适用于过程调用（CALLB 操作码）。

11.2.3.2　未处理功能检查的例子

下面说明出现功能检查且不被处理的情况，假设：

1、有两个程序，PGM1 和 PGM2，分别包含过程 PRC1 和 PRC2。

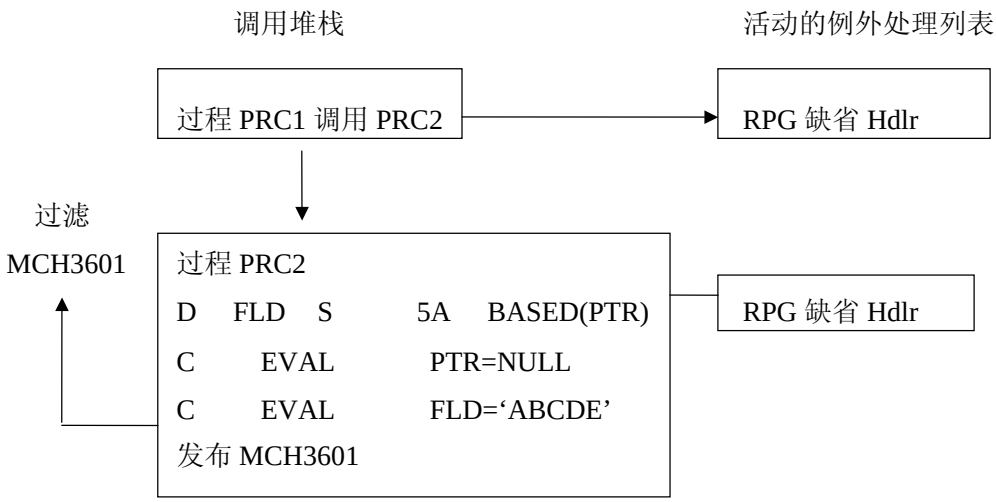
2、PRC1 动态调用 PGM2 则 PRC2 接收控制。

3、PRC1 中的 CALL 操作码没有错误指示器。

4、PRC2 中没有编写 RPG 异常处理，也就是，没有错误指示器也没有*PSSR 错误子例程。

5、 PRC2 有一个指针地址错误。

当 PGM1 调用 PGM2 时，由于基本指针定义为空，出现一个指针错误，这样必然生成信息 MCHB06。当 PRC2 试图渗透异常使其通过控制边界时，出现功能检查。



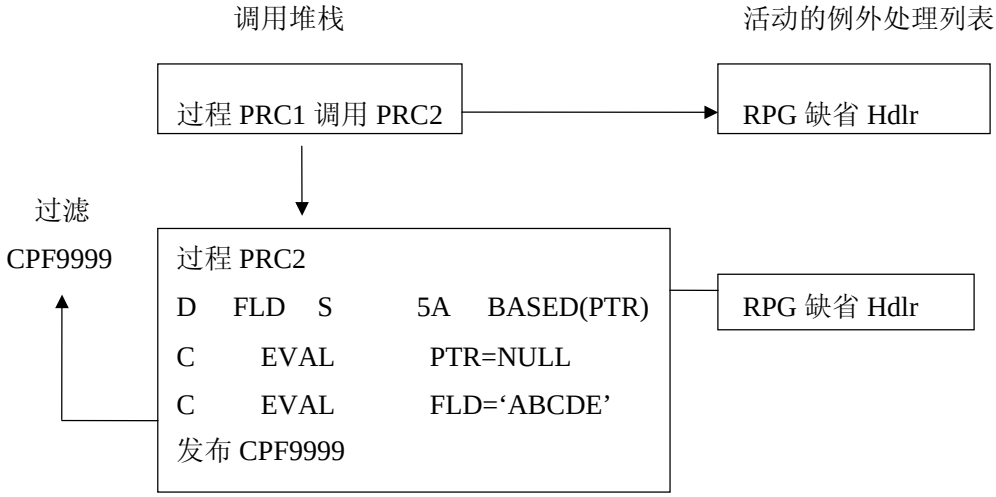


图 70　给出这个说明和出现的事件

对未处理功能检查的说明，将发生下面情况：

1、由于在 PRC2 中无错误指示器，PRC2 不能处理这个功能检查，所以它不被处理。

2、由于它是一个功能检查，则将发送一个询问信息以描述原始状况。

3、根据对询问信息的回答，PRC2 可能会被结束，且异常渗透到 PRC1（选“C”时）或继续在 PRC2 中处理（选“G”时）。

11.2.4 优化考虑

当运行一个“*FULL”优化的程序，在优化的程序处理期间，在予先定义的某点上，优化器可能频繁使用寄存器中的值，并把它们恢复到内存，异常处理会打破这个正常的执行过程，后续程序在寄存器中的变量可能不会返回到分配给它们的存储位置。

特别的，如果通过下列方法恢复异常时可能不会包含当前值：

*PSSR 错误子例程

INFSR 错误子例程

用户定义的异常处理

在询问信息中回答 GO(‘G’)

在询问信息中回答 Retry(‘R’)

即使使用全优化，ILE RPG/400 自动定义指示器以使它们包含当前值，为确保字段或数据结构的内容是当前值，要在相应的定义规范表里指定 NOOPT 键字。

有关 NOOPT 键字的规则，参见《ILE RPG/400 参考》。有关于优化的详细资料，见 2.3.4.2 “修改优化级”。

11.3 使用 RPG－特定处理例程

ILE RPG/400 为你提供三种方法来利用 HLL－指定处理例程使之从异常中恢复。

1、错误指示器

2、INFSR 错误子例程

3、*PSSR 错误子例程

通过编制数据结构、查询其中相关的字段，能获得关于错误的详细信息。

本节提供了“怎样使用各种 RPG 结构的例子。《ILE RPG/400 参考》提供了更多的关于*PSSR 和 INFSR 错误子例程、EXSR 操作码、INFDS 和 PSDS 数据结构的介绍。

11.3.1 规定错误指示器

要使用 RPG 错误指示器，要在下面列出的操作码的 73 和 74 列规定一个错误指示器。

表 10　位置 73－74 允许一个错误指示器的操作码

* 表 10. 在位置 73－74 允许一个错误指示器的操作码				*
* ACQ	* DSPLY	* POST	* SETGT	*
* ADDDUR	* EXFMT	* READ (n)	* SETLL	*
* CALL	* EXTRCT	* READC	* SUBDUR	*
* CALLB	* NEXT	* READE (n)	* SUBST (p)	*
* CHAIN (n)	* FEOD	* READP (n)	* TEST (x)	*
* CHECK	* IN	* READPE (n)	* UNLOCK	*

* CHECKR	* OCCUR	* REL	* UPDATE	*
* CLOSE	* OPEN	* ROLBK	* WRITE	*
* COMMIT	* ORxx	* SCAN	* XLATE (p)	*
* DELETE	* OUT	*	*	*

如果出现异常，指示器设为 ON。相应的数据结构(PSDS 或 INFDS)被更新，且控制返回到下一条指令。这时，可以测试指示器来决定采取什么动作。

当在操作码中规定错误指示器时可用 EXSR 操作。明确的调用一个文件错误子例程(INFSR)或程序错误子例程(*PSSR)。在 EXSR 操作中调用 INFSR 或*PSSR 时，ENDSR 的因子 2 是空白或规定的字段有一空值，则控制转给 EXSR 操作后边的下一条指令。

注：如果在操作上指定了一个错误指示器，但出现的错误与这个操作无关。（例如：CHAIN 操作中的一个数组下标错误），那么这个错误指示器被忽略。这个错误按程序错误处理。

11.3.2 使用一个错误子例程

写一个错误子例程时，要做两件事：

1、可使用 RPG 子例程做错误处理例程。

这个子例程处理异常错误并把控制传递给你的子例程。

2、可选的，可以自己规定一个恢复动作。

根据出现的错误，可使用错误子例程采取特定动作或一个一般动作（例如，对所有错误发送一个询问信息）。

下面的内容适用于错误子例程：

在 EXSR 操作中的因子 2 中指定子例程的名字，即可明确的调用一个错误子例程。

在子例程的 ENDSR 操作的因子 2 上规定一个值，就可以控制处理的返回点。

如果调用了一个错误子例程，那么认为 RPG 已经处理了这个异常，这样，对子例程的调用影响了程序执行的返回点，如果运行子例程时出现一个异常，则要再次调用这个子例程。这个过程将循环下去，除非编一个避免这个问题出现的子例程。

怎样编一个错误子例来避免这样的循环，见 3.2.3.2.3，“在错误子例程中避免循环”。

3.2.3.2.1 使用文件错误(INFSR)子例程

要处理一个文件错误或异常，可写一个文件错误(INFSR)子例程，当出现文件异常时：

1、INFDS 被更新。

2、如果在下列情况下出现，则控制转给 INFSR：

一个隐含的（主要或次要）文件操作。

一个明确的文件操作且在 73－74 列没有指定一个指示器。

注意：一个文件错误子例程可处理多个文件中的错误。

使用 INFSR 时，要遵守：

在程序的开始或结束期间，如果出现文件异常（例如，在周期开始隐含的文件打开），控制传递给 ILE RPG/400 缺省的异常处理，而不是给错误子例程处理。这样，就不执行文件错误子例程。

如果出现的错误与操作不相关。（例如，CHAIN 操作中的数组下标错误，则忽略 INFSR 错误子例程。这个错误类似其他程序错误。

要把文件错误子例程加到程序中，依下列步骤做：

1、在文件描述规范表的 INFSR 后输入子例程的名字。这个名字可以是*PSSR，它指出对异常的控制要转给这个程序错误子例程。

2、在可选的文件描述规范表上使用 INFSR，标识文件信息数据结构。

3、写 BEGSR 操作码因子 1 是与 INFSR 中规定的同一子例程名字。

4、标识一个返回点，写在子例程的 ENDSR 上操作中，因子 2 的有效值，见 3.2.3.3 “在 ENDSR 操作中规定返回点”。

5、编写文件错误子例程的其余部分。尽管任何 ILE RPG/400 编译操作都可用在文件错误子例程中，但不建议对获得错误的同一文件使用 I/O 操作。对文件错误子例程，ENDSR 操作必须放在最后。

3.2.3.2.1 中的图 71 是使用 INFSR 错误子例程进行异常处理的例子。程序 TRNSVPUT 是一个简单的存货清单更新程序，它使用一个交易文件 TRANSACT 来更新主存货清单文件 PRDMAS。如果出现一个 I/O 错误，则调用 INFSR 错误子例程。如果它是一个记录锁错误，则这个记录被写在一个后备日志文件上，否则，发送一个询问信息。

注意：PRDMAS 的文件规范表即标识 INFDS 也标识以与其匹配的 INFSR。

对 TRAHSACT 文件中的每一个记录，将做如下工作：

1、使用“交易产品号”，找出在产品主文件中的相应记录。

2、如果找到这个记录，则修改存货清单的数量。

3、如果在 UPDATE 操作中出现错误，控制传给 INFSR 错误子例程。

4、如果没找到记录，则把产品写到一个错误报表里。

```
*=================================================================*
            * TRNSUPDT: 这是一个存货单修改样本程序，交易文件(TRANSACT) 是顺序 *
            * 处理文件。交易中的产品号用作从主文件(PRDMAS)随机存取的键字。     *
            * 1. 如果找到记录，存货单的数量要做修改。                          *
            * 2. 如果没找到记录，打印错误报告。                                *
            * 3. 如果记录被锁住，把交易写到交易退回日志中等以后处理。          *
```

```
            * 4. 其它错误将产生运行时错误信息。                                    *

*================================================================*


*----------------------------------------------------------------*
            * 定义文件：                                              *
            *   1) PRDMAS      - 产品主文件
*
            *   2) TRANSACT    - 交易文件                              *
            *   3) TRNBACKLG   - 交易返回日志文件                      *
            *   4) PRINT       - 错误报告                              *
*----------------------------------------------------------------*
          FPRDMAS    UF   E           K DISK
          F                                    INFSR(PrdInfsr)
          F                                    INFDS(PrdInfds)
          FTRANSACT  IP   E             DISK
          FTRNBACKLG O    E             DISK
          FPRINT     O    F   80        PRINTER

*----------------------------------------------------------------*
            * 定义文件 PRDMAS 的文件信息数据结构，用*STATUS 字段确定采取      *
            * 的动作。                                                *

*----------------------------------------------------------------*
          D PrdInfds        DS
          D  PrdStatus        *STATUS

*----------------------------------------------------------------*
            * 例外列表
*

*----------------------------------------------------------------*
          D ErrRecLock      C                   CONST(1218)


*----------------------------------------------------------------*
            * 用交易产品号访问产品主文件。                            *
```

```
*----------------------------------------------------------------*
          C     TRNPRDNO      CHAIN     PRDREC                       10

*----------------------------------------------------------------*
           * 如果找到记录，修改主文件中的数量。                        *

*----------------------------------------------------------------*
          C                   IF        NOT *IN10
          C                   SUB       TRNQTY        PRDQTY
          C                   UPDATE    PRDREC

*----------------------------------------------------------------*
           * 如果没找到记录，写错误报告。                              *

*----------------------------------------------------------------*
          C                   ELSE
          C                   EXCEPT    NOTFOUND
          C                   ENDIF
          C                   SETON                                  LR
*----------------------------------------------------------------*
           * 错误处理子例程。                                        *

*----------------------------------------------------------------*
          C     PrdInfsr      BEGSR

*----------------------------------------------------------------*
           * 如果主文件记录当前被锁定，把交易记录写到退回日志文件中，跳到下  *
           * 一个交易。                                              *

*----------------------------------------------------------------*
          C     PrdStatus     DSPLY
          C                   IF        (PrdStatus = ErrRecLock)
          C                   WRITE     TRNBREC
          C                   MOVE      '*GETIN'      ReturnPt         6
*----------------------------------------------------------------*
           * 如要发生其它错误，给出查询信息。                          *
```

```
*-------------------------------------------------------------------*
          C                   ELSE
          C                   MOVE      *BLANK        ReturnPt
          C                   ENDIF
          C                   ENDSR     ReturnPt
*-------------------------------------------------------------------*
           * 错误报告格式。                                          *

*-------------------------------------------------------------------*
          OPRINT     E            NOTFOUND
          O                       TRNPRDNO
          O                                        29 'NOT IN PRDMAS FILE'
```

图 71　文件异常处理例子

当控制传给错误子例程时，发生下列情况：

如果错误是由一个记录锁引起的，则把记录写到后备日志文件中，控制返回给主程序处理下一个交易 。（用*GETIN 做返回点）。

如果错误是由其他原因引起，则空格被传送到 ReturnPt，这导致 RPG 缺省处理接收控制，在那点的恢复动作将取决于错误的性质。

注意：对记录锁错误的检查是看它是否匹配与 PRDMAS 中 INFDS 的*STATUS 子字段而不是由记录锁状态码中的 ErrReclock 字段来确定。INFSR 可以扩充来处理其它类型的 I/O 错误，通过定义、检查，然后采取相应的动作。

11.3.2.2　使用程序错误子例程

处理一个程序错误或异常，可以写一个程序错误子例程（*PSSR）。当程序发生错误时：

1、更新程序状态数据结构。

2、如果操作码的 73 和 74 列上设有定义指示器，错误被处理，控制转到*PSSR。

通过在文件描述规范表中的键字 INFSR 后指定*PSSR，也可以在文件发生错误时，明确的把控制转给一个程序错误子例程。

要在程序中加*PSSR 错误子例程，做如下工作：

1、可选的，在定义规范表的 23 列中指定一个 S，指出程序状态数据结构(PSDS)。

2、输入一个 BEGSR 操作，其因子 1 写*PSSR。

3、规定一个返回点，可以在子例程中 ENDSR 操作中指出它，有关因子 2 的有效内容，见 3.2.3.3，在 ENDSR 操作中指定一个返回点。

4、编写程序错误子例程的其他部分。ILE RPG/400 编译程序的操作都可以用于程序错误子例程。ENDSR 操作必须放在程序错误子例程的最后。

图 72 是程序错误子例程的一个例子。

```
*-------------------------------------------------------------------*
          *  定义程序状态数据结构的有关部分。                          *
*-------------------------------------------------------------------*
          D Psds           SDS
          D  Loc             *ROUTINE
          D  Err             *STATUS
          D  Parms           *PARMS
          D  Name            *PROC
 *-------------------------------------------------------------------*
          *  如果用零做除数会发生错误，控制转给*PSSR 子例程。           *
*-------------------------------------------------------------------*


*===================================================================*
          * *PSSR: 过程的错误子例程，要检查是否用零做除数。由测试状态是
*
          *        否为 102 判定。如果是，除数加 1，把*GETIN 传送到 ReturnPt，*
          *        继续执行。                                        *
*===================================================================*
          C     *PSSR         BEGSR
           C                   IF        Err = 102
          C                   ADD       1             Divisor
          C                   MOVE      '*GETIN'      ReturnPt        6
*-------------------------------------------------------------------*
          *        发生非期望错误，把*CANCL 传送到 ReturnPt，过程结束。  *
*-------------------------------------------------------------------*
          C                   ELSE
          C                   MOVE      '*CANCL'      ReturnPt
          C                   ENDIF
          C                   ENDSR     ReturnPt
```

图 72　*PSSR 子例程的例子

程序状态数据结构是在定义规范表中定义。子字段*STATUS，*ROUTINE，*PARMS和*PROGRAM 是已经定义好的，名字分配给这些子字段。

*PSSR 错误子例程是在计算规范表中。如果出现程序错误，ILE RPG/400 把控制传递给*PSSR 错误子例程。这个子例程做检查以决定是否异常。是否是由零做除数的操作引起的，如果是的话，除数加 1，且文字把’*DETC’传送到字段 ReturnPt 中，来指示程序从详细计算例程的起点开始执行。

如果异常不是由零做除数引起的则’*CANCL’送到 ReturnPt 字段，过程结束。

11.3.2.3 在错误子例程中避免循环

在上一个例子中，不可能因为*PSSR 中发生错误而导致循环。然而，*PSSR 如编的不好，可能在执行*PSSR 时出现循环。

避免循环的一种方法是在子例程中设置一个首次开关。如果不是第一次通过这个子例程，可以在 ENDSR 操作的因子 2 上指定一个适当返回点，比如*CANCL。

图 73 是一个程序 NOLOOP，产生异常，目的是怎样在一个*PSSR 子例程中避免循环，程序两次出现异常：

1、在代码的主体，把控制传递给*PSSR。

2、在*PSSR 内潜在的引起循环。

```
*=====================================================================*
            * NOLOOP: 给出如何在*PSSR 子例程中避免循环。                    *

*=====================================================================*
           FQSYSPRT   O    F  132        PRINTER
 *--------------------------------------------------------------------*
            * 用数组导致发生错误。
*
*---------------------------------------------------------------------*
           D Arr1            S             10A   DIM(5)
 *--------------------------------------------------------------------*
            * 产生一个数组超边界错误，使控制转给*PSSR。                     *

*---------------------------------------------------------------------*
           C                   Z-ADD     -1            Neg1              5 0
           C                   MOVE      Arr1(Neg1)    Arr1(Neg1)
```

```
          C                   MOVE      *ON           *INLR
 *============================================================*
          * *PSSR: 过程用的错误子例程，用变量 Inpssr 来确定在 PSSR 中的循环。 *
          *         如果检查到一个循环用*CANCL 中止过程。
*

*============================================================*
          C     *PSSR         BEGSR
          C                   EXCEPT    In_Pssr
          C                   IF        InPssr = 1
          C                   MOVE      '*CANCL'      ReturnPt          6
          C                   EXCEPT    Cancelling
          C                   Z-ADD     0             InPssr            1 0
          C                   ELSE
          C                   Z-ADD     1             InPssr
*                                                                       *
          *         在 PSSR 产生另外一个错误，看它如何撤消这个过程。          *
*                                                                       *
          C                   MOVE      Arr1(Neg1)    Arr1(Neg1)
*                                                                       *
          *         注意：如果 Neg1 仍然是负数，就不执行下面两个操作。        *
*                                                                       *
          C                   MOVE      '*GETIN'      ReturnPt
          C                   Z-ADD     0             InPssr
          C                   ENDIF
          C                   ENDSR     ReturnPt
*============================================================*
          * 过程输出。                                                  *

*============================================================*
          OQSYSPRT   E            In_Pssr
          O                                          'In PSSR'
          OQSYSPRT   E            Cancelling
          O                                          'Cancelling...'
```

图 73　在一个错误子例程中避免循环

使用图 73 中的源码程序，键入：

CRTBNDRPG　PGM(MYLIB/NOLOOP)

当调用这个程序时，发生如下情况：

1、当程序试图往使用负下标的数组上做 MOVE 操作时，出现异常，控制被传递给*PSSR。

2、既然这是首次通过*PSSR，变量 In－Pssr 不为 ON，要避免下一步循环，把变量 In－Pssr 置为‘on’。

3、过程继续执行*PSSR 中 ELSE 后的 MOVE 语句。两次出现异常，再执行*PSSR。

4、这次通过*PSSR 时，变量 In－Pssr 已为“1”，这指出子例程处于循环中，由于字段 Return 设为*CANCL，这个过程被取消。

5、ENDSR 操作接收控制，则这个过程被取消。

程序产生的输出显示如下：

In　PSSR

In　PSSR

Cancelling・・・・

这里用到的避免循环死锁的方法也能用在 INFSR 错误子例程中。

11.3.3 在 ENDSR 中指定一个返回点

使用 INFSR 或*PSSR 错误子例程时，可以指示程序将要接着执行的返回点，可在 ENDSR 语句的因子 2 上输入如下内容之一。输入必须是六个长的字符字段、文字、命名的常量。数组元素或表名，它们的值指定了返回点。

注：如果返回点指定为文字，则必须放在括号内，且用大写（如，*DETL，不是*detl）。如果返回点指定为字段或数组元素，它必须左对齐。

*DETL	在明细行的开始继续执行
*GETIN	在得到输入记录例程继续执行
*TOTC	在汇总计算的开始继续执行
*TOTL	在汇总行的开始继续执行
*OFL	在溢出行的开始继续执行
*DETC	在明细计算的开始继续执行
*CANCL	取消程序的执行

BLANKS 把控制返回给 ILE RPG/400 缺省异常处理程序。当因子 2 为空值或没定义时，发生这种情况。如果由 EXSR 调用这个子例程且因子 2 为空，控制返回到下一个的指令。

在 INFSR 或*PSSR 子例程中的 ENDSR 运行后，ILE RPG/400 编译程序把因子 2 上指定的字段或数组元素置为空。这样就可以在异常的子例程中设计返回点使它更适合处理发生的异常。

子例程结束时，如果这个字段包含空值，除非由 EXSR 调用 INFSR 或*PSSR 子例程，ILE RPG/400 缺省异常处理接收控制运行子例程下面的语句。如果子例程通过 EXSR 被调用且 ENDSR 操作的因子 2 为空。控制转向紧跟着 EXSR 操作的下一条指令。

11.4 ILE 条件处理

ILE 条件处理例程是异常处理例程，它在运行时由可联接的 API 条件处理例程(CEEHDLR)记录在寄存器中。用它们渗透或升级异常处理。异常用 ILE 条件格式表现为某个条件处理，可以寄存多个 ILE 条件处理。ILE 条件处理也可调用 API 的非寄存 ILE 条件处理(CEEHDLU)来取消。

使用 ILE 条件处理，响应的异常信息来指出它已被处理，如果不是，那么系统就认为信息未被处理。要修改信息，参考《系统 API 参考》中修改异常信息“(QMHCHGEM)API”的内容。

使用一个 ILE 条件处理有几个原因：

不用语言规定处理，而用自己的方式来处理异常。

能在一个应用程序中不同的 ILE HLLS 模块中使用同一异常处理机制。

使用这个 API 把异常处理放到一个调用堆栈入口。

在 ILE RPG/400 中，ILE 连接的 API CEEHDLR 对包含它的过程都起作用。它将一直有效直到注销它或过程返回。

注：从任何明细、汇总或计算子例程对 CEEHDLR API 的调用将使条件处理对整个过程是活动的，包括所有输入、计算和输出操作。

关于怎样使用 ILE 条件处理的详细介绍，参考《ILE 概念》。

11.4.1 使用一个条件处理

下面的例子说明怎样：

1、编写一个条件处理来处理 RPG‘越界’错误。

2、登记一个条件处理。

3、注销一个条件处理。

4、编一个*PSSR 错误子例程。

这个例子由两个过程组成。

RPGHDLR 是用户编写的针对越界子串错误的条件处理。

SHOWERR，测试 RPGHDLR 过程。

SHOWERR 主要用来展示 RPGHDLR 是怎样工作的。两个过程结合在一起，说明怎样确定 ILE 异常处理。两个过程在它们被执行时都对 QSYSPRT 做输出。也可以修改这些过程，设计成你想让 ILE 异常处理完成的工作。

图 74 是过程 RPGHDLR 的源码，这个过程定义了 3 个过程参数：一个是 ILE 条件象征结构，另一个是包含出错过程名的字段，最后一个是包含可能的动作，重开始或“渗透”的字段(RPGHDLR 不引发任何异常)。

RPGHDLR 的基本逻辑如下：

1、检测信息 ID 来查看是否是有越界错误。如果有，则向’QSYSPRT’写’Hardling・・・’，然后设置动作到为“Resume”。如果不是，那么把“Percolating”写到 QSYSPRT，然后，设置动作为“Percolate”。

2、返回。

```
*=================================================================*
            * RPGHDLR: RPG 例外处理过程，这个过程做以下事情：                *
            *           如果 RPG 超出边界错误（RNX0100），就做例外处理。      *
            *           如果不是，过滤这个例外。打印结果。
*
            *                                                         *
            * 注：      这是对 SHOWERR 例外处理的过程。                 *

*=================================================================*
           FQSYSPRT   O    F  132        PRINTER
    *-----------------------------------------------------------------*
            * 过程参数：
*
            * 1.输入：条件标识结构                                     *
            * 2.输入：发生错误的过程名                                 *
            * 3.输出：在例外时完成动作的标识                           *
            * 4.输出：如果决定放宽条件，给出新条件。即然这个处理只是一个假定  *
            *         和过滤，我们忽略这个参数。                       *

*-----------------------------------------------------------------*
           D CondTok         DS
           D  MsgSev                        4B 0
           D  MsgNo                         2A
           D                                1A
           D  MsgPrefix                     3A
           D                                4A
           D ProcName        S             10A
           D Action          S              9B 0
            *
            * Action codes are:
            *
           D Resume          C                   10
           D Percolate       C                   20
```

```
*------------------------------------------------------------------*
           * Parameters                                              *
*------------------------------------------------------------------*
          C     *ENTRY        PLIST
          C                   PARM                    CondTok
          C                   PARM                    ProcName
          C                   PARM                    Action
 C*-----------------------------------------------------------------*
          C*      如果有子串错，就处理过滤。注意信息号码是用 16 进制表示。     *
C*-----------------------------------------------------------------*
          C                   EXCEPT
          C                   IF        MsgPrefix = 'RNX' AND
          C                             MsgNo     = X'0100'
          C                   EXCEPT    Handling
          C                   EVAL      Action    = Resume
          C                   ELSE
          C                   EXCEPT    Perclating
          C                   EVAL      Action    = Percolate
          C                   ENDIF
          C                   RETURN
*==================================================================*
           * Procedure Output                                        *
*==================================================================*
          OQSYSPRT   E
          O                                         'In Handler for    '
          O                    ProcName
          OQSYSPRT   E         Handling
          O                                         ' Handling...'
          OQSYSPRT   E         Perclating
          O                                         ' Percolating..    .'
```

图 74　越界子串错误条件处理的源码

图 75 是过程 SHOWERR 的源码，在这个过程中，条件处理例程 RPGHDLR 被登记。

这个过程的参数包括一个指向 RPGHDLR 的过程指针和一个指向程序状态数据结构的指针。此外，它需要定义一个发生错误的数组 ARR1，以及一个 ILE 可连接 API 的 CEEADLR 和 CEEHDLU 使用的参数列表。

这个程序的基本的逻辑如下：

1、使用 Reghndlr 子例程登记 RPGHDLR。

这个子例程调用 CEEHDLR API，把它的过程指针传给 RPGHDLR。

2、产生一个越界子串错误。

RPGHDLR 被自动调用。它处理异常且指出它应返回到错误后的下一个机器指令。注意下一个机器指令也许不是下一个 RPG 操作。

3、产生一个越界数组错误。

再次自动调用 RPGHDLR，然而这次不处理异常，它把异常过滤给与这个过程有关的下一个异常处理，即*PSSR 错误子例程。*PSSR 取消了这个过程。

4、通过调用 CEEHDLU 注销条件处理 RPGDHDLR。

5、返回。

用 RPGHDLR 过程，SHOWERR 写给 QSYSPRT 来通知执行时发生的情况。

```
*=================================================================*
          * SHOWERR: 给出使用用户定义的例外处理程序来处理例外条件。          *

*=================================================================*
          FQSYSPRT   O    F  132        PRINTER

*-----------------------------------------------------------------*
          * 下面是为 CEEHDLR API 定义的参数。第一个是过程指针，指向要处理    *
          * 的例外。第二个是通讯区指针，把它传给例外处理程序。这个例子，
*
          * 我们把指针传给过程名。                                          *
*-----------------------------------------------------------------*
          D pConHdlr        S               *   PROCPTR
          D                                     INZ(%paddr('RPGHDLR'))
*-----------------------------------------------------------------*
          * PSDS                                                          *
*-----------------------------------------------------------------*
          D DSPsds          SDS                 NOOPT
          D   ProcName        *PROC
 *-----------------------------------------------------------------*
```

```
          * 用来引起错误的数组                                            *

*------------------------------------------------------------------*
         D Arr1           S              10A   DIM(5)
 *------------------------------------------------------------------*
          * CEEHDLR 接口                                                  *

*------------------------------------------------------------------*
         C     CEEHDLR_PL    PLIST
         C                   PARM                    pConHdlr
         C                   PARM                    ProcName
         C                   PARM                    *OMIT
 *------------------------------------------------------------------*
          * CEEHDLU 接口                                                  *

*------------------------------------------------------------------*
         C     CEEHDLU_PL    PLIST
         C                   PARM                    pConHdlr
         C                   PARM                    *OMIT
*------------------------------------------------------------------*
          * 寄存处理程序且生成一个错误                                    *

*------------------------------------------------------------------*
         C                   EXSR      RegHndlr
C*------------------------------------------------------------------*
         C*     生成子串错                                                *

C*------------------------------------------------------------------*
         C                   Z-ADD     -1            Neg1              5 0
         C     Neg1          SUBST     'Hello'       Examp            10
*------------------------------------------------------------------*
          *     由处理程序处理例外，控制到这里。
*

*------------------------------------------------------------------*
         C                   EXCEPT    ImBack
*------------------------------------------------------------------*
          *     生成超边界错                                              *
```

```
*-------------------------------------------------------------------*
          C                   MOVE      Arr1(Neg1)     Arr1(Neg1)
 C*------------------------------------------------------------------*
          C*      例外由处理程序处理。这样，控制没返回到这，过滤例外，      *
          C*      控制转给*PSSR。                                    *
 C*------------------------------------------------------------------*

*-------------------------------------------------------------------*
           *      寄存处理程序。                                     *
           *      注意，如果在寄存处理程序前发生例外，那么再寄存。     *
           *      当撤消过程时，它自动的寄存起来。                   *
*-------------------------------------------------------------------*
          C                   EXSR      DeRegHndlr
          C
          C                   SETON                                LR
 *==================================================================*
           * RegHdlr - 调用 API，寄存调用程序                        *

*==================================================================*
          C     RegHndlr      BEGSR
          C                   CALLB     'CEEHDLR'      CEEHDLR_PL
          C                   ENDSR
*==================================================================*
           * DeRegHndlr - 调用 API，不寄存调用程序                   *

*==================================================================*
          C     DeRegHndlr    BEGSR
          C                   CALLB     'CEEHDLU'      CEEHDLU_PL
          C                   ENDSR

*==================================================================*
           * *PSSR: 过程的错误例程                                   *

*==================================================================*
          C     *PSSR         BEGSR
          C                   EXCEPT    InPssr
          C                   EXCEPT    Cancelling
```

```
            C                   ENDSR     '*CANCL'
*================================================================*
             * 过程输出
*

*================================================================*
            OQSYSPRT    E           ImBack
            O                                       'I''m Back'
            OQSYSPRT    E           InPssr
            O                                       'In PSSR'
            OQSYSPRT    E           Cancelling
            O                                       'Cancelling...'
```

图 75　登记一个条件处理例程的源码

如果要试一下这些过程，按如下步骤执行：

1、用图 74 中的源码建立过程 RPGHDLR，键入：

CRTRPGMOD　MODULE(MYLIB/RPGHDLR)

2、用图 75 中的源程序，建立过程 SHOWERR，键入：

CRTRPGMOD　MODULE(MYLIB/SHOWERR)

3、建立程序 ERRORTEST，键入：

CRTPGM　PGM(MYLIB/ERRORTEST)　MODULE(SHOWERR RPGHDLR)

4、运行程序 ERRORTEST，键入：

OVRPRTF　FILE(QSYSPRT)　SHARE(*YES)

CALL　PGM(MYLIB/ERRORTEST)

输出如下：

In Handler for SHOWERR

Handling

I’m Back

In Handler for SHOWERR

Percolating

In PSSR

Cancelling

11.5 使用取消处理

取消处理提供了一个重要的功能：当调用堆栈入口由于非正常返回结束时，允许得到控制来做清除和恢复动作。例如：你想通过 SYSREQ 的第二项选择来控制过程的结束或在询问信息回答‘C’也能结束过程。

一个取消处理对过程中每部分代码都起作用，只有当调用堆栈入口的挂起点在那部分代

码之中时，它只对其一部分代码起作用，仅它才得到控制且调用堆栈入口被取消。

登记调用堆栈入口终端用户退出过程(CEERTX)和注销调用堆栈入口终端用户退出过程(CEETUTX)的 ILE 连接 API 提供了一种方法：当它登记的调用推栈入口被取消时，动态地登记一个用户定义的例程，《系统 API 参考》中有关于 ILE 可连接 API 的介绍。

十二章　获得转储

本章描述怎样获得 ILE RPG/400 的格式转储，并且提供了一个格式转储样本。

12.1 获得 ILE RPG/400 格式转储

要得到一个正在运行过程的 ILE RPG/400 格式转储（内存的打印输出），可以：

在计算规范表里编写一个或多个 DUMP 操作码。

用 D 或 F 回答运行时信息，也能自动产生一个转储，参见《CL 程序设计》中“系 统回答列表”。

格式转储包含字段内容，数据结构内容、数组和表内容、文件信息数据结构，过程的程序状态数据结构、转储写到各自的 QPPGMDMP 的文件中（系统非正常转储写到文件 QPSRVDMP 中）。

如果用 F 回答了 ILE RPG/400 运行时信息，转储也包括打开数据路径的十六进制格式（ODP，一个数据管理控制块）。

如果程序中有多个过程，则格式转储中的内容，反映的是在转储请求时，活动的过程信息。

注：要获取变量数据的转储，程序目标必须有调试数据，就是它必须除了用*NONE 生成调试视图。

12.2 使用 DUMP 操作码

在源程序里，编写一个或多个 DUMP 操作码以获取一个 ILE RPG/400 的格式转储，在规范表中 DUMP 操作可被编写在任何一点或几个点上。无论何时出现 DUMP 操作，都写出输出的记录。

在使用 DUMP 操作前，应知道它的如下特性：

只有在控制规范表中的规定键字 DEBUG(*YES)”DUMP”操作才可执行，如果没有这个键字或规定 DEBUG(*NO)，DUMP 操作只找到错误且把这条语句打印在列表中，但不做 DUMP 操作。

如果 DUMP 操作是有条件的，只有当条件满足时才做转储。

如果一个 DUMP 被 GOTO 操作略过，则不做 DUMP 操作。

关于“DUMP”的详细内容请见《ILE RPG/400 参考》

12.3 一个格式转储的例子

下图是 DBGEX 过程的格式转储的例子，(DBGEX 是第 10 章“调试程序”3.1 中程序 DEBUGEX 中两个过程之一)。为了说明格式转储中怎样处理数据缓冲区，因此，增加一个输出文件“QSYSPRT”。

这个例子是一个全格式转储，也就是对一个询问信息回答为“F”时产生的转储。

程序状态信息：

```
Program Status Area:
Procedure Name . . . . . :   DBGEX2      澳哪哪*
Program Name . . . . . :   TEST            *  A
   Library . . . . . :   MYLIB           *
Module Name. . . . . :   DBGEX2      澳哪哪*
Program Status . . . . . :   00202        B
Previous Status  . . . . . :   00000        C
Statement in Error . . . . . :   00000088     D
RPG Routine  . . . . . :   RPGPGM       E
Number of Parameters . . . . . :
Message Type . . . . . :   MCH          F
Additional Message Info  . . . . . :   4431
Message Data . . . . . :
             Program signature violation.
Status that caused RNX9001 . . . . . :   0000        澳哪哪*
Last File Used . . . . . :                  *
Last File Status . . . . . :                  * G
Last File Operation  . . . . . :                  *
Last File Routine  . . . . . :                  *
Last File Statement  . . . . . :                  *
Last File Record Name  . . . . . :            澳哪哪*
Job Name . . . . . :   QPADEV0010  澳哪哪*
User Name  . . . . . :   MYUSERID          *
Job Number . . . . . :   002273            *
Date Entered System  . . . . . :   09/30/1994       *
Date Started . . . . . :   *N/A*            * H
Time Started . . . . . :   *N/A*            *
Compile Date . . . . . :   093094          *
```

```
  Compile Time . . . . . :   153438           *
  Compiler Level . . . . . :   0001           *
  Source File  . . . . . :   QRPGLESRC        *
    Library . . . . . :   MYLIB           *
  Member . . . . . :   DBGEX2      澳哪哪*
```

图 76　格式转储的程序状态信息部分

A、过程标识：过程名、程序名和库名，及模块名。

B、当前状态代码

C、先前状态代码

D、出错的 ILE RPG/400 源语句

E、出现异常或错误的 ILE RPG/400 例程

F、与机器异常有关的 CPF 或 MCH

G、在一个异常或错误出现前所用最后一个文件信息，本例中，没有使用文件。

H、程序信息，‘*N/A*’指出程序中对这个字段设有可用信息，这些字段如果包括在 PSDS 中时才更新。

反馈区：

```
  INFDS FILE FEEDBACK   I
  File . . . . . :   QSYSPRT
  File Open  . . . . . :   YES
  File at EOF  . . . . . :   NO
  File Status  . . . . . :   00000
  File Operation . . . . . :   OPEN I
  File Routine . . . . . :   *INIT
  Statement Number . . . . . :   *INIT
  Record Name  . . . . . :
  Message Identifier . . . . . :

 OPEN FEEDBACK   J
  ODP type . . . . . :   SP
  File Name  . . . . . :   QSYSPRT
    Library . . . . . :   QSYS
  Member . . . . . :   Q501383525                .
  Spool File . . . . . :   Q04079N002
    Library . . . . . :   QSPL
```

```
Spool File Number  . . . . . :   7
Primary Record Length  . . . . . :   80
Input Block Length . . . . . :   0
Output Block Length  . . . . . :   80
Device Class . . . . . :   PRINTER
Lines per Page . . . . . :   66
Columns per Line . . . . . :   132
Allow Duplicate Keys . . . . . :   *N/A*   K
Records to Transfer  . . . . . :   1
Overflow Line  . . . . . :   60
Block Record Increment . . . . . :   0
File Sharing Allowed . . . . . :   NO
Device File Created with DDS . . . . . :   NO
IGC or graphic capable file. . . . . :   NO
File Open Count. . . . . :   1
Separate Indicator Area. . . . . :   NO
User Buffers . . . . . :   NO
Open Identifier. . . . . :   Q04079N002
Maximum Record Length. . . . . :   0
ODP Scoped to Job. . . . . :   NO
Maximum Program Devices. . . . . :   1
Current Program Device Defined . . . . :   1
Device Name  . . . . . :   *N
Device Description Name. . . . . :   *N
Device Class . . . . . :   '02'X
Device Type. . . . . :   '08'X

COMMON I/O FEEDBACK   L
Number of Puts . . . . . :   0
Number of Gets . . . . . :   0
Number of Put/Gets . . . . . :   0
Number of other I/O  . . . . . :   0
Current Operation  . . . . . :   '00'X
Record Format  . . . . . :
Device Class and Type. . . . . :   '0208'X
Device Name  . . . . . :   *N
Length of Last Record  . . . . . :   80
Number of Records Retrieved. . . . . :   80
```

```
        Last I/O Record Length . . . . . :   0
        Current Block Count. . . . . :   0

        PRINTER FEEDBACK:
        Current Line Number. . . . . :   1
        Current Page . . . . . :   1
        Major Return Code. . . . . :   00
        Minor Return Code. . . . . :   00

        Output Buffer:
           0000   00000000  00000000  00000000  00000000  00000000  00000000
00000000  00000000      *                                   *
           0020   00000000  00000000  00000000  00000000  00000000  00000000
00000000  00000000      *                                   *
           0040   00000000  00000000  00000000  00000000
*                              *
```

图 77　格式转储的反馈区部分

I、这是 INFDS 的文件反馈区部分，仅在文件类型被打印时，这些字段才可用。由于它们只有在程序中说明时才被更新，所以 INFDS 反馈部分的其它内容不做转储。

K、这是文件打开反馈信息，关于这些字段的说明，见《数据管理》一书的有关内容。

L、这是文件的一般 I/O 反馈信息，关于这些字段的说明，见《数据管理》一书的有关内容。

```
        Open Data Path:
           0000   64800000  00001AF0  00001B00  000000B0  00000140  000001C6
00000280  000002C0      *       0               F         *
           0020   00000530  00000000  00000000  00000380  00000000  06000000
00000000  00000000      *                                   *
           0040   00008000  00000000  003AC02B  A00119FF  000006C0  00003033
00000000  00000000      *                                   *
           0060   80000000  00000000  003AC005  CF001CB0  00000000  00000000
00000000  00000000      *                                   *
           0080   80000000  00000000  003AA024  D0060120  01900000  00010000
00000050  00000000      *                          &      *
```

```
          00A0   1F000000  00000000  00000000  00000000  E2D7D8E2  E8E2D7D9
E3404040  D8E2E8E2       *                SPQSYSPRT   QSYS*
          00C0   40404040  4040D8F0  F4F0F7F9  D5F0F0F2
*       Q04079N002QSPL          &  *
       Open Feedback:
          0000   E2D7D8E2  E8E2D7D9  E3404040  D8E2E8E2  40404040  4040D8F0
F4F0F7F9  D5F0F0F2       *SPQSYSPRT   QSYS       Q04079N002*
          0020   D8E2D7D3  40404040  40400007  00500000  D8F5F0F1  F3F8F3F5
F2F50000  00000000       *QSPL          &  Q501383525       *
          0040   00500002  00000000  42008400  00000000  0000D5A4  00100000
00000008  00000000       * &         d       Nu             *
          0060   00000000  00000000  00000100  3C000000  0005E000  5CD54040
40404040  40400001       *                    *N           *
          0080   00000000  00001300  00000000  00000000  00010001  5CD54040
40404040  40400000       *                    *N           *
          00A0   07100000  00000000  00450045  00450045  07A10045  00450045
00700045  00450045       *                                 *
          00C0   00450045  00450045  002F0030  00040005  5CD54040  40404040
40400208  00000000       *                *N               *
          00E0   20000000  00000000  00000000  00000000  00000000  00000001
C2200000  00059A00       *                        B        *
          0100   00000000  00000000  00000000  00000000  00000000  4040
*                                *
       Common I/O Feedback:
          0000   00900000  00000000  00000000  00000000  00000000  00000000
00000000  00000208       *                                 *
          0020   5CD54040  40404040  40400000  00500000  00000000  00000000
00000000  00000000       **N            &                 *
          0040   00000000  00000000  00000000  00000000  00000000  00000000
00000000  00000000       *                                 *
          0060   00000000  00000000  00000000  00000000  00000000  00000000
00000000  00000000       *                                 *
          0080   00000000  00000000  00000000  00000000
*                                *
       I/O Feedback for Device:
          0000   00010000  00010000  00000000  00000000  00000000  00000000
00000000  00000000       *                                 *
          0020   0000F0F0  0001
```

```
*  0000                                *
```

图 78　提供格式转储的信息

如果用一个‘F’回答 ILE RPG/400 询问信息，有关这个文件的一般打开数据路径和反馈区也包括在转储中。

数据信息：

```
        ILE RPG/400 FORMATTED DUMP
        Module Name. . . . . . . . . . . . . . :   DBGEX
        Optimization Level . . . . . . . . . . :   *NONE          M

         N
         Halt Indicators:
H1 '0'   H2 '0'   H3 '0'   H4 '0'   H5 '0'   H6 '0'   H7 '0'   H8 '0'   H9 '0'
         Command/Function Key Indicators:
KA '0'   KB '0'   KC '0'   KD '0'   KE '0'   KF '0'   KG '0'   KH '0'   KI '0'   KJ '0'
KK '0'   KL '0'   KM '0'   KN '0'   KP '0'   KQ '0'   KR '0'   KS '0'   KT '0'   KU '0'
KV '0'   KW '0'   KX '0'   KY '0'
Control Level Indicators:
L1 '0'   L2 '0'   L3 '0'   L4 '0'   L5 '0'   L6 '0'   L7 '0'   L8 '0'   L9 '0'
         Overflow Indicators:
OA '0'   OB '0'   OC '0'   OD '0'   OE '0'   OF '0'   OG '0'   OV '0'
         External Indicators:
U1 '0'   U2 '0'   U3 '0'   U4 '0'   U5 '0'   U6 '0'   U7 '0'   U8 '0'
         General Indicators:
01 '0'   02 '1'   03 '0'   04 '1'   05 '0'   06 '1'   07 '0'   08 '0'   09 '0'   10 '0'
11 '0'   12 '0'   13 '0'   14 '0'   15 '0'   16 '0'   17 '0'   18 '0'   19 '0'   20 '0'
21 '0'   22 '0'   23 '0'   24 '0'   25 '0'   26 '0'   27 '0'   28 '0'   29 '0'   30 '0'
31 '0'   32 '0'   33 '0'   34 '0'   35 '0'   36 '0'   37 '0'   38 '0'   39 '0'   40 '0'
41 '0'   42 '0'   43 '0'   44 '0'   45 '0'   46 '0'   47 '0'   48 '0'   49 '0'   50 '0'
51 '0'   52 '0'   53 '0'   54 '0'   55 '0'   56 '0'   57 '0'   58 '0'   59 '0'   60 '0'
61 '0'   62 '0'   63 '0'   64 '0'   65 '0'   66 '0'   67 '0'   68 '0'   69 '0'   70 '0'
71 '0'   72 '0'   73 '0'   74 '0'   75 '0'   76 '0'   77 '0'   78 '0'   79 '0'   80 '0'
81 '0'   82 '0'   83 '0'   84 '0'   85 '0'   86 '0'   87 '0'   88 '0'   89 '0'   90 '0'
91 '0'   92 '0'   93 '0'   94 '0'   95 '0'   96 '0'   97 '0'   98 '0'   99 '0'
         Internal Indicators:
LR '0'   MR '0'   RT '0'   1P '0'
```

```
 0
NAME                 ATTRIBUTES          VALUE
_QRNU_DSI_DS1        BIN(9,0)            1           P
_QRNU_DSI_DS2        BIN(9,0)            2
_QRNU_TABI_TABLEA    BIN(9,0)            1
ARRY                 ZONED(3,2)          DIM(2)
                       (1-2)             1.23
BASEONNULL           CHAR(10)            NOT ADDRESSABLE
BASEPTR              POINTER             SPP:C01947001218
BASESTRING           CHAR(6)             'ABCDEF'
BIGDATE              DATE(10)            '1994-09-30'
BIGTIME              TIME(8)             '12.00.00'
BIGTSTAMP            TIMESTAMP(26)       '9999-12-31-12.00.00.000000'
BIN4D3               BIN(4,3)            -4.321
BIN9D7               BIN(9,7)            98.7654321
DBCSSTRING           GRAPHIC(3)          ' BBCCDD '

DS1                  DS                  OCCURS(3)   Q
  OCCURRENCE(1)
    FLD1             CHAR(5)             '1BCDE'
    FLD1A            CHAR(1)             DIM(5)
                       (1)               '1'
                       (2)               'B'
                       (3)               'C'
                       (4)               'D'
                       (5)               'E'
    FLD2             BIN(5,2)            123.45
  OCCURRENCE(2)
    FLD1             CHAR(5)             '1BCDE'
    FLD1A            CHAR(1)             DIM(5)
                       (1)               '1'
                       (2)               'B'
                       (3)               'C'
                       (4)               'D'
                       (5)               'E'
    FLD2             BIN(5,2)            123.45
  OCCURRENCE(3)
    FLD1             CHAR(5)             '1BCDE'
```

```
    FLD1A           CHAR(1)             DIM(5)
                      (1)               '1'
                      (2)               'B'
                      (3)               'C'
                      (4)               'D'
                      (5)               'E'
    FLD2            BIN(5,2)            123.45
DS2                 CHAR(10)            DIM(2)     R
                      (1)               'aaaaaaaaaa'
                      (2)               'bbbbbbbbbb'
DS3                 DS                            S
  FIRSTNAME         CHAR(10)            'Fred      '
                    VALUE IN HEX        'C69985844040404040'X
  LASTNAME          CHAR(10)            'Jones     '
                    VALUE IN HEX        'D1969585A24040404040'X
  TITLE             CHAR(5)             'Mr.  '
                    VALUE IN HEX        'D4994B4040'X
EXPORTFLD           CHAR(6)             'export'
NULLPTR             POINTER             SYP:*NULL
PACKED1D0           PACKED(5,2)         -093.40
PARM1               PACKED(4,3)         6.666
PROCPTR             POINTER             PRP:A00CA02EC200   T
SPCPTR              POINTER             SPP:A026FA0100C0
SPCSIZ              BIN(9,0)            000000008.
STRING              CHAR(6)             'ABCDEF'
TABLEA              CHAR(3)             DIM(3)
                      (1)               'aaa'
                      (2)               'bbb'
                      (3)               'ccc'
ZONEDD3D2           ZONED(3,2)          -3.21
```

图 79　格式转储的数据节

M　优化级

N　一般指示器 1--99 和它们的当前值（1 为 ON，0 为 OFF），指示器*IN02，*IN04 和*IN06 没设置

O　用户变量的开始，用字母数字顺序排列

P　内部定义字段，它包括表索引和多次出现的数据结构

Q　多次出现的数组结构

R　无子字段的数据结构做为字符串显示

S　用字母数字顺序列出的数据结构子字段，不用它们定义的顺序列出，也没有给出子字段定义的间隔

T　基本和过程指针之间属性没有不同点

第三部分

处理文件和设备

这一部分说明了如何在 ILE 云 RPG/400 程序中使用文件和设备，特别是怎样去：

- 表示文件和设备间的关系
- 定义文件（程序描述和外部描述）
- 处理文件
- 访问数据库文件
- 访问外部连接的设备
- 编写交互应用程序

注：RPGⅣ Program 指的是包含一个或多个用 RPGⅣ 写的过程的 ILE 程序。

第十三章　定义文件

文件在程序和 I/O 设备之间进行连接，系统中每个文件都有一个描述，它给出文件属性及文件有关数据。如何组成记录和字段。

执行某些 I/O 操作的程序，必须给出程序引用的文件描述，所用的 I/O 设备类型及如何组织数据。

这一章给出下面的内容：

- 与 I/O 设备有关的文件描述
- 定义外部描述文件
- 定义程序描述文件
- 数据管理操作

随后的章节介绍了在不同类型的设备上怎样使用外部描述和程序描述文件。

13.1　与 I/O 设备有关的文件

AS/400 上所有输入/输出操作的关键元素是文件，系统支持以下文件类型：

数据库文件	允许在系统上永久存储的数据
设备文件	允许访问外部连接设备，包括显示文件，打印文件，磁带文件，磁盘文件和 ICF 文件。
备份文件	在磁盘上存储备份的数据
DDM 文件	允许访问存储在远程系统上的数据文件。

每个 I/O 设备都有一个相应的上面给出的文件描述，程序用它来访问设备。

文件被处理的同时建立设备联系：当处理文件时才从设备上读或写数据。

为了告诉操作系统程序所使用哪个文件描述，在文件描述规范表的 7－16 列指定文件名，在 36－42 列指出 RPG 设备名，设备名指出 RPG 操作使用这个设备的相关文件。设备名可以是：DISK，PRINTER，WORKSTN，SEQ 或者 SPECIAL。

```
     *..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+...
*

FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords++++++++++++++++++++++++++++++
FFILEX     CF   E             WORKSTN
```

图 80　为显示文件 FILEX 的文件描述规范

在 36－42 列指出的不是设备名，是文件名,它向 OS/400 指出实际设备的文件描述。

下面给出与 RPG 各类设备相关的文件类型：

表 11. 与 RPG 各类设备相关的 AS/400 文件类型

RPG 设备类型	AS/400 文件类型
DISK	数据库文件、备份文件、DDM 文件
PRINTER	打印文件
WORKSTN	显示文件、ICF 文件
SEQ	磁带、软盘、备份文件、打印文件、数据库文件
SPECIAL	N/A

在编译时，特定的 RPG 设备名只对某些 RPG 操作是有效的。这时，RPG 操作是依赖设备的，如 EXFMT 操作码只对 WORKSTN 设备有效。

其他操作码是不依赖设备的，意思是它们能用于任意设备类型。

图 81　解释了图 80 编码的 RPG 文件 FILEX，它对系统文件描述是一个显示文件。

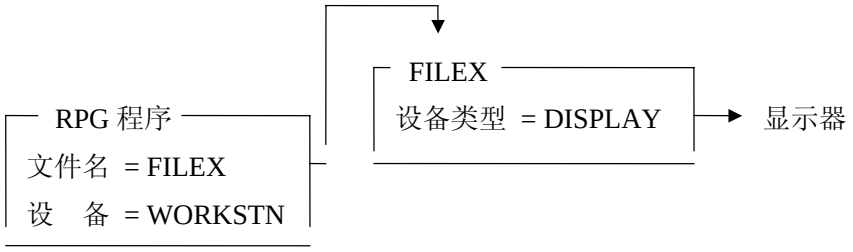


图 81　与显示文件描述有关的文件名

SEQ 设备是非依赖设备类型。图 82 解释了名为 FILEY 的 RPG 文件与一个顺序设备的系统文件描述有关。当程序运行时，实际 I/O 设备在 FILEY 文件描述中指定。例如，设备可以是打印机。

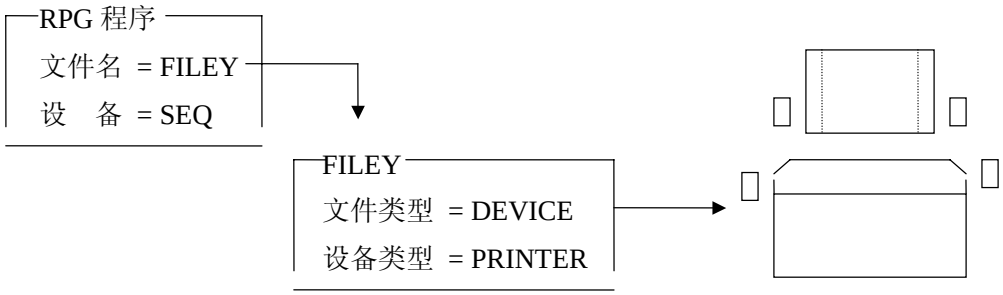


图 82　与显示文件描述有关的文件名

尽管文件名和文件类型在 RPG 程序中指定，但在许多情况下可以修改程序中所用的文件类型，而不用修改程序，具体做法请看 4.2.1 中“复盖和改变文件输入输出”。

13.2 文件命名

AS/400 中，文件是由成员组成，这些文件放在库中。文件名的常规形式是库名/文件名。

在 ILE　RPG/400 程序中，文件名在文件描述规范表的 7－16 列给出，文件名最长可为 10 位字符但必须是唯一的，不能在程序中用带库名的文件名，运行时，系统在与你作业相关的库列表中检索文件。如果你想修改名字，成员或指定某个库，可以使用文件复盖命令。详情请看 4.2.1 的复盖和改变文件的输入输出。

13.3 文件描述类型

说明程序用的文件描述时，必须指出是程序描述文件还是外部描述文件。

程序描述文件的字段级描述，在 RPG 源成员内，用输入/输出规范表进行，对操作系统所做的文件描述包括数据来源及文件中记录的长度。

对于外部描述文件，编译程序从用 DDS，IDDU 或 SQL/400 命令建立的外部文件描述中提取字段描述，因此，不必在 RPG 源成员中的输入/输出规范表中有相应的编码。

外部描述包括数据来源，象数据库或具体设备，还包括每个字段以及它的属性的描述。在编译程序之前文件必须存在并能在库列表中访问到。

外部描述文件具有下列优点：

• 减少程序内的编码数量，如果多个程序使用同一文件，字段可以在 OS/400 上定义一次而为所有的程序使用，这就为使用外部描述文件的 RPG 程序节省了必需的输入/输出规范表的编写。

• 文件的记录格式修改时减少了维护所做的工作。可以通过改变文件记录格式，然后重新编译程序的方法来经常地更新程序，而不用修改程序内任意一条已经编写的语句 。

• 由于使用同一文件的程序用的记录格式与字段一致，故改善了文本的编制。

• 提高了可靠性。如果规定了级别检测，RPG 程序会注意用户是否修改了外部描述。有关更多的资料见 4.14.8 节“级别检查”。

如果对设备 SEQ 或 SPECIAL 指定了一个外部描述文件（文件描述规范表的 22 列用 E 标识），RPG 程序使用文件的字段描述，但是，与 OS/400 的接口就好象文件是程序描述文件一样，外部描述文件不能指定设备相关的功能，象 PRINTER 文件的控制，它已经在外部描述中定义好了。

13.3.1 用外部描述文件作为程序描述文件

外部描述建立的文件可以象程序描述文件一样在程序中使用。如果这样使用，要：

1、在程序中文件描述规范表的 22 列标识 F；

2、在程序的输入/输出规范表的记录中描述字段。

编译时，编译程序使用输入/输出规范表中的字段描述。不检索外部描述。

13.3.2 程序和文件之间的某些有典型关系的例子

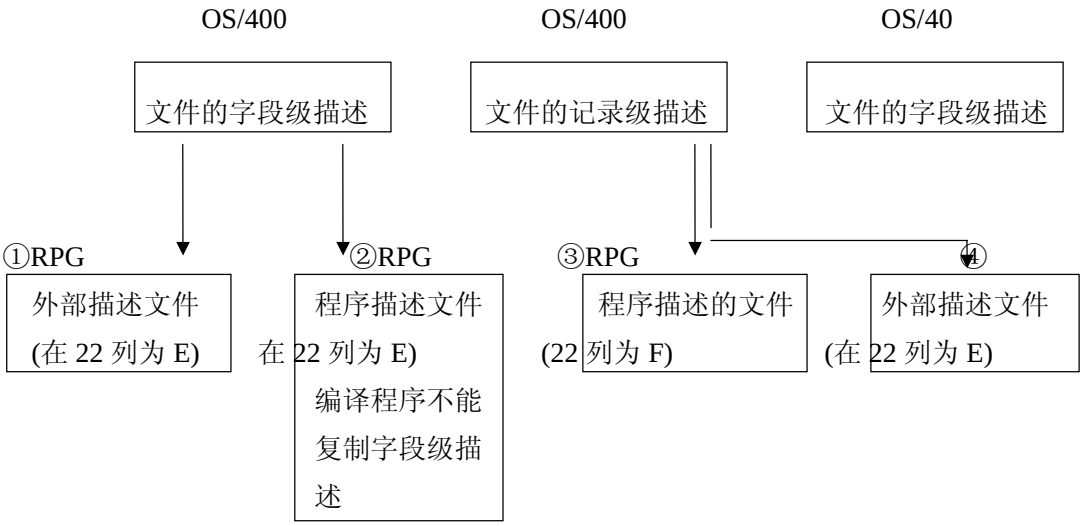


图 83　AS/400 系统中 RPG 程序和文件之间的典型关系

1、程序使用在 OS/400 上定义的字段级描述，外部描述文件在文件描述规范表的 22 列上用 E 标识，编译时，编译程序拷贝外部字段级描述的内容。

2、在 RPG 程序中，一个外部描述文件（即外部描述字段级文件）作为程序描述文件来使用。程序描述文件在文件描述规范表的 22 列上用 F 标识。这项告诉编译程序不用拷贝外部字段级描述的内容。这个文件在编译时可以不存在。

3、文件对 OS/400 仅做记录级描述。记录中的字段在程序中描述；所以，文件描述规范表的 22 列上必须是 F。这个文件在编译时可以不存在。

4、可以在编译时指定一个文件名（即在 RPG 源成员中编码），而在运行时指定一个不同的文件名，文件描述规范表的 22 列的 E 指出在编译时要拷贝该文件的外部描述。运行时，可以使用文件复盖命令，程序使用不同的文件。在运行时，复盖一个文件，必须确保两个文件中的记录名是相同的，RPG 程序在输入/输出操作上使用记录格式名：例如，READ 操作，它指出所要的记录在哪儿是什么样的记录类型。详细资料见 4.2.1。

13.4　定义外部描述文件

可以用 DDS 对 OS/400 系统描述文件。文件中的每个记录类型用一个唯一的记录格式名来标识。

文件描述规范表的 22 列上用 E 来标识一个外部描述文件。E 指出，程序被编译时，它是从系统接收该文件的外部描述的。

外部描述的内容包括：

- 文件信息，如文件类型，文件属性，比如，存取方式(用键字或者用相对记录号访问）。
- 记录格式描述，它包括记录格式名和字段的描述（名，位置和属性）。

编译程序从外部描述中检索信息，在编译源成员时使用*EXPDDS 选项的 CRTRPGMOD

或 CRTBNDRPG 命令，打印在编译清单上（这两个命令的缺省值为*EXPDDS）。

下面的部分介绍如何用文件描述规范表重命名或忽略记录格式及如何用输入/输出规范表修改外部描述规范。记住，输入/输出规范对外部描述文件是可选的。

13.4.1 重新命名记录格式名

对外部描述文件规定的许多功能（例如，象 CHAIN 操作），都是用文件名或者记录格式名来操作。因此，程序中的每个文件和记录格式名，必须是唯一的符号名。

要重新给记录格式命名，在文件描述规范表上用 RENAME 键字。格式是 RENAME（旧名：新名）。如图 84 所示。

```
     *..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+...
*

FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords++++++++++++++++++++++++++++
FITMMSTL    IP    E            K DISK     RENAME(ITEMFORMAT:MSTITM)       *
```

图 84　外部描述文件中记录格式名的 RENAME 键字

程序使用有相同记录格式名的两个文件时使用图 84 中的外部描述文件 ITMMSTL 中的 ITEMFORMAT 被重命名为 WSTITM，在这个程序中使用 WSTITM。

13.4.2 忽略记录格式

如果外部描述文件中的一个记录格式在程序中不用，可以用 IGNORE 键字，程序运行时就好象在文件中没有这个记录格式一样。对逻辑文件，这意味着与那个记录格式相联系的所有数据不能由程序访问。在外部描述文件的文件描述规范表中使用 IGNORE 键字如图 85 所示。

要忽略记录格式，文件必须有两个以上的记录格式而且不是所有的记录格式都能被忽略了至少必须保留一个。除这个必要条件之外，任意数目的记录格式都能被忽略。

一旦某记录格式被忽略，它就不能再用其它键字（SFILE，RENAME 或 INCLLODE），也不能说明另一个 IGNORE。

被忽略的记录格式名出现在交叉引用表上，但有标记说明它们是被忽略的。

要指出一个外部描述文件中被忽略的记录格式，可在文件描述规范表的键字区输入键字和参数 IGNORE（记录格式名）。

另外，可用 INCLUDE 键字说明程序中用到的记录格式名，而文件中其他的记录格式程序都不用。

```
     *..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+...
*

FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords++++++++++++++++++++++++++++++
*
*  假定文件中有以下记录格式：EMPLNO、NAME、ADDR、TEL、WAGE。
*  如果要让程序只处理 EMPLNO 和 NAME 记录，可用下面两种方法之一：
*
FITMMSTL   UF   E           K DISK    IGNORE(ADDR:TEL:WAGE)
*  OR:      *
FITMMSTL   UF   E           K DISK    INCLUDE(EMPLNO:NAME)
*
```

图 85　外部描述文件记录格式中的 IGNORE 键字

13.4.3 用输入规范表修改外部描述

可用输入规范表去修改输入文件的外部描述中的某些信息，或者对外部描述增加 RPG 的一些功能。在输入规范表上，可以做：

- 分配记录标识指示器（见图 86）。
- 重命名一个字段（见图 86）。
- 给字段分配一个控制级指示器（见图 86）。
- 给匹配记录处理的字段分配匹配字段值（见图 87）。
- 分配字段指示器（见图 87）。

不能用输入规范表修改外部描述文件中字段的位置。外部描述文件中的字段是用它们在数据描述规范中的顺序放置的。另外，设备相关功能（象格式控制），在 RPG 程序中对外部描述文件无效。

```
    *..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+... *
IRcdname+++....In.....................................................*
IMSTRITEM       01  1

I.............Ext-field+.................Field+++++++++L1M1..PlMnZr......
I               ITEMNUMB  2               ITEM          L1  3       *
IMSTRWHSE       02
I               ITEMNUMB                  ITEM          L1        *
```

图 86　对外部描述修改和增加 RPG 功能

1、给外部描述文件中的一个记录分配记录标识指示器，在输入规范表的 7－16 列上指

出记录格式名并在 21 和 22 列上分配一个有效的记录标识指示器。输入规范表和外部描述文件一起使用的一个典型用处是分配记录标识指示器。

在这个例子中，把记录标识指示器 01 分配给记录 MSTRITEM， 把指示器 02 分配给记录 MSTRWHSE。

2、要重命名外部描述记录中的一个字段，在字段描述行的 21－30 列上左对齐指出该字段的外部名，在 49－62 列上指出用在程序中的名。

在这个例子中，在两个记录中都有的字段 ITEMNUMB 被重命名为 ITEM。

3、要给外部描述记录中的字段分配控制级指示器，在 49－62 列指出字段名，在 63 和 64 列上指定一个控制级指示器。

在这个例子中，在两个记录 MSTRITEM 和 MSTRWHSE 中都有的 ITEM 字段说明了 L1 级控制字段。

```
*..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+...  *

IFilename++SqNORiPos1+NCCPos2+NCCPos3+NCC.............................
IMSTREC         01  1

I.............Ext-field+.................Field+++++++++L1M1..PlMnZr......
I                                         CUSTNO          M1  1
*
IWKREC       02
I                                         CUSTNO          M1
I                                         BALDUE                  98  2
*
```

图 87　对一个外部描述增加 RPG 功能

1、要给外部描述记录的字段分配一个匹配值，在记录标识行的 7－16 列上指出记录格式名，在字段描述行的 49－62 列指定字段名并在 65 和 66 列上分配一个匹配级值。

在这个例子中，给两个记录 MSTREC 和 WKREC 中都有的 CUSTNO 字段分配了匹配级值 M1。

2、要给外部描述记录中的字段分配字段指示器，在记录标识行 7－16 列上指定记录格式名，在字段描述行上，在 49－62 列指定字段名，在 69－74 列上指定一个指示器。

在这个例子中，记录 WKREC 中的字段 BALDUE，当被读到程序中时做是否为零的检测，如果字段的值是零，指示器 98 置 ON。

13.4.4 使用输出规范表

输出规范表对外部描述文件是可选的，RPG 支持使用外部记录格式描述的文件操作码，象 WRTTE 和 UPDATE，对外部描述文件不需要用输出规范表来描述输出记录。

可以用输出规范表控制什么时候写数据，或者有选择的写字段。外部描述文件的字段描述行的有效输入是输出指示器(21－29 列)，字段名(30－43 列)，及写后空(45 列)。外部描述文件中字段的编辑字和编辑码是在文件的 DDS 中说明的，设备相关功能，象取溢出(18 列)，或者空行/跳行(40－51 列)在 RPG 程序中对外部描述文件无效。外部描述文件溢出指示器也无效。在 DDS 中怎样指定编辑的介绍，见 DDS Refenence。

如果为一个外部描述文件使用输出规范表，在 7－16 列上用记录格式名代替文件名。

如果外部描述文件中的所有字段都放在输出记录中，在字段描述行的 30－43 列上填入*ALL。如果指定了*ALL，就不能在此记录中指定其它的字段描述行。

如果你想在输出记录上仅放某几个字段，在 30－43 列上填入字段名。在这些列上指出的字段名必须是在外部记录描述中定义的字段名，除非是在输入规范表中已重命名的字段。见图 88。

对外部描述文件使用输出规范表，应该了解以下的有关事项：

•在一个更新记录的输出中，仅在输出字段说明中指出的并且满足由输出指示器规定条件的那些记录才被放在要写出的输出记录中。没在输出规范表上说明的字段用读进来的值写出。这个技术提供了对更新全部字段的 UPDATE 操作做相对控制的一个好方法。

•在生成一个新记录时，在输出字段规范表上说明的字段被放在该记录中。没在输出字段规范表上说明的字段或者没有满足的由输出指示器所说明条件的哪些记录用缺省值写出，缺省值由在外部描述中所说明的数据类型来决定。（例如：字符型字段缺省为空格；数字型字段是零）。

```
*..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+... *

OFilename++DF..N01N02N03Excnam++++B++A++Sb+Sa+.........................*
OITMREC     D     20

O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++
O                         *ALL   1
*
OSLSREC     D     30
O                         SLSNAM   2
O                         COMRAT
O                    15   BONUS
```

图 88　外部描述文件的输出规范表

1.对一个更新文件，记录中的所有字段都用程序中记录所有字段的当前值写到外部描述记录 ITMREC 上。

对新建立的记录，记录中的所有字段都用程序中这个记录所有字段的当前值写到外部描述记录 ITMREC 上。

2.对于更新记录，当指示器 30 是 on 时，字段 SLSNAM 和 COMRAT 写到外部描述记录 SLSREC 中，当指示器 30 和 15 是 on 时字段 BONUS 写到记录 SLSREC 中，记录中的所有其他字段用读入的值写出。

对新生成的记录，当指示器 30 是 on 时，字段 SLSNAM 和 COMRAT 被写到外部描述记录 SLSREC 中。当指示器 30 和 15 是 on 时，字段 BONUS 被写出。记录中的所有其他字段用缺省值写出，缺省值取决于数据类型。（例：字符型字段为空格，数字型字段为零）。

13.4.5 处理浮点字段

ILE　RPG/400 程序不支持使用浮点字段。如果处理一个有浮点字段的外部描述文件，将发生下列情况：

- 不能访问浮点字段。
- 生成新记录时，记录中的浮点字段值为零。
- 更新存在的记录时，浮点字段不改变。
- 如果企图用一个浮点字段作一个键字字段，程序收到一个编译时错误。

13.4.6 处理变长字段

ILE　RPG/400 程序不支持使用变长字符或图形字段。但如果在 CRTRPGMOD 或 CRTBNDRPG 命令中指定了 CNTOPT(*VARCHAR)或 CVTOPT(*VARGRAPHIC)时，ILE RPG/400 编译程序会识别外部描述文件中的变长字符型或图形字段，并且作为定长字符字段带入程序。字段的头两个字节是可变的长度，剩下的字节是字符数据。

注意：当使用这些字段时，处理这些数据的依据是编译程序怎么样把数据放进定长字段中。这样，在新记录生成或修改时，必须确保包含可变长度的头两个字节正确地初始化。

如果在生成命令中没有指定 CVTOPT(*VARCHAR)或 CUTOPT(*VARGRAPHIC)编译程序会忽略外部描述的这些字段并且程序也不能访问这些字段。

当建立一个新记录，字段的头两个字节为零，剩余字节为空格。

修改已存在记录时，变度字段不改变。

如果试图用变度字段作为键字字段，程序收到编译时错误。

13.4.7 处理空值

ILE　RPG/400 程序不支持使用空值，但如果在 CRTRPGMOD 或 CRTBNDRPG 命令中指定 ALWNULL(*YES)，ILE　RPG/400 编译程序会接收仅输入文件中有空值字段的记录。空值会用缺省值读进，缺省值取决于外部描述中指定的数据类型。（例：字符型字段为空格；

数字型字段为零）。

如果在生成命令中没指定 ALWNULL(*YES)时企图检索一个有空值的记录，将发生数据映像错误，并且程序的记录中没有可访问的数据。

13.4.8 级别检查

由于 HLL 程序是依赖运行时所接收的外部描述文件的格式是否与编译时拷贝到程序中的格式相一致，因此系统提供一个级别检测功能，以确保格式始终是相同的。

当使用外部描述的 DISK，WORKSTN 或 PRINTER 文件时，RPG 编译器总是提供级别检查所需的信息。级别检查功能可以在建立，修改和覆盖文件命令上请求。建立文件命令的缺省值是请求级别检查。

级别检查在打开文件时，在记录格式主要部分上进行，除非覆盖命令或建立文件命令时指定 LVLCHK(*NO)。如果级别检查值不匹配，程序出错。RPG 程序随后处理 OPEN 错误。见 11 章 3.2 节“处理异常错误”。

RPG 程序对程序描述文件或使用 SEQ 及 SPECIAL 的文件不做级别检查。

有关如何指定级别检查的详细资料，见《数据管理》。

13.5 定义程序描述文件

记录和字段都在使用该文件的程序中由输入/输出规范表描述的文件是程序描述文件。

要在 RPG 程序中使用程序描述文件，必须做：

1.在文件描述规范表中标识文件。

2.如果是输入文件，在输入规范表中描述记录和字段。输入规范表的 7～16 列上的文件名必须和文件规范表中的名相同。

在记录标识项应该指出怎样在文件中做记录的顺序检查。

3.如果需要在计算规范表的因子 2 输入与第一步相同的文件名。例如，对程序描述文件的写操作需要在结果字段输入数据结构名。

4.如果是输出文件，在输出规范表中描述记录和字段。此外，还应该指出怎样打印输出。输出规范表的 7～16 列上的文件名必须和文件规范表中输入的名相同。

程序能运行之前，程序描述文件必须在系统中存在，并且加在库列表中。要建立这个文件，可用建立文件命令，此命令可在《CL 参考》中找到。

13.6 数据管理操作和 ILE　RPG　I/O 操作

数据管理是操作系统的一部分，它控制应用程序的数据存储和访问。表 12 表明了 AS/400 提供的数据管理操作和相应的 ILE　RPG/400 操作。它也列出了 ILE　RPG/400 的各种设备类型允许的操作。

表 12　数据管理操作和相应的 RPG　I/O 操作

* 表 12　数据管理操作和相应的 ILE　RPG/400　I/O 操作

*

*	* 数据管理操作	* ILE RPG/400 I/O 操作	*
*	* OPEN	* OPEN	*
*	*	*	*
*	* READ	*	*
*	*　By relative *　*　record number	* READ, CHAIN *	* *
*	*　By key	* READ, READE, CHAIN, primary file	*
*	*　Sequential	* READ	*
*	*　Previous	* READP, READPE	*
*	*　Next	* READ, READE	*
*	*　Invited Device	* READ	*
*	*	*	*
*	* WRITE-READ	* EXFMT	*
*	*	*	*
*	* WRITE	*	*
*	*　By relative *　*　record number	* WRITE *	* *
*	*　By key	* WRITE, EXCEPT, primary file	*

*	*	Sequential	* WRITE, EXCEPT *
*	*		* *
*	*	FEOD	* FEOD *
*	*		* *
*	*	UPDATE	* *
*	*	By relative record number	* UPDATE, primary file * * *
*	*	By key	* UPDATE, primary file *
*	*		* *
*	*	DELETE	* *
*	*	By relative record number	* DELETE, primary file * * *
*	*	By key	* DELETE, primary file *
*	*		* *
*	*	ACQUIRE	* ACQ *
*	*		* *
*	*	RELEASE	* REL *
*	*		* *
*	*	COMMIT	* COMMIT *
*	*		* *

*	* ROLLBACK	* ROLBK	*
*	*	*	*
*	* CLOSE	* CLOSE, LR RETURN	*
*	*	*	*

第十四章　一般文件考虑

这一章提供了在 AS/400 上使用 RPG 处理文件的一些考虑：

覆盖和改变文件的输入输出；

RPG 程序文件锁；

RPG 程序记录锁；

共享一个打开数据路径；

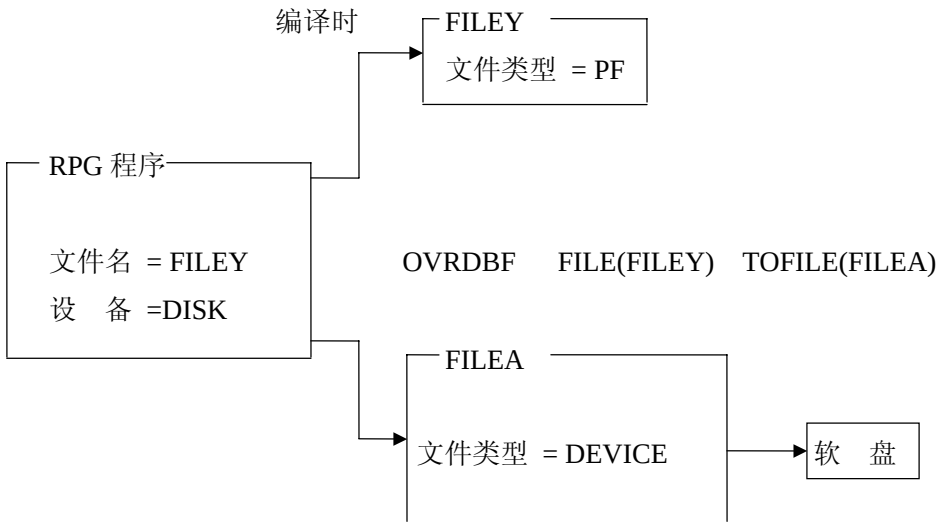
AS/400 假脱机功能；

非成块和成块记录对性能的影响；

在 RPG 程序中用 SRTSEQ/ALTSEQ 不用 DDS 文件。

14.1 覆盖和改变文件的输入输出

AS/400 命令可以在编译和运行时复盖文件描述规范中的参数或者定向一个文件。文件重定向允许在运行时指定一个文件，来替换程序中规定的文件（编译时）。



设备类型 = DISKETTE

在前面的例子中，CL 命令 OVRDBF(覆盖数据库文件)允许程序运行一个与编译时指定的完全不同的设备文件。想要在运行时覆盖文件，必须确保两文件中的记录名是相同的。RPG 程序在输入/输出操作上运用记录格式名，例如 READ 操作，它要规定使用的记录类型。

并不是所有的文件覆盖和重定向都是有效的。在运行时，要做检测来保证 RPG 程序中的规范表对正在被处理的文件是有效的。OS/400 系统允许某些在程序中规定的设备文件重定向。例如，如果 RPG 设备名是 PRINTER，而实际程序连接的不是打印机文件，OS/400 系统忽略 RPG 打印空行和跳行的规定。也有一些 OS/400 系统不允许文件重定向，它会导致程序结束。例如，如果 RPG 设备名是 WORKSTN，程序中又规定了 EXFMT 操作，而程序连接的实际文件既不是显示文件又不是 ICF 文件，则程序终止。在 ILE 中，覆盖适用于活动组级，作业级，或调用级。在活动组范围中的覆盖将会一直保持到它们被删除，替代或者活动组结束。作业级范围的复盖一直保持到它们被删除，替代或者作业结束。而和指定复盖的活动组无关。调用级范围的覆盖也一直有效，直到它们被删除、替代或规定覆盖的程序或过程结束。

覆盖的缺省范围是活动组，对于作业级，要在覆盖命令指定 OVRSCOPE(*JOB)，对于调用级，在覆盖命令中指定 OVRSCODE(*CALLLVL)。

关于文件重定向和文件覆盖的详细介绍见《数据管理》。《ILE 概念》中也有关于覆盖和活动组不用作业级范围的内容。

14.1.1 重定向文件输入输出的例子

下面的例子给出编译时文件覆盖的用法。假定要用一个没有字段级规定的磁带设备外部描述文件，必须：

1.定义一个物理文件 FMT1，它有一个记录格式并且包含记录格式中每个字段描述。记录格式在 DDS 中定义。对于磁带设备，外部描述文件只包含一个记录格式。

2.用建立物理文件的 CL 命令生成名为 FMT1 的文件。

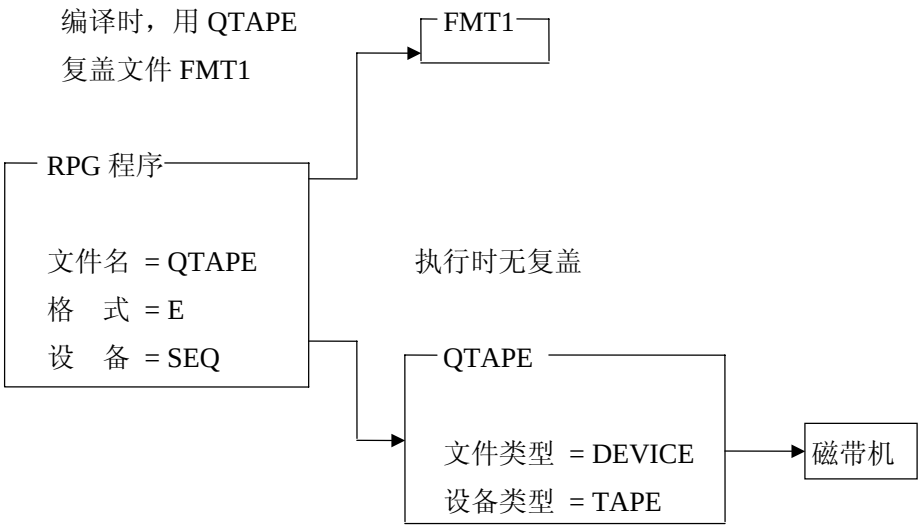
3.在 RPG 程序中指定名为 QTAPE 的文件。(QTAPE 是 IBM 支持的磁带设备文件名)，标识文件是外部描述的(文件描述规范表的 22 列指定 E)，在 36～42 列指定设备名为 SEQ。

4.编译时用覆盖命令--OVRDBF FILE(QTAPE) TOFILE(FMT1)，用 FMT1 文件覆盖 QTAPE 文件。这条命令让编译程序拷贝 FMT1 文件的外部描述，其中有记录格式的描述规定。

5.用 CRTBNDRPG 或 CRTPGM 命令建立 RPG 程序。

6.运行时调用程序。当程序运行时对文件 FMT1 的覆盖不会起作用，如果要让覆盖有效，在调用程序之前运行 CL 命令 DLTOVR(删除覆盖)。

注意：在调用程序之前一定要执行 OVRTAPF 命令，这样可以为打开磁带文件提供必要信息。



14.2 文件锁

在作业执行期间，OS/400 系统允许在它所用的文件上设置一个锁定状态（拒绝，只读，更新共享，非更新共享，或者读共享）。作业内的程序不受文件锁定状态的影响，文件锁定状态仅为另一个作业中的程序试图同时使用该文件时发生。文件锁定状态可以用 CL 命令 ALLOBJ（配置目标）设置。有关配置资源，锁定状态的详细资料，见《数据管理》。

当文件打开时，OS/400 在数据库文件上放置下列锁定状态：

文件类型	锁定状态
输入(Input)	读共享
更新(Update)	更新共享
增加(Add)	更新共享
输出(Output)	更新共享

读共享锁定状态允许其他的用户用读共享，更新共享，非更新共享，或者只读锁定状态来打开文件，但这个用户不能指定拒绝使用文件。更新共享锁定状态允许其他的用户用读共享或更新共享锁定状态打开文件。

RPG 程序在设备上设置只读锁定状态，其他用户可以用读共享锁定状态的打开文件。

如果用配置目标命令，可以改变由 RPG 程序放置在该文件上的锁定状态。

14.3 记录锁

当程序读一条记录时，用两种方式：输入和更新。如果程序为更新而读记录时，把锁放在记录上，其他的程序不能读同一个记录来做更新，直到第一个程序解锁。如果程序为输入而读一个记录时，记录上不会加锁。由一个程序锁定的记录可以被另一个程序用来做输入的读。

在 RPGⅣ程序中，可以用读更新文件做更新操作。除了更新以外其他类型的文件的读记录只能做查询。当然，从更新文件中读记录是为更新而读。对于更新文件，可以用输入操

作，象 CHAIN，READ，READE，READP 或 READPE，为指定读一条做输入的记录，可在操作码之后指定一个操作码扩展(N)。

当 RPGⅣ程序锁上一条记录时，直到下列情况出现才解锁：

- 记录被更新
- 记录被删除
- 从文件中读了另外的记录（既可以是查询又可以是更新）
- 对文件完成了 SETLL 或 SETGT 操作
- 对文件完成 UNLOCK 操作
- 对文件完成一个没有字段名和输出规范表定义的输出操作

注意：对文件增加记录的操作不会导致记录解锁。

如果程序为更新读一条记录，而那条记录又被作业中的其他程序或另外作业锁定，读操作将等待直到该记录被解锁（除非是共享文件，见 4.2.4）。如果等待时间超出了文件中 WAITRCD 规定的值，将发生一个异常。当记录锁出现超时时，如果你的程序没有处理这个异常(RNX1218)，那么控制转给缺省的错误处理程序，并且发出一条 RNQ1218 询问信息。这个信息列出的可选操作之一是在超时出现时，再重做一次操作。这会导致超时出现时操作会重新发出，程序继续做下去，就好象没有出现记录锁超时一样。

注意：如果文件规定了 INFSR，它的一次 I/O 操作，在缺省错误管理程序得到控制之前完成，如果输入操作是顺序操作，由于文件指针已经改变，将会出现意外的结果。

如果不对上锁记录做修改，可以释放加锁状态，这时不用修改文件指针，可使用 UNLOCK 操作或处理用不包括字段名的输出规范表规定的输出操作。这些输出操作可由 EXCEPT 输出，明细输出或者总计输出执行。

当操作是在落实控制下时，这些规则可能有些变化。详细资料见 4.3.6 节“使用落实控制”。

14.4 共享打开数据路径（共享 ODP）

一个打开数据路径是对文件进行的全部输入和输出操作所使用路径。通常，每打开一个文件就定义一个打开数据路径。但如果在文件生成或者复盖时指定 SHARE(*YES)，则该文件的第一个程序的打开路径同时打开这个文件，为其他程序所共享。当前记录的位置保留在打开数据路径中为使用该文件的所有的程序使用。如果用一个程序读一个记录然后用一个被调用的程序读一个记录，由第二个读操作接收的记录取决于是否共享打开数据路径。如果是共享被调用程序中当前记录的位置由调用程序中当前位置来决定。如果不共享，对当前记录每个程序有一个独立的位置。

如果第一个程序在共享文件中保持一个记录锁；然后调用一个读共享文件做更新第二个程序，你会通过下列方法解除第一个程序的锁。

由第二个程序在更新文件上完成一个读操作，使用 UNLOCK 或读无锁操作。

在 ILE 中，共享文件能用于作业级也能用于活动组级。作业级的共享文件能

被作业中运行的任何活动组中的程序所共享。活动组级的共享文件只能被在同一活动组中运行的程序所共享。

共享文件的缺省范围是活动组级。对于作业级范围，要在覆盖命令中指定OVRSCOPE(*JOB)。

ILE RPG/400 在 ODPS 区有些增强功能。如果一个程序或过程完成了一次读操作，另外的程序或过程会在文件中指定 SHARE(*YES)时更新记录。另外，使用多设备文件时，如果一个程序获得了一个设备，共享 ODP 的其他程序也会获得这个设备。要保证需要完成更新的所有数据对被调用的程序都有效，对编程人员来说要求比较高。

由于 OS/400 系统不必再生成新的打开数据路径来共享打开数据路径，从而改善了性能。但是，共享打开数据路径也可能引起一些问题。例如，在下列情况下要出错：

•如果一个共享打开数据路径的程序试图做了不是第一次打开所做的那些操作。（例　　如，第一次打开仅指定了输出操作而在共享时企图执行输入操作。）

•如果一个共享外部描述文件打开数据路径的程序，试图去使用第一个程序忽略的记　　录格式。

• 如果一个共享程序描述文件打开数据路径的程序，指定了一个记录长度，它超出了　　第一次打开时建立的长度。

当一个程序中的几个文件在运行时被覆盖为一个共享文件时，文件打开顺序是重要的，为控制文件打开顺序，调用程序前应该用程序员控制的打开或者用 CL 程序打开文件。

如果一个程序共享了主文件或次文件的打开数据路径，程序必须在调用共享打开数据路径的另一个程序之前处理记录的明细计算。另外，如果用了先行或者在总计时做调用，共享主文件或次文件的打开数据路径可能使被调用的程序从文件中错误的记录中读数据。必须确认当共享文件第一次打开时，要规定好该文件所有随后打开的要求选项。如果这些选项没指定，在共享文件第一次打开时程序接收到一条错误信息。

表 13 详细列出了可以指定的系统打开选项。

表 13

RPG 用户打开选项	系统打开选项
INPUT	INPUT
OUTPUT	OUTPUT　（程序建立的文件）
UPDATE	INPUT，UPDATE，DELETE
ADD	OUTPUT（已存在文件）

关于共享打开数据路径的详细资料，见《DB2/400 数据库编程》，ILE 概念

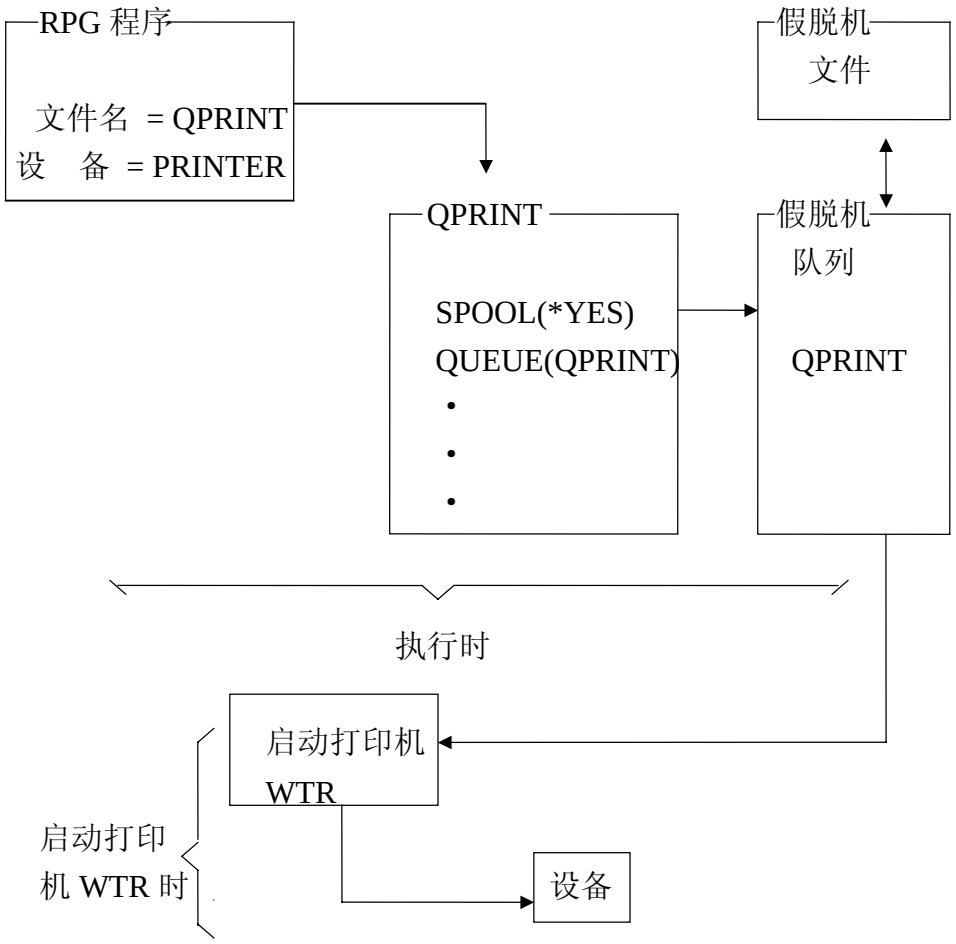
中也有它的资料。

14.5 假脱机

假脱机是一个系统功能，它把数据放到一个存储域中等待处理。AS/400 系统提供了用于输入和输出的假脱机功能。每个包含假脱机属性的文件描述决定了运行中的文件是否用到了假脱机功能。RPG 程序不知道它使用的假脱机。从实际物理设备文件读或写是由假脱机读或假脱机写决定的。关于假脱机详细资料，见《数据管理》。

14.5.1 输出假脱机

输出假脱机对批作业或交互作业都有效，由文件名在 RPG 程序内所说明的文件，包括了假脱机的说明，如下图所示：



在运行时可用文件覆盖命令覆盖假脱机在文件描述上指出的可选项，比如要打印的份数等。另外，AS/400 假脱机支持用户在程序运行以后更换一个文件，他可以把同样的打印输出到一个不同的设备上，比如软盘上。

14.6 成块和非成块记录

如果下列条件成立，RPG 编译程序对 SEQ 或 DISK 文件将非成块输入记录和成块输出记录：

• 文件是仅输出文件（在文件规范表的 17 列指定为 0）而且如果文件是外部描述的，仅有一个记录格式。

• 文件是仅输入文件（文件描述规范表 17 列是 I），如果文件是外部描述的，仅有一个记录格式，而且仅使用 OPEN，CLOSE，FEOD 和 READ 操作码。

文件描述规范表上没指定 RECNO 键字。

RPG 编译对所有满足这些条件的 SEQ 或 DISK 文件生成成块和非成块记录的目标程序码。某些 OS/400 的限制可能不支持成块和非成块。在这种情况下，性能无法改善。

文件信息数据结构的输入/输出和设备相关反馈信息在每次对由 RPG 编译程序做的成块和不成块记录的读或写以后都不更新，（除非在做块读入时规定 RRN 或键字信息）反馈仅是在每次记录块转换时更新。详细的文件信息数据结构的资料见《ILE　RPG/400 参考》。

可用指定 SEQ ONLY(*NO)的 CL 命令 OVRDBF 来获得有效的更新反馈信息。如果用文件覆盖命令，RPGⅣ语言不在文件中做成块和非成块记录。

14.7　RPG 程序中的 SRTSEQ/ALTSEQ

用 SRTSEQ 和 LANGID 建立键字文件时，对文件做 CHAIN，SETLL，SETGT，READE 和 READPE 操作期间，SRTSEQ 要与字符键字比较。在建立 RPG 程序和过程时不必指定相同或另外的 SRTSEQ。在 CRTBNDRPG 或 CRTRPGNVOD 中指定 SRTSEQ 值时，程序中所有的字符比较操作都会用 SRTSEQ。这个值影响到所有字段的比较，包括键字字段，从其他文件中来的字段和在程序中说明的字段。

程序员应该决定在 RPG 程序中用 SRTSEQ 的依据是如何用象 IF xx，COMP 和 SORTA 一类操作去处理字符数据，而不是在建立文件时指定了什么。

第十五章　访问数据库文件

可在文件规范表中用文件名和 DISK 设备，从程序中访问数据库文件。

DISK 文件也同样和 DDM（分布数据管理）文件相连系，它允许访问远程系统文件作为数据库文件。

15.1 数据库文件

数据库文件是 AS/400 中类型为*FILE 的目标。它们既可以是物理文件又可以是逻辑文件，既可以是外部描述的又可以是程序描述的。可以通过文件描述规范表中的文件名和 36--42 列上指出 DISK 磁盘设备访问数据库文件。

数据库文件可用 OS/400 建立文件命令来生成。关于生成和描述数据库文件的资料，参考《DB2/400 数据库编程》和《DDS 参考》。

15.1.1 物理文件和逻辑文件

物理文件含有实际的数据，它也描述了这些数据如何在程序中出现和由程序接收。文件中仅有一个记录格式，允许有一个或多个成员。数据库文件中记录可以是外部描述的也可以是程序描述的。物理文件可以有键字顺序的访问路径。就是说，数据是根据文件中一个或多个键字字段的顺序向程序中传送数据。

逻辑文件不包含数据，它们有访问一个或更多物理文件中记录的路径。逻辑文件是一个或多个物理文件的视图或代表。有多个记录格式的逻辑文件叫做多格式逻辑文件。

如果程序处理多格式逻辑文件，可以用记录格式读去设置要用的记录格式。

15.1.2 数据文件和源文件

数据文件含有实际数据，或数据的视图。文件中的记录被组织在成员中，文件中所有记录可能在同一成员中，也可能在不同成员中。大多数数据库命令和操作的缺省值都假定数据库文件仅有一个成员。这意味着程序访问含有数据的数据库文件时，不必指定文件中的成员名，除非文件中有多于一个的成员。如果文件中有多个成员而又没指定哪个成员，将用第一个成员。

一般来讲，包含源程序的数据库文件经常由多个成员组成。把源程序组织到数据库文件的成员中是为了更好地管理程序。源成员包括系统用来建立程序目标的源语句。

15.2 使用外部描述的磁盘文件

外部描述的 DISK 文件是在文件描述规范表 22 列上用 E 来标识。E 指出，

当程序编译时，编译程序从系统接收该文件的外部描述。所以，文件必须在编译程序之前建立。

一个 DISK 文件的外部描述包括：

• 记录格式说明，包括记录中字段的描述。

• 访问路径说明，它描述了怎样去接收记录。

这些说明从该文件的 DDS 中产生，并用 OS/400 建立文件命令生成该文件。

15.2.1 记录格式说明

记录格式说明描述记录中的字段和它的位置。字段的位置按在 DDS 中所说明的顺序排放。字段描述通常包括字段名，字段类型以及字段长度（数字型字段要有小数位数）。可以用一个字段引用文件来定义字段属性，可代替用记录格式为一个物理文件或逻辑文件指出字段属性。

此外，DDS 关键字可用来：

• 指出文件不允许重复的键字值(UNIQUE)

• 指出记录格式或字段的说明(TEXT)

对于数据库文件有效的完整的 DDS 的关键字，见《DB2/400 数据库编程》。

4.3.2.2 的图 89 表示一个数据库文件的 DDS 例子，图 90 是字段引用文件的例子，它定义了数据库文件中所用字段的属性。有关字段引用文件的详细介绍，见《DDS 参考》。

15.2.2 访问路径

外部描述文件的描述包括访问路径，它描述了怎样从文件中接收记录。可用到达顺序（无键字）访问路径或键字顺序访问路径来检索记录。

到达顺序访问路径是根据记录在文件中存储的顺序。往文件中加记录是一个接着一个记录进行。

对键字顺序访问路径，文件中记录的顺序是根据键字字段的内容，键字字段是在文件中的 DDS 中定义的。例如，图 89 所示的 DDS 例子中，CUST 是定义作键字字段。每当记录增加、删除或者是键字字段内容修改时，就更新了键字顺序访问路径。

有关外部描述的数据库文件访问路径的完整描述见《DB2/400 数据库编程》。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
A.........T.Name++++++.Len++TDpB......Functions++++++++++++++++++++*
A** LOGICAL  CUSMSTL    CUSTOMER MASTER FILE
A                                      UNIQUE
A          R CUSREC                    PFILE(CUSMSTP)
```

```
A                                     TEXT('Customer Master Record')
A            CUST
A            NAME
A            ADDR
A            CITY
A            STATE
A            ZIP
A            SRHCOD
A            CUSTYP
A            ARBAL
A            ORDBAL
A            LSTAMT
A            LSTDAT
A            CRDLMT
A            SLSYR
A            SLSLYR
A          K CUST
```

图 89　数据库文件的 DDS 描述

这个简单的 DDS 是顾客主逻辑文件 CUSMSTL 的描述。该文件有一个记录格式 CUSREC（顾客记录）。这个文件的数据包括在物理文件 CUSMSTP 中，用键字 PFILE 来说明。UNIQUE 键字指出，这个文件不允许重复的键字值。CUST 字段在最后一行的 17 列上用 K 来标识，它是这个记录格式的键字字段。

记录格式中所包括的字段按它们在记录中出现的顺序列出。字段的属性从物理文件 CUSMSTP 中取得。该物理文件用一个字段引用文件来得到字段的属性。图 90 表示了字段引用文件。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
A.........T.Name++++++RLen++TDpB......Functions++++++++++++++++++++*
A**FLDRED    DSTREF     DISTRIBUTION APPLICATION FIELD REFERENCE
A          R DSTREF                    TEXT('Distribution Field Ref')
A* COMMON FIELDS USED AS REFERENCE
A            BASDAT         6  0       EDTCDE(Y)  1
A                                       TEXT('Base Date Field')
A* FIELDS USED BY CUSTOMER MASTER FILE
A            CUST           5          CHECK(MF)  2
A                                       COLHDG('Customer' 'Number')
A            NAME          20          COLHDG('Customer Name')
```

```
A            ADDR      R              REFFLD(NAME)  3
A                                     COLHDG('Customer Address')
A            CITY      R              REFFLD(NAME)  3
A                                     COLHDG('Customer City')
A            STATE          2         CHECK(MF)  2
A                                     COLHDG('State')
A            SRHCOD         6         CHECK(MF)  2
A                                     COLHDG('Search' 'Code')
A                                     TEXT('Customer Number Search +
A                                     Code')
A            ZIP            5  0      CHECK(MF)  2
A                                     COLHDG('Zip' 'Code')
A            CUSTYP         1  0      RANGE(1 5)  4
A                                     COLHDG('Cust' 'Type')
A                                     TEXT('Customer Type 1=Gov 2=Sch+
A                                     3=Bus 4=Pvt 5=Oth')
A            ARBAL          8  2      COLHDG('Accts Rec' 'Balance')  5
A                                     EDTCDE(J)  6
A            ORDBAL    R              REFFLD(ARBAL)
A                                     COLHDG('A/R Amt in' 'Order +
A                                     File')
A            LSTAMT    R              REFFLD(ARBAL)
A                                     COLHDG('Last' 'Amount' 'Paid')
A                                     TEXT('Last Amount Paid in A/R')
A            LSTDAT    R              REFFLD(BASDAT)
A                                     COLHDG('Last' 'Date' 'Paid')
A                                     TEXT('Last Date Paid in A/R')
A            CRDLMT    R              REFFLD(ARBAL)
A                                     COLHDG('Credit' 'Limit')
A                                     TEXT('Customer Credit Limit')
A            SLSYR     R+   2         REFFLD(ARBAL)
A                                     COLHDG('Sales' 'This' 'Year')
A                                     TEXT('Customer Sales This Year')
A            SLSLYR    R+   2         REFFLD(ARBAL)
A                                     COLHDG('Sales' 'Last' 'Year')
A                                     TEXT('Customer Sales Last Year')  7
```

图 90　字段引用文件的例子

字段引用文件的例子给出了图 89 中 CUSMSTL（顾客逻辑文件）文件中所用字段的定义，字段引用文件一般包括另一个文件所用字段的定义。下面介绍这个引用文件的一些内容：

1.BASDAT 字用 Y 编辑码编辑，用键字 EDTCDE(Y)指出。如果这个字段用在 ILE RPG/400 程序的外部描述的输出文件中，所用的编辑码在字段引用文件中指出；它不能由 ILE RPG/400 程序覆盖。可是，如果字段是用在 ILE RPG/400 程序描述的输出文件中，编辑码必须在输出规范表中为该字段说明。

2.(HECKLMF)项说明，当它从一个显示工作站进入时，该字段是一个强制填充字段。强制填充的意思是这个字段的所有字符必须从显示工作站进入。

3.字段 ADDR 和 CITY 共享相同属性，该属性由 NAME 字段定义，用键字 REFFLD 来指出。

4.键字 RANGE 说明字段 CUSTYP，保证从显示工作站进入到这个字段中有效数字仅是 1--5。

5.键字 COLHDG 说明交互式数据库实用程序(IDU)使用该字段时的列标题。

6.字段 ARBAL 用 J 编辑码，用键字 EDTCDE(J)指出。

7.为某些字段提供的注释说明(TEXT 键字)。TEXT 键字用于给文件做说明，并可在各种列表上出现。

15.2.3 记录或文件的有效键字

对一个键字顺序访问路径，可以在 DDS 中定义一个或多个字段作为一个记录格式的键字字段。（不能用浮点字段，变长字段和空值字段作键字字段。）一个文件中所有记录类型不必非要有相同的键字字段。例如，一个订货标题记录中 ORDER 字段被定义为键字字段，而订货的明细记录中 ORDER 和 LINE 字段被作为键字字段定义。

一个文件的键字是通过文件中记录类型的有效键字来决定。文件的键字用下列方式决定：

• 如果文件中全部的记录类型有相同数目的键字字段，（它们的属性在 DDS 中定义），则文件的键字由记录类型的所有键字字段构成。（相应的字段不必有相同的名）。例如，如果文件有三个记录类型，每个记录类型由字段 A，B 或 C 构成键字，那么文件的键字由字段 A，B 和 C 构成。即，文件的键字和记录的键字是相同的。

• 如果文件的所有记录类型没有相同的键字字段，文件的键字由所有记录类型的公共键字来决定。例如，一个文件有三个记录类型，定义的键字字段如下：

—REC1 有键字字段 A

—REC2 有键字字段 A 和 B

—REC3 有键字字段 A，B 和 C

那么，文件的键字是字段 A--该键字字段是所有记录类型都有的。

—如果所有的记录类型中没有共同的键字字段，则文件没有键字。在 ILE

RPG/400 中，可以在某些文件操作码上指定一个检索自变量来标识要处理的记录。ILE RPG/400 把文件或记录的键字和检索自变量进行比较，而处理那些键字和检索自变量相匹配的那些记录的操作。

15.2.3.1 有效的检索自变量

可以在 ILE RPG/400 操作码 CHAIN，DELETE，READE，READPE，SETGT 和 SETLL 上规定文件名或记录格式名，再说明一个检索自变量。

对文件名的操作，在检索自变量中能指定的字段最大数等于文件有效键字字段总数，例如，文件中全部的记录类型不含有相同键字字段，可以用一个键字列表（KLIST）来说明一个检索自变量，它由文件中所有记录类型的共同字段构成。如果一个文件有三个记录类型，而键字字段做如下定义：

—REC1 有键字字段 A

—REC2 有键字字段 A 和 B

—REC3 有键字字段 A，B 和 C

检索自变量只能有一个字段 A，因为三个记录类型仅有的公共键字字段是 A。检索自变量不能有浮点字段，变长字段或者空值字段。

对一个记录名的操作，在检索自变量中能说明的最大键字字段数等于那个记录类型的有效的键字字段的总数。

如果检索自变量由一个字段构成，可以说明一个文字，一个字段名或一个有一个 KFLD 的 KLIST 名。如果检索自变量是由多个字段（一个复合键字）构成，你必须说明带多个 KFLD 的 KLIST。

检索自变量中每个字段的属性必须用文件或记录键字字段一致的属性。该属性包括长度，数据类型以及小数位。这些属性在编译列表的键字字段信息数据表上列出。见“键字字段信息”的例子。

在所有这些文件操作中(CHAIN，DELETE，READE，READPE，SETGT，SETLL)，也可以说明少于文件或记录中有效字段总数的检索自变量。这样的检索自变量引用部分键字。

15.2.3.2 引用部分键字

引用部分键字的检索自变量说明的规则如下：

• 检索自变量是由文件或记录键字的最左边的字段（高位）组成。

• 对部分键字的检索自变量，仅最右边的字段可以从键字列表(KLIST)中省略，例如，文件或记录的总键字是由键字字段 A，B 和 C 构成。那么有效的部分键字检索自变量是字段 A，以及字段 A 和 B。

• 检索自变量中的每个字段必须用和文件或记录中键字字段一致的属性标识。该属性包括长度，数据类型，小数位以及格式（例如，压缩型或区十进制型）。

• 一个检索自变量不能引用一个键字字段的一部分。

如果一个检索自变量引用一个部分键字，文件定位在满足检索自变量的第一个记录上或者接收的记录是满足检索自变量的第一个记录。例如，SETGT 和 SETLL 操作把文件定位在满足该操作和检索自变量的访问路径中的第一个记录上。CHAIN 操作接收满足检索自变量访问路径上的第一个记录。DELETE 操作删除满足检索自变量访问路径上的第一个记录。如果访问路径上记录（或规定类型的记录）的键字部分和检索自变量相匹配，READE 读取下一个记录。READPE 读取前一个记录。

关于这些操作码的详细资料，见《ILE RPG/400 参考》。

15.3 使用程序描述的磁盘文件

程序描述文件，在文件描述规范表上 22 列用 F 标识，可以说明为索引文件，顺序文件或记录地址文件。

15.3.1 索引文件

一个索引文件是一个程序描述的 DISK 文件，访问路径是用键字值建立的。必须通过使用数据描述规范为一个索引文件建立访问路径。

一个索引文件用文件描述规范表的 35 列的一个 I 来标识。

在索引文件中用键字字段标识记录。在文件描述规范表的 29--33 列上指定键字字段的长度，在 34 列上指出键字字段的格式，用 KEYLOC 键字给出键字字段的开始位置。

一个索引文件可以用键字顺序处理，界限内顺序处理，或者用键字随机处理。

15.3.1.1 有效的检索自变量

对一个程序描述的 DISK 文件，检索自变量必须是一个字段。对于 CHAIN 和 DELETE 操作，检索自变量必须与在索引文件的文件描述规范表中定义的键字字段有相同的长度。对于其他文件操作，检索自变量可以是一个部分字段。

由 DDS 指出用做键字的字段。文件描述规范表的 KEYLDC 键字指定了第一个键字字段的开始位置。文件描述规范表的 29～33 列上的内容必须指出在 DDS 中定义的键字的长度。

图 91 和 92 给出用 DDS 描述索引文件访问路径的例子。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
A.........T.Name++++++.Len++TDpB......Functions++++++++++++++++++++*
A           R FORMATA                   PFILE(ORDDTLP)
```

```
A                                     TEXT('Access Path for Indexed +
A                                     File')
A            FLDA          14
A            ORDER          5  0
A            FLDB         101
A          K ORDER
A*
 *.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords++++++++++++++++++++++++++++++
FORDDTLL   IP   F  118     3PIDISK    KEYLOC(15)
F*
```

图 91　DDS 和相应的文件描述规范表定义索引文件的访问路径

必须用数据描述规范表来建立一个程序描述的索引文件的访问路径。

在 DDS 中逻辑文件 ORDDTLL 的记录格式为 FORMATA，五位长的字段 ORDER 定义为压缩型的键字字段。ORDER 作为键字字段为这个文件建立键字访问路径。其他的两个字段，FLDA 和 FLDB 为字符型字段。

在文件描述规范表上把程序描述的输入文件 ORDDTLL 定义为索引文件。29～33 列必须指出在 DDS 中定义的键字字段在记录中的位数，即占 3 个位置。KEYLOC 键字指出 15 列为记录中键字字段的开始位置。因为这个文件是用 22 列上的 F 定义为程序描述的，ILE　RPG/400 在程序编译时不接收文件的外部字段描述；所以，记录中的字段必须在输入规范表上描述。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
A.........T.Name++++++.Len++TDpB......Functions++++++++++++++++++++*
A          R FORMAT                   PFILE(ORDDTLP)
A                                     TEXT('Access Path for Indexed +
A                                     File')
A            FLDA          14
A            ORDER          5
A            ITEM           5
A            FLDB          96
A          K ORDER
A          K ITEM
```

图 92(2-1)　用 DDS 定义索引文件的访问路径（复合键字）

在这个例子中，数据描述规范表定义了逻辑文件 ORDDTLL 中记录格式 FORMAT 的两个键字字段。要使这两个字段用作程序描述索引文件的复合键字，这两个字段在记录中必须是相邻的两个键字字段。

在文件描述规范表上，键字字段在 29～33 列上定义为 10 个长（是 ORDER 和 ITEM 两个字段位数和）。用键字 KEYLOC 规定键字字段开始位置是 15（44 列开始）。这个开始位置必须是第一个键字字段的第一个位置。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords++++++++++++++++++++++++++++++
FORDDTLL   IP   F  120    10AIDISK    KEYLOC(15)

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++++++
DKEY         DS
D K1                       1      5
D K2                       6     10

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
CLONO1Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....
C                   MOVE      ORDER         K1
C                   MOVE      ITEM          K2
C     KEY           CHAIN     ORDDTLL                             99
```

图 93(2-2)　用数据描述规范表定义索引文件的访问路径（复合键字）

当 DDS 说明一个复合键字时，必须在程序中建一个检索自变量给访问文件的 CHAIN 使用。（KLIST 不能用于程序描述文件）。一个方法是生成一个数据结构（使用定义规范表），它的子字段等于 DDS 中定义的键字字段，然后，在计算中设置这个子字段等于键字字段的值。在 CHAIN 操作中用数据结构名作为检索自变量。

在这个例子中，MOVE 操作使子字段 K1 和 K2 分别等于 ORDER 和 ITEM 的值。然后在 CHAIN 操作中用数据结构名(KEY)作为检索自变量。

15.3.2　顺序文件

文件中记录的顺序由记录放到文件中的顺序（即到达顺序）决定的，这样的文件叫顺序文件。例如，放在文件中的第 10 个记录有第 10 个记录位置。

顺序文件可以用相对记录号随机地处理，或者用一个记录地址文件来处理。另外，可用 SETLL 或 SETGT 操作码在文件上设置界限。

15.3.3 记录地址文件

可用一个记录地址文件处理另外的文件。一个记录地址文可以包括⑴界限记录，用来顺序处理界限内文件，或者⑵相对记录号，用相对记录号来处理文件。记录地址文件本身必须按顺序处理。

在文件描述规范表的 18 列上用 R 来标识一个记录地址文件，如果记录地址文件中有相对记录号，在 35 列上必须有一个 T。由记录地址文件处理的文件名必须在文件描述规范表中指出，可用键字 RAFDATA(文件名)来指定文件。

15.3.3.1 界限记录

要做界限内顺序处理，记录地址文件要有界限记录。界限记录包括文件中要读记录的最低记录键字和最高记录键字。在记录地址文件中界限记录的格式为：

• 低键字在记录的第一列开始；高键字紧随其后，两者之间不能有空格。

• 记录地址文件中的每个记录仅有一个界限设置。记录长必须大于或等于记录键字长的两倍。

• 界限记录中的低键字和高键字必须长度相同。键字长度必须等于要处理文件的键字字段的长度。

•记录键字字段长度是空格导致 RPG/400 编译器读记录地址文件中下一个记录。

15.3.3.2 相对记录号

用相对记录号处理，记录地址文件包括相对记录号。根据记录地址文件中的相对记录号，从要处理的文件中取每一个记录。有相对记录号的记录地址文件不能用于界限处理。记录地址文件中每个相对记录号是一个多字节的二进制字段。根据源文件可以指定记录地址文件长为 4、3 或者空格。在 AS/400 环境中，当使用记录地址文件时，指定记录地址文件长度为 4，因为每个字段是 4 个字节长。在系统 36 环境中，同样原因记录地址文件长是 3。如果指定记录地址文件长是空格，编译程序会在运行时检查主记录长度来决定记录长是 3 还是 4。负数 1（-1 或十六进制的 FFFFFFFF）的相对记录号值停止使用相关的记录地址文件。当记录地址文件中所有记录处理完时文件结束。

15.4 处理磁盘文件的方式

处理磁盘文件的方式包括：

• 连续处理
• 用键字顺序处理
• 用键字随机处理
• 界限内顺序处理
• 相对记录号处理

图 94 表示了各种文件类型和处理方式在文件描述规范表上 28、34 和 35 列上的内容。

处理方式	界限处理 28 列	记录地址类型 34 列	文件组织 35 列
外部描述文件			
用键字			
顺序处理	空格	K	空格
随机处理	空格	K	空格
界限内顺序处理（用记录地址文件）	L	K	空格
不用键字	空格	空格	空格
随机/连续			
程序描述文件			
用键字（索引文件）			
顺序处理	空格	A,D,G,P,T 或 Z	I
随机处理	空格	同上	I
界限内顺序（用记录地址文件）	L	同上	I
不用键字			
随机/连续	空格	空格	空格
用记录地址文件	空格	空格	空格
作记录址文件（相对记录号）	空格	空格	T
作记录地址界限文件	空格	A,D,G,P,T,Z 或空格	空格

图 94　磁盘文件处理方式

15.4.1 连续处理

连续处理期间，用记录在文件中出现的顺序来读。

对输出文件和不使用随机功能（象 SETLL，SETGT，CHAIN 或 ADD）的输入文件。ILE　RPG/400 编译 程序用缺省值在 CL 命令 OVRDBF(覆盖数据库文件)中指出的 SEQONLY(*YES)处理。ILE　RPG/400 对更新文件不做这样的处理，SEQONLY(*YES)允许把多个记录放在内部数据管理缓冲器内；然后把输入

记录一次一个地传送给 ILE　RPG/400。在同一个作业或活动组中，如果两个逻辑文件用同一个物理文件，一个是顺序处理而另一个是随机更新处理，放在缓冲区内的记录都能被更新情况下，当从顺序文中处理记录时，记录不反映更新的数据。要解决这个问题，用 CL 命令 OVRDBF 并说明可选项 SEQONLY(*NO)，它指出不需要为一个顺序处理的文件传输多个记录。

有关处理顺序文件的资料，见《DB2/400 数据库编程》。

15.4.2 用键字顺序处理

用键字顺序处理方式，是以键字顺序从文件中读记录。

键字顺序处理方式，对用作主文件，次文件或全过程文件的键字文件有效。

对输出文件以及不用随机功能（象 SETLL，SETGT，CHAIN 或 ADD）并仅有一个记录格式的输入文件，ILE　RPG/400 用缺省值或用在 CL 命令 OVRDBF 上指出的 SEQONLY(*YES)那样操作。(ILE　RPG/400 对更新文件不做那样处理)。SEQONLY(*YES)允许把多个记录放在内部数据管理缓冲区内；然后，一次一个地把这些记录传送到 RPG/400 中。在同一个作业中，如果两个文件用同一个物理文件，一个文件是顺序处理而另一个文件是随机更新处理，已经放在缓冲区内的记录都能被更新。在这种情况下，当从顺序文件中处理的记录时，该记录不反映更新的数据。要解决这个问题，用 CL 命令 OVRDBF 并说明可选项 SEQONLY(*NO)，它指出不需要为一个顺序处理的文件传送多个记录。

有关顺序处理的资料，见《DB2/400 数据库编程》。

15.4.2.1 用键字顺序处理的例子

下面的三个例子表示了用键字顺序处理的不同方法。

图 95 和图 96 是例子中用到的物理文件的 DDS。

图 97 是头三个例子中用到的逻辑文件的 DDS。

```
A*********************************************************************
A* 说明：这是物理文件 EMPMST 的 DDS，它有一个记录格式 EMPREC，      *
A*       对公司的每个雇员都有一条记录。                               *
A*********************************************************************
A*
A          R EMPREC
A            ENUM            5  0       TEXT('EMPLOYEE NUMBER')
A            ENAME          20          TEXT('EMPLOYEE NAME')
A            ETYPE           1          TEXT('EMPLOYEE TYPE')
A            EDEPT           3  0       TEXT('EMPLOYEE DEPARTMENT')
```

```
A            ENHRS          3  1       TEXT('EMPLOYEE NORMAL WEEK HOURS')
A          K ENUM
```

图 95　数据库文件 EMPMST（物理文件）的 DDS

```
A*********************************************************************
A* 说明：这是物理文件 TRWEEK 的 DDS，记录格式名为 PCWEEK，          *
A*       它包括报表系统的每周日期项。                                  *
A*********************************************************************
A*
A          R RCWEEK
A            ENUM           5  0       TEXT('EMPLOYEE NUMBER')
A            WEEKNO         2  0       TEXT('WEEK NUMBER OF CURRENT YEAR')
A            EHWRK          4  1       TEXT('EMPLOYEE HOURS WORKED')
A          K ENUM
A          K WEEKNO
```

图 96　数据库文件 TRWEEK（物理文件）的 DDS

```
A*********************************************************************
A* 有关文件：     EMPMST（物理文件）                                  *
A*                TRWEEK（物理文件）                                  *
A*   说     明：这是逻辑文件 EMPL1 的 DDS。它有两个记录格式 EMPREC   *
A*               和 RCWEEK。                                         *
A*********************************************************************
A          R EMPREC                   PFILE(EMPMST)
A          K ENUM
A*
A          R RCWEEK                   PFILE(TRWEEK)
A          K ENUM
A          K WEEKNO
```

图 97　数据库文件 EMPL1（逻辑文件）的 DDS

样板程序 1（使用键字顺序的主文件）

这个例子中，雇员主记录(EMPREC)和每周工作时间记录(RCWEEK)包含在

同一逻辑文 EMPL1 中。EMPL1 文件是主输入文件并且是按键字顺序读的文件。在文件的数据描述规范表中，EMPREC 记录的键字定义为 ENUM(雇员号)字段，RCWEEK 记录的键字定义为 ENUM 和 WEEKNO 字段，它是一个复合键字。

```
*********************************************************************
* 程 序 名:     YTDRPT1                                              *
* 有关文件:     EMPL1（逻辑文件）                                      *
*               PRINT（打印文件）                                     *
* 说    明:     这个程序给出用键字顺序处理记录的例子。它打印出    *
*               每个雇员的资料及每周工时。                            *
*********************************************************************

FPRINT     O    F   80        PRINTER
FEMPL1     IP   E           K DISK

*  对每个记录分配记录标识指示器，用来控制不同记录类型的处理。

IEMPREC         01
I*
IRCWEEK         02
I*

*  由于是用键字顺序读 EMPL1 文件，对有效的雇员号，在 RCWEEK 中的
*  ENUM 必须与最后一次读的 EMPREC 中的 ENUM 相同。把 EMPREC 中的
*  ENUM 保存到 EMPNO 中，并与从读到的 RCNEEK 记录中的 ENUM 比较来
*  检查。如果 ENUM 有效，*IN12 为 ON，它用来控制打印 RCWEEK 记录。

C                   SETOFF                                      12
C   01              MOVE      ENUM          EMPNO             5 0
C*
C                   IF        (*IN02='1') AND (ENUM=EMPNO)
C                   SETON                                       12
C                   ENDIF

OPRINT     H    1P                    2  6
O                                         40 'EMPLOYEE WEEKLY WORKING '
O                                         52 'HOURS REPORT'
```

```
0          H    01                   1
0                                        12 'EMPLOYEE: '
0                         ENAME          32
0          H    01                   1
0                                        12 'SERIAL #: '
0                         ENUM           17
0                                        27 'DEPT: '
0                         EDEPT          30
0                                        40 'TYPE: '
0                         ETYPE          41
0          H    01                   1
0                                        20 'WEEK #'
0                                        50 'HOURS WORKED'
0          D    12                   1
0                         WEEKNO         18
0                         EHWRK       3  45
```

图 98　按键字顺序处理，例 1

样板程序 2（键字顺序的 READ）

这个例子和前一个例子相同，只是定义 EMPL1 为全过程文并且读文件是由 READ 操作码完成的。

```
*****************************************************************
* 程 序 名：    YTDRPT2                                          *
* 有关文件：    EMPL1（逻辑文件）                                *
*               PRINT（逻辑文件）                                *
*  说    明：   这个程序给出用读操作码的方法处理记录的例子。它打 *
*               印出每个雇员的资料及每周工时。                   *
*****************************************************************

FPRINT     O    F   80        PRINTER
FEMPL1     IF   E           K DISK

*  同一个文件中有两个记录（EMPREC 和 RCWEEK），对每个记录分配
*  一个记录标识指示器。用来控制处理不同类型的记录。对全过程文
*  件不能规定控制级和匹配字段。
```

```
IEMPREC         01
I*
IRCWEEK         02
I*
```

```
*  READ 从 EMPL1 中读一条记录。在 58-59 列规定一个文件末指示器。如果
*  到文件末，指示器 99 为 ON。程序转到 EOFEND 处，且处理文件末例程。
```

```
C                   SETOFF                                      12
C                   READ       EMPL1                                99
C   99              GOTO       EOFEND
C*
C   01              MOVE       ENUM           EMPNO            5 0
C*
C                   IF         (*IN02='1') AND (ENUM=EMPNO)
C                   SETON                                       12
C                   ENDIF
```

```
*  因为 EMPL1 定义为全过程文件。在处理完最后一条记录后，设*INLR
*  为 ON 来结束程序。
```

```
C     EOFEND        TAG
C   99              SETON                                       LR
```

```
OPRINT     H    1P                    2  6
O                                         40 'EMPLOYEE WEEKLY WORKING '
O                                         52 'HOURS REPORT'
O          H    01                    1
O                                         12 'EMPLOYEE: '
O                        ENAME            32
O          H    01                    1
O                                         12 'SERIAL #: '
O                        ENUM             17
O                                         27 'DEPT: '
O                        EDEPT            30
O                                         40 'TYPE: '
O                        ETYPE            41
```

```
0          H    01                    1
0                                          20 'WEEK #'
0                                          50 'HOURS WORKED'
0          D    12                    1
0                         WEEKNO           18
0                         EHWRK       3    45
```

图 99　　键字顺序处理，例 1

样板程序 3（匹配记录技术）

在这例子中，TRWEEK 文件定义为次输入文件。EMPREC 和 RCWEEK 记录作为匹配记录处理，两个记录中的 ENUM 字段分配了匹配级值 M1。

给记录分配了记录标识指示器 01 和 02 来控制处理不同类型的记录。

```
**********************************************************************
* 程 序 名：     YTDRPTS                                              *
* 有关文件：     EMPMST（物理文件）                                     *
*           TRWEEK（物理文件）                                        *
*           PRINT      （打印文件）                                    *
* 说    明：     这个程序给出用匹配记录方式处理记录的例子。它打印   *
*                出每个雇员的资料，每周工时及超时总数。              *
**********************************************************************

FPRINT     O    F   80        PRINTER
FEMPMST    IP   E           K DISK
FTRWEEK    IS   E           K DISK

IEMPREC         01
I                                        ENUM           M1
IRCWEEK         02
I                                        ENUM           M1

C   01              Z-ADD     0            TOTHRS          5 1
C   01              Z-ADD     0            TOTOVT          5 1
C   01              SETOFF                                   12
C*
C   MR              IF        (*IN02='1')
```

```
C                   ADD       EHWRK          TOTHRS
C     EHWRK         SUB       ENHRS          OVTHRS             4 111
C   11              ADD       OVTHRS         TOTOVT
C                   SETON                                         12
C                   ENDIF

OPRINT     H    1P                      2  6
O                                              50 'YTD PAYROLL SUMMARY'
O          D    01                      1
O                                              12 'EMPLOYEE: '
O                         ENAME                32
O          D    01                      1
O                                              12 'SERIAL #: '
O                         ENUM                 17
O                                              27 'DEPT: '
O                         EDEPT                30
O                                              40 'TYPE: '
O                         ETYPE                41
O          D    02 MR                   1
O                                               8 'WEEK #'
O                         WEEKNO               10
O                                              32 'HOURS WORKED = '
O                         EHWRK          3     38
```

```
*   如果*IN01 为 ON 且没有找到匹配记录（即对这个雇员没有找到
*   RCWEEK 记录）。处理 2 行详细输出。在这种情况下，总计字段
*   （TOTHRS 和 TOTOVT）为零。

O          D    01NMR                   1
O                                              70 'YTD HOURS WORKED = '
O                         TOTHRS         3     78
O          D    01NMR                   1
O                                              70 'YTD OVERTIME HOURS = '
O                         TOTHRS         3     78
```

```
*   在做详细计算之前处理这 2 个总计行。这样，如果规定的指示器为 ON，
*   打出最后读出的雇员的总计字段（TOTHRS 和 TOTOVT）。
```

```
0          T    01 12                  1
0         OR    LR 12
0                                            70 'YTD HOURS WORKED = '
0                      TOTHRS        3       78
0          T    01 12                  1
0         OR    LR 12
0                                            70 'YTD OVERTIME HOURS = '
0                      TOTOVT        3       78
```

图 100　键字顺序处理，例 3

15.4.3 用键字随机处理

要用键字随机方式处理，在计算规范表中 CHAIN 操作的因子 1 上要给出标识要读记录键字的检索自变量。4.3.4.3.1 中图 102 给出了一个用键字随机处理一个外部描述 DISK 文件的例子。在明细计算或在总计计算时都可以从文件中读指定的记录。

用键字随机方式可以处理全过程输入和更新文件。

对外部描述文件，文件描述规范表的 34 列必须有一个 K， K 指出该文件是根据键字建立的访问路径来处理的。

文件的数据描述规范(DDS)说明了有键字值的字段(键字字段)。文件描述规范表的 35 列必须是空格。

对一个程序描述文件，必须把它定义为一个索引文件，（35 列上是 I）文件描述规范表的 34 列上必须是 A,D,G,P,T 或 Z。29～33 列上标识键字字段的长度，KEYLOC 键字标识键字字段的开始位置，必须用数据描述规范建立程序描述输入文件的访问路径。（见 4.3.3.1 中“索引文件”）。

15.4.2.1 用键字随机处理的例子

下面是如何使用键字随机处理的例子。4.3.4.2.1 中图 95 和图 101 表示的是由程序 EMSTUPD(图 102)使用的物理文件的 DDS 描述。

```
A*********************************************************************
A*  有关程序：EMSTUPD                                                *
A*  说     明： 这是物理文件 CHANGE 的 DDS。它有 CHGREC 记录格式，   *
A*               它是用来更新 EMPMST 文件的新数据。                   *
A*********************************************************************
A*
```

```
A          R CHGREC
A            ENUM            5  0       TEXT('EMPLOYEE NUMBER')
A            NNAME          20          TEXT('NEW NAME')
A            NTYPE           1          TEXT('NEW TYPE')
A            NDEPT           3  0       TEXT('NEW DEPARTMENT')
A            NNHRS           3  1       TEXT('NEW NORMAL WEEK HOURS')
A          K ENUM
```

图 101　数据库文件 CHANGE（物理文件）的 DDS 描述

样板程序

在这个例子中，EMPMST 文件定义为更新全过程文件。用键字处理更新文件 CHANGE。DDS 为每个外部描述文件(EMPMST 和 CHANGE)定义了 ENUM 字段为键字字段，读/更新处理都由计算规范表中指定的操作控制。

```
*****************************************************************
*  程 序 名:    EMSTUPD                                          *
*  有关文件:    EMPMST（物理文件）                                 *
*           CHANGE（物理文件）                                    *
*  说    明:  它给出用键字随机处理记录的方法。用 CHAIN 操作码来   *
*             做。物理文件 CHANGE 包括所有修改 EMPMST 文件用的记  *
*             录。它的记录格式为 CHGREC，其中有些字段是空格，     *
*             在这种情况下，不做修改。                           *
*****************************************************************

FCHANGE    IP   E           K DISK
FEMPMST    UF   E           K DISK

*  当从 CHANGE 读每个记录时，用雇员号（ENUM）做检索自变量来
*  连接 EMPMST 文件。如果没找到相应的记录，*IN03 为 ON，当用
*  无效的 ENUM 查询 CHGREC 时，发生这种情况。

C      ENUM          CHAIN     EMPREC                          03
C    03              GOTO      NEXT
C      NNAME         IFNE      *BLANK
C                    MOVE      NNAME          ENAME
C                    ENDIF
```

```
C      NTYPE          IFNE       *BLANK
C                     MOVE       NTYPE          ETYPE
C                     ENDIF
C      NDEPT          IFNE       *ZERO
C                     MOVE       NDEPT          EDEPT
C                     ENDIF
C      NNHRS          IFNE       *ZERO
C                     MOVE       NNHRS          ENHRS
C                     ENDIF
C                     UPDATE     EMPREC
C*
C      NEXT           TAG
```

图 102 用键字随机处理的外部描述文件

15.4.4 界限内的顺序处理

用一个记录地址文件的界限内顺序处理是在文描述规范表的 28 列上的 L 来指定，它对有键字访问的文件有效。

可以为一个主文件，次文件或全过程文件的输入或更新指定界限内顺序处理。该文件可以是外部描述文件也可以是程序描述文件（索引文件）。文件应该用升序键字。

要从记录地址文件去处理界限内顺序，程序要读：

• 记录地址文件的一个界限记录。

• 文件中键字大于或等于界限中低记录键字，小于或等于界限中高记录键字的记录。如果记录地址文件中的两个界限相等，仅接收规定键字的记录。

程序重复这个过程，在到记录地址文件抵达文件末时为止。

15.4.4.1 界限内顺序处理的例子

图 103 给出一个界限内顺序处理索引文件的例子。

图 105 程序描述文件的同样例子。

图 95(4.3.4.2.1 中)给出在程序 ESWLIM1(图 103)和 ESWLIM2(图 105)中的物理文件的 DDS 描述。

样板程序 1（界限内顺序处理）

EMPMST 是由记录地址文件 LIMITS 界限内顺序处理（28 列上有 L）。记录地址文件的每个界限由 EMPMST 文件中记录的低和高雇员号组成。由于雇员号键字字段(ENUM)是 5 个数字长，每个界限由两倍 5 个数字长键字组成。（注

意 ENUM 是压缩格式，因此，它要用 3 位代替 5）。

```
*****************************************************************
* 程 序 名：     ESWLIM1                                          *
* 有关文件：     EMPMST（物理文件）                               *
*          LIMITS（物理文件）                                    *
*             PRINT    (打印文件)                                *
* 说    明：    这个程序给出有限顺序处理下标文件的例子，也打印出 *
*             雇员号在 LIMLTS 文件限制之内的雇员资料。          *
*****************************************************************
FLIMITS    IR   F    6     3  DISK    RAFDATA(EMPMST)
FEMPMST    IP   F   28L    3PIDISK    KEYLOC(1)
FPRINT     O    F   80        PRINTER
  *  必须有输入规范，用来说明程序描述文件 EMPMST 中的记录。

IEMPMST    NS  01
I                              P     1     3 0ENUM
I                                    4    23  ENAME
I                                   24    24  ETYPE
I                              P    25    26 0EDEPT

 *  当用每组限制处理 EMPMST 时，打印相应的记录。当记录地址文件
 * LIMITS 到了文件末时，EMPMST 文件处理完毕。

OPRINT     H    1P                   1
0                                         12 'SERIAL #'
0                                         22 'NAME'
0                                         45 'DEPT'
0                                         56 'TYPE'
0          D    01                   1
0                         ENUM            10
0                         ENAME           35
0                         EDEPT           45
0                         ETYPE           55
```

图 103　外部描述文件的界限内顺序处理

图 104 给出程序 ESWLIM2（图 105）使用的记录地址界限文件的 DDS 描述。

```
A*****************************************************************
A* 有关程序：ESWLIM                                                *
A* 说     明：这是物理文件 LIMITS 的 DDS，记录格式名为 LIMIT。      *
A*****************************************************************
A
A          R LIMIT
A            LOW            5  0
A            HIGH           5  0
```

图 104 记录地址文件 LIMITS（物理文件）的 DDS

这个程序的作用同上一个程序，不同点只是物理文件 EMPMST 定义为程序描述文件。

```
     ****************************************************************
     * 程 序 名：     ESWLIM2                                        *
     * 有关文件：     EMPMST     （物理文件）                         *
     *          LIMITS     （物理文件）                              *
     *          PRINT      （打印文件）                              *
     * 说     明：    这个程序给出有限顺序处理外部描述文件的例子，也    *
     *             打印出雇员号在 LIMITS 文件限制之内的雇员信息。     *
     ****************************************************************

FLIMITS    IR   F    6     3  DISK    RAFDATA(EMPMST)
FEMPMST    IP   E     L     K DISK
FPRINT     O    F   80        PRINTER

     *  对外部描述文件输入规范是可选的，这里，定义*IN01 为记录标识指示器，
     *  用 EMPREC 来控制处理记录。

IEMPREC         01

OPRINT     H    1P                    1
O                                        12 'SERIAL #'
O                                        22 'NAME'
O                                        45 'DEPT'
```

```
0                                        56 'TYPE'
0          D     01                  1
0                      ENUM              10
0                      ENAME             35
0                      EDEPT             45
0                      ETYPE             55
0*
```

图 105　程序描述文件的界限内顺序处理

15.4.5 相对记录号处理

用相对记录号随机输入或更新处理仅适用于全过程文件。用 CHAIN 操作码访问所描述的记录。

相对记录号是记录相对于文件开始的位置。例如，第 1，第 5 和第 7 个记录的相对记录号是 1，5 和 7。

对一个外部描述文件，用相对记录号做输入或更新处理是由文件描述规范表的 34 列上是空格并使用 CHAIN 操作码来决定的，用相对记录号作输出处理是由该文件的文件描述规范表 34 列为空格及用 RECNO 键字来决定的。

用文件描述规范表上 RECNO 键字指出一个数字型字段，它指出加到文件中的新记录的相对记录号。RECNO 字段一定要定义为小数位是零的数字型字段。字段的长度一定要足够大能放得下该文件的最大记录号。如果新记录是用输出规范表或一个 WRITE 操作放到文件中，那就必须说明 RECNO 字段。

当用相对记录号给文件增加或更新记录时，该记录在成员中必须有一位置。对更新记录，那个位置可以是一个有效的已存在的记录；对一个新的记录来说那个位置可以是一个已删除的记录。

可以用 CL 命令 INZPFM 来初始化使用相对记录号的记录。对检索操作或定位文件的操作（例如，SETLL，CHAIN，READ 操作），在 RECNO 字段中放置当前的相对记录号。

15.5 有效的文件操作

图 106 给出了用键字处理的 DISK 文件有效的文件操作码。图 107 给出了不用键字方式处理的 DISK 文件允许的操作码。这些操作对外部描述 DISK 文件和程序描述的 DISK 文件都是有效的。

运行程序之前，可以用一个文件覆盖另一个文件。在程序中用顺序文件覆盖外部描述的键字文件（使该文件作为一个顺序文件处理），也可以在程序中用键字文件覆盖另一个键字文件，它提供的键字字段是复合的。例如，覆盖文件的键字字段不能比程序中说明的那个键字字段短。

注意：删除一条数据库记录时，物理记录被标记为删除。如果用初始化物理文件成员（INZPFM）命令做删除记录的初始化时，文件中的记录就被删除了。记录一旦被删除，就不能再读了，但可用相对记录号定位这条记录然后再写它的内容。

文件描述规范表中位置					计算规范表中位置
17	18	20	28(1)	34(2)	26-35
I	P/S			K/A/P/D /T/Z	/ CLOSE, FEOD, FORCE
I	P/S	A		K/A/P/D /T/Z	/WRITE, CLOSE, FEOD, FORCE
I	P/S		L	K/A/P/D /T/Z	/CLOSE, FEOD, FORCE
U	P/S			K/A/P/D /T/Z	UPDATE, DELETE, CLOSE, FEOD, FORCE
U	P/S	A		K/A/P/D /T/Z	/UPDATE, DELETE, WRITE, CLOSE, FEOD, FORCE
U	P/S		L	K/A/P/D /T/Z	/UPDATE, CLOSE, FEOD, FORCE
I	F			K/A/P/D /T/Z	/READ, READE, READPE, READP, SETLL, SETGT, CHAIN, OPEN, CLOSE, FEOD
I	F	A		K/A/P/D /T/Z	/WRITE, READ, READPE, READE, READP, SETLL, SETGT, CHAIN, OPEN, CLOSE, FEOD
I	F		L	K/A/P/D /T/Z	/READ, OPEN, CLOSE, FEOD
U	F			K/A/P/D /T/Z	/READ, READE, READPE, READP, SETLL, SETGT, CHAIN, UPDATE, DELETE, OPEN, CLOSE, FEOD
U	F	A		K/A/P/D /T/Z	/WRITE, UPDATE, DELETE, READ, READE, READPE, READP, SETLL, SETGT, CHAIN, OPEN, CLOSE, FEOD
U	F		L	K/A/P/D /T/Z	/READ, UPDATE, OPEN, CLOSE, FEOD
O	Blank	A		K/A/P/D /T/Z	/WRITE (add new records to a file), OPEN, CLOSE, FEOD

0	Blank			K/A/P/D/T/Z	/WRITE (initial load of a new file)(3), OPEN, CLOSE, FEOD
注：⑴必须在 28 列说明一个 L 来指出用记录地址文件界限内顺序处理一个输入或更新文件。					
注：⑵外部描述文件在 34 列上需要指定 K；程序描述文件在 34 列上需要指定 A,P,G,D,T 或 Z 而在 35 列上指定 I。					
注：⑶往一个新文件中初始装入记录不用在 20 列上指定 A。如果在 20 列上指定了 A 那么在输出规范表上必须说明 ADD，该文件一定要用 OS/400 生成文件命令生成。					

图 106　键字处理方式的有效文件操作(键字随机处理、键字顺序处理、界限内顺序处理)

文件描述规范表中位置				计算规范表中位置	
17	18	20	34	44-80	26-35
I	P/S		Blank		CLOSE, FEOD, FORCE
I	P/S		Blank	RECNO	CLOSE, FEOD, FORCE
U	P/S		Blank		UPDATE, DELETE, CLOSE, FEOD, FORCE
U	P/S		Blank	RECNO	UPDATE, DELETE, CLOSE, FEOD, FORCE
I	F		Blank		READ, READP, SETLL, SETGT, CHAIN, OPEN, CLOSE, FEOD
I	F		Blank	RECNO	READ, READP, SETLL, SETGT,
U	F		Blank		READ, READP, SETLL, SETGT, CHAIN, UPDATE, DELETE, OPEN, CLOSE, FEOD
U	F		Blank	RECNO	READ, READP, SETLL, SETGT, CHAIN, UPDATE, DELETE, OPEN, CLOSE, FEOD
U	F	A	Blank	RECNO	WRITE (overwrite a deleted record), READ, READP, SETLL, SETGT, CHAIN, UPDATE, DELETE, OPEN, CLOSE, FEOD
I	R		A/P/G/D/T/Z/Blank(1)		OPEN, CLOSE, FEOD
I	R		Blank(2)		OPEN, CLOSE, FEOD

O	Blank	A	Blank	RECNO	WRITE(3) (add records to a file), OPEN, CLOSE, FEOD
O	Blank		Blank	RECNO	WRITE(4) (initial load of a new file), OPEN, CLOSE, FEOD
O	Blank		Blank	Blank	WRITE (sequentially load or extend a file), OPEN, CLOSE, FEOD
注：⑴如果一个记录地址界限文件的 34 列是空格，则记录地址文件的键字格式是和正处理的文件的键字格式相同的。					
注：⑵有相对记录号的记录地址文件在 35 列上需要指定一个 T。					
注：⑶有相对记录号的 RECNO 字段必须在 WRITE 操作前设置，或者在输出规范表上说明 ADD。					
注：⑷往一个新文件中初始装入记录不用在 20 列上指定 A。如果在 20 列上指定了 A 那么在输出规范表上必须说明 ADD，该文件一定要用 OS/400 生成文件命令生成。。					

图 107　无键字处理方式的有效文件操作(顺序处理、相对记录号随机处理和连续处理)

15.6 使用落实控制

这一部分讲述如何用落实控制处理一组文件操作。使用落实控制，可以确保文件操作的两个结果之一：

①全部文件操作都是成功的（落实操作）

②全部文件操作都是无效的（取消操作）

用这种方式，可以把一组操作当作一个单元来处理。

使用落实控制，要做：

在 AS/400 上：

1.使用落实控制的准备：用 CL 命令 CRTJRN（生成日志），CRTJRNRCV（生成日志接收器）和 STRJRNPF（日志物理文件）。

2.通知 AS/400 启动和结束落实控制的时间：用 CL 命令 STRCMTCTL（启动：落实控制）和 ENDCMTCTL（结束落实控制）。关于这些命令的介绍见《CL 参考》。

在 RPG 程序中：

1.在要做落实控制文件的文件描述规范表上指出落实控制(COMMIT)

2.用 COMMIT（落实）操作码实施对在落实控制下文件的一组修改，或者用 ROLBK（取消）操作码撤消对在落实控制下文件的一组修改。

注：落实控制只适用于数据库文件。

15.6.1 启动和结束落实控制

CL 命令 STRCMTCTL 通知系统启动落实控制。

LCKLVL（锁定级）参数允许选择落实控制下记录上锁的级别。有关上锁级别的详细介绍见 4.3.6.1.1 中“控制上锁”及《CL 程序设计》。

可以建立落实控制条件，某种意义上来说它决定了在运行时怎样处理落实控制文件。详细资料见 4.3.6.4 中《指定条件落实控制》。

当完成有COM1T操作的一组修改时，允许指定一个标号来标识该组的结束。如果发生一个异常作业结束，这个标号写到一个文件，信息队列或数据区中，这样能知道哪一组修改是最后成功完成，可在 STRCMTCTL 命令中指定这个文件，信息队列或数据区。

在调用说明了落实控制处理文件的程序之前，要发出 STRCMTCTL 命令。如果在发出 STRCMTCTL 命令之前调用了一个程序，该程序说明了落实控制文件，则文件的打开将失败。

CL 命令 ENDCMTCTL 通知系统，活动组或作业已经结束了落实控制下的文件处理。有关 STRCMTCTL 和 ENDCMTCTL 命令的详细资料见《CL 参考》。

15.6.1.1 落实控制锁

在 STRCMTCTL 命令中，指定一个上锁级别，LCKLVL(*ALL)，LCKLVL(*CHG)或 LCKLVL(*CS)。当你的程序在落实控制下操作，并且对一个文件的记录已在落实控制下进行了输入或输出处理，记录由落实控制加锁：

• 程序可以访问该记录。

• 在你活动组或作业内的其他程序，如使用这个在落实控制下的文件，可以读记录。如果该文件是一个共享文件，第二个程序也可以更新该记录。

• 在你活动组或作业内的其他程序没有落实控制下的文件不能读或更新记录。

• 在另外的活动组或作业中的其他程序，用落实控制下的文件，如果指定 LCKLVL(*CHG)，那么可以读记录，如果指定了 LCKLVL(*ALL)就不能读该记录。用这两种上锁级别，第二个程序不能更新记录。

•没有在落实控制下文件的其它程序，而且程序又不在你的活动组或作业中，这个程序可以读但是不能更新记录。

落实控制锁和平常的锁不一样，它在 LCKLVL 中指定并只能由 COMMIT 和 ROLBK 操作打开。

COMMIT 和 ROLBK 操作打开记录上的锁。UNLOCK 操作不能打开由落实控制锁上的记录。有关锁定级的详细介绍见《CL 参考》。

在 COMMIT 和 ROLBK 操作之前，能被落实控制锁定的数且是有限的。详细资料见《备份和恢复》，SC41-3305。

注：一个读操作（不为更新）将锁住落实控制下的记录，在同样情况下，SETLL 和 SETGT 操作也将给记录上锁。

15.6.1.2 落实控制范围

当用 STRCMTCTL 命令启动落实控制时，系统建立一个落实定义。落实定义包括作业中落实控制下修改的资料信息。落实定义只适用于发出 STRCMTCTL 命令之后的作业并且在发出 ENDCMTOTL 命令时结束。

落实定义范围指出了作业中哪些程序要使用落实控制。落实定义范围在活动组级或作业级。

缺省的落实定义范围是发出 STRCMTCTL 命令的程序所在的活动组，也就是活动组级。只在这个活动组中运行的程序才能使用落实定义。OPM 程序将使用*DFTACTGRP 落实定义。ILE 程序用与之有关的活动组。

你可以在 STRCMTCTL 命令的落实范围(CMTSCOPE)参数上指定落实定义的范围。ILE 落实控制范围的详细介绍，参考《数据管理范围》(ILE 概念)，也可以查阅《数据管理》。

15.6.2 规定落实控制文件

要指定一个磁盘文件在落实控制下运行，在文件描述规范表的键字字段上输入 COMMIT。

当一个程序描述文件被落实控制时，它仅对由程序对文件所做的输入和输出操作有用。落实控制不对其他的非输入和输出操作有用，也不对程序中没有规定做落实控制的文件的 I/O 操作有用。

当这个程序访问共享文件时，要么所有的程序都要指定落实控制，要么都不指定。

15.6.3 使用 COMMIT 操作

COMMIT 操作告诉系统，已完成了在落实控制下文件的一组修改。ROLBK 操作撤消在落实控制下文件当前的一组修改。怎样指定操作码和操作码的内容介绍见《ILE　RPG/400 参考》。

如果系统失败，那么隐含的是发出一个 ROLBK 操作，使用在 COMMIT 操作中因子 1 上的标号来检查最后成功完成的一组修改的标识，并通知在 STRCMTCTL 命令上指出的目标。

当作业或活动组结束，或者发出 ENDCMTOTL 命令时，OS/400 隐含发出 ROLBK，它撤消从最后一个 ROLBK 或 COMMIT 操作以来的任何修改。要保证全部文件操作生效，在结束在落实控制下作业或活动组之前发出一个 COMMIT 操作。

OPEN 操作允许对文件进行输入和输出操作，而 CLOSE 操作停止对一个文件的输入和输出操作。但是，打开和关闭操作不影响 COMMIT 和 ROLBK 操作；COMMIT 或 ROLBK 操作影响一个文件，即使在文件关闭之后也一样。

例如，程序包括下列步骤：

1.发出 COMMIT（文件已经在落实控制下打开）。

2.打开一个规定在落实控制下的文件。

3.对这个文件执行某些输入和输出操作。

4.关闭该文件。

5.发出 ROLBK。

即使在第四步关闭了该文件，在第三步所做的修改在第五步被 ROLBK 操作撤消了。TOLBK 操作可以从同一作业或活动组的其他程序发出。

一个程序中的所有文件不一定都要在落实控制下处理，如果那样做可能对执行有不利的影响。COMMIT 和 ROLBK 操作对不在落实控制下的文件没有影响。

注：当多个设备连接到一个应用程序时，程序中使用的文件实行落实控制，COMMIT 或 ROLBK 操作是基于文件操作而不是通过设备。数据库可能由于 COMMIT 做不完全的更新，撤消了其它用户已经完成的修改。保证不发生这些情况是程序员的责任。

15.6.3.1 落实控制的例子

这个例子解释了在落实控制下程序所用的规范表和 CL 命令。

要准备使用落实控制，发出下列 CL 命令：

1.CRTJRNRCV JRNRCV(RECEIVER)

这个命令生成一个名为 RECEIVER 的日志接收器。

2.CRTJRN JRN(JOURNAL)JRNRCV(RECEIVER)

这个命令生成一个名为 JOURNAL 的日志并连接名为 RECEIVER 的日志接收器。

3.STRJRNPF FILE(MASTER TRANS) JRN(JOVRNAL)

这个命令把文件 MASTER 和 TRANS 的日志项记到日志 JOURNAL 中。

在程序中，为文件 MASTER 和 TRANS 指出 COMMIT。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords++++++++++++++++++++++++
FMASTER    UF   E      K      DISK    COMMIT
FTRANS     UF   E      K      DISK    COMMIT
F*
```

```
*..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+... *
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....
C                   :
C                   :
C*
C*  使用 COMMIT 操作，如果成功，做完整的一组操作。如果不成功，退回
C*  所有的修改。
C*
C                   UPDATE    MAST_REC                          90
C                   UPDATE    TRAN_REC                          91
C                   IF        *IN90 OR *IN91
C                   ROLBK
C                   ELSE
C                   COMMIT
C                   ENDIF
```

图 108　落实控制的例子

要在落实控制下操作程序，发出命令：

1.STRCMTCTL LCKLVL(*ALL)

这个命令启动带有最高级锁定的落实控制。

2.CALL　　REVISE

这个命令调用 REVISE 程序。

3.ENDCMTCTL

这个命令结束落实控制，并产生一个隐含的 ROLL BACK 操作。

15.6.4 规定有条件的落实控制

可以编写一个程序来控制打开一个运行时做落实控制的文件。通过完成有条件落实控制，可不用写出两套相同的程序：一套在落实控制下执行，另一套不在。

COMMIT 键字有一个可选参数，它允许你指定有条件的落实控制。在这个文件的文件描述规范表的键字区输入 COMMIT 键字。ILE　RPG/400 编译程序定义一个一个字节的与参数中规定的同名字符型字段。如果参数为 1，文件就在落实控制下运行。

COMMIT 键字参数必须在文件打开之前设置。可以在调用程序时传递一个值来设置，也可以在程序中直接设为“1”。

对于共享式打开，如果文件已经打开了，即使 COMMIT 键字参数设为 1 也没有作用。

图 109 是有条件的落实控制例子。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords++++++++++++++++++++++
FMASTER    UF   E      K      DISK    COMMIT(COMITFLAG)
FTRANS     UF   E      K      DISK    COMMIT(COMITFLAG)

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....
C*  如果 COMMITFLAG=‘1’，文件在落实控制下打开，否则不是。
C     *ENTRY        PLIST
C                   PARM                    COMITFLAG
C                   :
C                   :
C*
C*  使用 COMMIT 操作，如果成功，做完整的一组操作。如果不成功，退回
C*  所有的修改。仅可在文件是在落实控制打开时，才能用 COMIT 和 ROLBK
C*  操作（也即 COMITFLAG=‘1’）。
C*
C                   UPDATE    MAST_REC                          90
C                   UPDATE    TRAN_REC                          91

C                   IF        COMITFLAG = '1'

C                   IF        *IN90 OR *IN91
C                   ROLBK
C                   ELSE
C                   COMMIT
C                   ENDIF

C                   ENDIF
C*
```

图 109　使用条件落实控制下的例子

15.6.5 在程序周期中的落实控制

落实控制是给全过程文件用的，它的输入输出是在程序员控制下。对主文件和次文件不使用落实控制。它的输入和输出是由 RPG 程序周期控制的，原因如

下：

- 在程序中不能为最后总计输出发出一个 COMMIT 操作。
- 周期内的程序从一个上锁记录条件中恢复是困难的。
- 级别指示器不能由 ROLBK 操作更新设置。
- 一个 ROLBK 操作以后，处理匹配记录可能产生一个顺序错误。

15.7 DDM 文件

ILE　RPG/400 通过分布式数据管理(DDM)访问远程系统文件。DDM 允许一个系统上的应用程序使用存储在远程系统上的数据库文件。ILE　RPG/400 程序中不需要特殊的语句支持 DDM 文件。

DDM 文件由用户或程序在本地（源）系统中建立带有*FILE 类型目标的文件标识一个建立在远程（目的）系统中的文件。DDM 文件提供本地系统确定远程系统位置及访问源文件数据的必要信息。关于如何使用和建立 DDM 文件的详细资料，参考《分布式数据管理》。

15.7.1 使用 3.1 版本前的 DDM 文件

如果正在使用 3.1 版本以前的 DDM 文件，在 READE，READPE 或者 SETLL 操作期间或在记录地址文件做界限内顺序处理期间，不在数据管理级上做键字比较，而用*HEX 对照顺序来比较键字。

当建立访问路径上的字段内容和字段的实际内容不同时得到不同的结果。导致不同键字比较的某些 DDS 特点如下：

- 文件中规定 ALTSEQ。
- 键字字段 ABSVAL，ZONE，UNSIGNED 或 DIGIT 键字。
- 变长，日期，时间或时间标记键字字段。
- 文件的 STRSEQ 不是*HEX。

注意：如果数字型字段的符号与系统提供的符号不同时，键字比较也会不同。

在 READE，READPE 或 SETLL 操作或由记录地址文件界限内顺序处理期间，V3R1 版前的 DDM 文件上不做数据管理级的第一次键字比较，则会发出一条 RNI2002 信息。

注：可能没找到记录(SETLL，CHAIN，SETGT，READE，READPE)的 I/O 操作会比 V3R1 版本前慢一些。

第十六章　访问外部连接的设备

用设备文件访问 RPG 外部连接设备，设备文件是提供访问外部连接硬件（诸如打印，磁带机，磁盘驱动器，显示站和用通讯线连接的其他系统）的文件。

这一章介绍如何用 RPG 设备名 PRINTER，SEQ 和 SPECIAL 来访问外部连接设备。对于显示工作站和 ICF 设备的资料见 17 章 4.5“使用 WORKSTN 文件”。

16.1 设备文件类型

在程序能对系统上设备进行读写操作之前，向系统说明设备硬件性能的设备描述文件必须在设备配置时建立。设备文件规定了如何使用设备。通过参考指定的设备文件，RPG 程序用向系统描述设备的方法使用设备。设备文件从 RPG 程序中组织数据输出到设备上，或者从设备组织数据输入到 RPG 程序中。

用表 14 列出的设备文件访问相关的外部连接设备。

表 14　AS/400 设备文件，有关的 CL 命令和 RPG 设备名

设备文件	相关的外部连接设备	CL 命令	RPG 设备名
打印机文件 PRINTER	对打印设备提供访问和描述打印输出格式	CRTRRTF CHGPRTF OVRPRTF	
磁带文件	对储存在磁带设备上的数据文件进行访问	CRTTAPF CHGTAPF OVRTAPF	SEQ
软盘文件	访问储存在软盘设备上的数据文件	CRTDKTF CHGDKTF OVRDKTF	DISK
显示文件 WORKSTN	访问显示设备	CRTDSPF CHGDSPF OVROSPF	
ICF 文件 WORKSTN	允许一个系统中的程序与同一系统或另外系统中的程序通信	CRTICFF CHGICFF OVRICFF	

16.2 访问打印设备

ILE　RPG/400程序中的PRINTER文件与AS/400系统中的打印机文件相联：

打印文件允许打印输出文件。这一节讲述如何在 ILE　RPG/400 程序上规定和应用打印机文件。

16.2.1 规定 PRINTER 文件

为表明程序要访问打印机文件，在文件描述规范表上指定 PRINTER 设备名。每个文件一定要有一个唯一的文件名。每个程序最多允许有八个打印文件。

PRINTER 文件既可以是外部描述文件又可以是程序描述文件。对于外部描述的 PINTER 文件不能使用溢出指示器 OA-OG，OV，取溢出，空行/跳行项和 PRTCTL 键字。外部描述文件的有效的输出规范表登记项见《ILE　RPG/400 参考》。外部描述打印文件的数据描述规范（DDS）的资料见《DDS 参考》。

对一个外部描述的 PRINTER 文件，可以指定 DDS 键字 INDARA。如果要把这个键字用于程序描述的 PRINTER 文件，将收到一个运行时错误。

可以用 CL 命令 CRTPRTF（生成打印文件）来生成一个打印文件（关于 CRTPRTF 命令的详细资料见《CL 参考》）；或者也可用 IBM 提供的文件名。这些文件名的介绍见《数据管理》。

对打印文件有效的文件操作码是 WRITE，OPEN，CLOSE 和 FEOD。这些操作码的完整描述见《ILE　RPG/400 参考》。

16.2.2 处理页溢出

使用 PRINTER 文件的一个重要考虑是页溢出。对于外部描述的打印文件，要对页溢出做出响应，做法如下：（选其一）

• 在文件描述规范表的键字字段上用 OFLIND 键字（溢出指示器）来指定 *IN01 到*IN99 中的一个指示器作为溢出指示器。

•检查打印设备 INFDS 反馈部分中有关行数和页溢出情况，详细资料见《ILE RPG/400 参考》。

• 计算每页的输出行数。

• 在规定做输出的计算规范表 73 和 74 列上指定一个指示器来检测文件异常/错误，或者通过一个能处理文件异常/错误子例程（INFSR）来做。INFDS 有该文件异常/错误的详细资料。关于异常和错误处理方面的资料，见 11 章“处理异常”。

对于一个程序描述或一个外部描述文件，都能在文件描述规范表上使用 OFLIND 键字（溢出指示器）来指定*IN01 到*IN99 中的一个指示器。当打印溢出时，或者在空行或跳行操作时抵达或通过溢出行时，该指示器置为 ON。用该指示器来做条件对溢出做响应。该指示器不能象溢出指示器（*INOA—*INOG，

*INOV）中做的那样限制 RPG 溢出逻辑，你要负责设置该指示器为 OFF。

对程序描述和外部描述文件，文件的 INFDS 中的打印反馈部分，行号和页号是有用的。在文件规范表上指定 INFDS 键字来访问这些信息。在规范表上，在数据结构的 367-368 列上定义行数而在 369-372 列上定义页数。行数和页数字段必须定义为小数位为零的二进制数。因为每次打印机输出操作之后 INFDS 会被更新，这些字段也能用来确定程序中没有行计数逻辑的当前行数。

注：如果一个打印文件被另一个不同设备，象磁盘复盖，INFDS 的打印反馈部分不被更新，并且你的行计数逻辑无效。

对程序描述的 PRINTER 文件，溢出指示器和取溢出逻辑在下面一节介绍。

16.2.2.1 在程序描述文件中使用溢出指示器

当打印完通过一页的最后一行时，溢出指示器(OA-OG,OV)置为 ON。溢出指示器可以用来指定打印在下一页上的行，溢出指示器仅能用于程序描述的 PRINTER 文件，并主要用于限制打印标题行。在文件描述规范表上使用 OFLIND 键字指定溢出指示器并可以用来限制计算规范表（9-11 列）和输出规范表（21-29 列）中的操作。如果没有指定溢出指示器，编译程序给 PRINTER 文件分配一个没用到的溢出指示器。溢出指示器也可在计算规范表上（71-76 列）作为结果指示器来用。

仅在一页上第一次出现一个溢出条件时编译程序设置溢出指示器为 ON。当发生下列条件之一时就存在一个溢出条件：

- 打印的一行通过该溢出行
- 空行操作期间通过了溢出行
- 跳行操作时通过了溢出行

表 15 说明了在文件描述和输出规范表上有没有设置溢出指示器的结果。

下列是在输出规范表上使用溢出指示器的几点考虑：

- 空行通过溢出行时设置溢出指示器为 on。
- 跳行跳过溢出行到了同一页中的任意一行，置溢出指示器为 on。
- 跳行跳过溢出行到新页中的任意一行，不置溢出指示器为 on，除非指出一个 SKIP,由它通过规定的溢出行。
- 当跳到新一页的某行但这行不是由溢出指示器规定的，则在格式推进到新页之后把溢出指示器置为 off。
- 如果说明了跳到新行而打印机正在那行上，不出现跳行。溢出示器也设置成 off，除非该行通过溢出行。
- 当为一个输出打印记录说明了 OR 行时，使用前行的空行和跳行登记项，如果和前行不同，在 OR 行上要填上空行和跳行项。
- 控制级指示器可以和溢出指示器一起用，使每页包含的信息是一个控制组的，见 4.4.2.2.3 中的图 111。

• 对条件限制的一个溢出行，溢出指示器可以出现在 AND 或一个 OR 关系中。对 AND 关系，溢出指示器必须在考虑做溢出行的主说明行上出现。对一个 OR 关系，溢出指示器可以在主说明行上出现也可在 OR 行上出现，但是，仅一个溢出指示器可以和一组输出指示器相联。对一个 OR 关系，仅可以在规定溢出指示器的行上说明条件指示器，用来限制溢出行。

• 如果在 AND 行上使用一个溢出指示器，该行不是一个溢出行。在这种情况下，溢出指示器象其他输出指示器一样对待。

• 当溢出指示器用在一个与记录标识指示器有 AND 关系时，常产生不合需要的结果，因为当发生溢出时，可能不是读进记录的类型。所以，当记录标识指示器不是 on，用溢出和记录标识两个指示器限制的所有行都不能打印。

• 一个溢出指示器可以条件限制一个例外行（17 列为 E），并能条件限制例外记录中的字段。

表 15　设置或不设置一个溢出指示器的结果

文件描述规范表 44-80 列	输出规范表 21-29 列	现　　象
没有登记	没有登记	在溢出时用第一次没用到的溢出指示器做条件限制跳到下一页
没有登记	登　　记	出现编译时错误：从输出规范表中去掉溢出指示器，使用第一次没用的溢出指示器做条件限制在溢出时跳到下一页
登　　记	没有登记	继续打印，无溢出识别
登　　记	登　　记	执行正常溢出

16.2.2.2 在每页上打印标题的例子

图 110 是在每页上打印标题所需编码的例子：由于控制字段(L2 是 on)的改变都在第一页，每个溢出页和每个新页开始。当出现一个溢出时(OA 是 on 并且 L2 为 off)，第一行允许在新页的顶端打印标题(跳到 06)。

仅在新的控制组(L2 是 on)开始时第二行允许在新页上打印标题。这个方法，通过 L2 和 OA 不同时是 on 而做标题的复制。如果在记录上说明了控制字段，由于第一个记录总是产生控制中断(L2 为 on)，所以第二行允许在读入第一个记录之后在第一页上打印标题。

```
*..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+... *
OFilename++DF..N01N02N03Excnam++++B++A++Sb+Sa+.........................
```

```
OPRINT     H    OANL2                  3  6
O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++
O         OR    L2
O                                          8 'DATE'
O                                         18 'ACCOUNT'
O                                         28 'N A M E'
O                                         46 'BALANCE'
O*
```

图 110　在每页上打印标题

16.2.2.3 在每页上打印字段的例子

图 111 是在每页上打印某个字段所必需的编码；在溢出条件或在控制级(L2)改变时，跳到 06。NL2 指示器防止在同一周期中打印和跳行两次。

```
*..  1 ...+...  2 ...+...  3 ...+...  4 ...+...  5 ...+...  6 ...+...  7 ...+... *
OFilename++DF..N01N02N03Excnam++++B++A++Sb+Sa+.........................
OPRINT     D    OANL2                  3  6
O         OR    L2
O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++
O                       ACCT                8
O*
```

图 111　在每页上打印字段

16.2.3 在程序描述文件中使用取溢出例程

在一页上没有足够的位置来打印用溢出指示器限制的明细，总计，例外和标题行时，可以调用取溢出例程。这个例程导致溢出。要决定什么时候用取溢出例程，研究所有可能的溢出情况。通过计算行数和空行数，能推测在每个明细，总计和例外行上出现溢出时的情况。

取溢出例程允许修改基本的 RPG/400 溢出逻辑以防止打印超过一张纸的限度并让你使用一页中尽可能多的行。在一般的程序周期内，编译程序在总计输出以后，立即检查一次，看溢出指示器是否为 on。当说明了取溢出功能时，编译程序在规定了取溢出的每行上检查溢出。

图 112 表示的是取溢出置为 on 或 off 时溢出打印的正常处理过程。

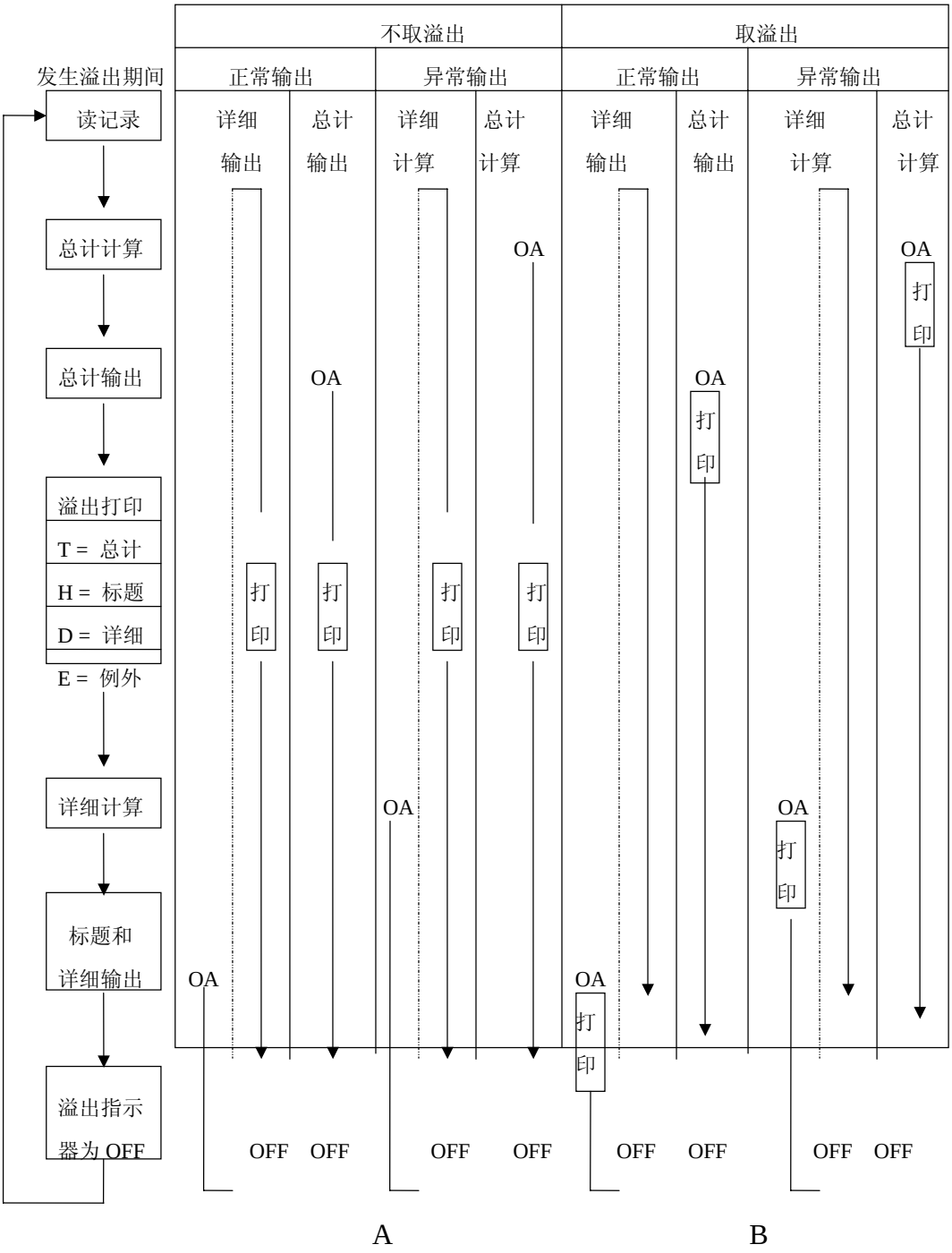


图 112　溢出打印：溢出指示器的设置

A.没有说明取溢出时，总计输出后打印溢出行。当溢出发生时(OA 是 on)无情况，溢出指示器 OA 在整个溢出输出时保持为 on，而当标题和明细输出时以后置为 off。

B.当说明了取溢出时，如果溢出指示器 OA 是 on，在指定了取溢出的输出行之前写溢出行。当 OA 是 on 时，它保持 on 状态直到标题和明细输出后。在溢出

输出时，溢出行不做第二次写出，除非最后一次写溢出行时之后又发生溢出。

16.2.3.1 规定取溢出

对 PRINTER 文件的每个明细，总计或例外行的输出规范表 18 列上用一个 F 来指定取溢出。取溢出例程不自动使格式推进到下一页。

输出期间，检测输出行上的条件指示器来决定该行是否写出。如果该行是要写的而且在 18 列上说明了 F，编译程序检测溢出指示器是否为 on。如果是 on，那么调用取溢出例程而且发出下列的操作：

1.仅检测为输出说明了取溢出文件的溢出行。

2.写出用溢出指示器限制的所有总计行。

3.当跳行数小于打印机当前所在行而这行又是由溢出指示器规定的行，则格式推进到新页。

4.写出用溢出指示器限制的标题，明细和例外行。

5.写出溢出例程所取的行。

6.写出其余的由程序周期写的明细和总计行。

如果溢出例程用于 OR 关系上的每个记录，那么每个 OR 行上的 18 列必须含有 F。如果在同一个说明行中的 21--29 列上指定了溢出指示器就不能使用取溢出。如果是这种情况，不取溢出例程。

16.2.3.2 指定取溢出的例子

图 113 表明取溢出的应用。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
OFilename++DF..N01N02N03Excnam++++B++A++Sb+Sa+.............................
OPRINTER   H    OA                    3 05
O.............N01N02N03Field+++++++++YB.End++PConstant/editword/DTformat++
O                                          15 'EMPLOYEE TOTAL'
O          TF   L1                    1
O                         EMPLTOT          25
O          T    L1                    1
O                         EMPLTOT          35
O          T    L1                    1
O                         EMPLTOT          45
O          TF   L1                    1
O                         EMPLTOT          55
O          T    L1                    1
O                         EMPLTOT          65
```

```
0          T    L1                    1
0                         EMPLTOT              75
0          T    L1                    1
0*
```

图 113　取溢出的应用

18 列有 F 的总计行可以取溢出例程。如果在打印这些行的某行之前发生了溢出，则取溢出例程。在执行取溢出之前，要检查溢出指示器是否为 on。如果它是 on，取溢出例程，打印由溢出指示器限制的标题行，并处理总计操作。

16.2.4 在程序描述文件中修改格式控制信息

PRTCTL 键字允许修改格式控制信息并访问程序中描述的 PRINTER 文件当前行值。在 PRINTER 文件的文件描述规范表上指出键字 PRTCTL（数据结构名）。

可以在源文件中指定两种 PRTCTL 数据结构类型：一种是 OPM 定义的数据结构，另一种是 ILE 数据结构。

缺省值是使用表 16 中给出的 ILE 数据结构样本。要使用 OPM 定义的数据结构样本，规定 PRTCTL（数据结构名：*OMPAT）。表 17 给出的是 OPM　PRTCTL 数据结构样本。

ILE　PRTCTL 数据结构必须在定义规范中定义。它需要至少 15 个字节并且必需至少包含按下列顺序说明的五个子字段：

表 16　ILE　PRTCTL 数据结构设计

列	子字段内容
1--3	一个 3 位长的字符型字段，含有打印前的空行值（有效值：空格或 0--255）
4--6	一个 3 位长的字符型字段，含有打印后的空行值（有效值：空格或 0--255）
7--9	一个 3 位长的字符型字段，含有打印前的跳行值（有效值：空格或 0--255）
10--12	一个 3 位长的字符型字段，含有打印后的跳行值（有效值：空格或 0--255）
13--15	一个 3 位长无小数数字型字段，含有当前的行计数值。

OPM　PRTCTL 数据结构必须在定义规范表上定义而且必须至少含有按下列顺序说明的五个子字段：

表 17　OPM　PRTCTL 数据结构设计	
列	子字段内容
1	一个 1 位长的字符型字段，含有打印前的空行值（有效值：空格或 0--3）
2	一个 1 位长的字符型字段，含有打印后的空行值（有效值：空格或 0--3）
3--4	一个 2 位长的字符型字段，含有打印前的跳行值(有效值：空格,1--99,A0--A9 对应 100--109,130--132 对应 110--112)
5--6	一个 2 位长的字符型字段，含有打印后的跳行值（有效值：空格,1--99,A0--A9 对应 100--109，130--132 对应 110--112）
7--9	一个两位长无小数数字型字段，含有当前的行计数值

ILE　PRTCTL 数据结构的头 4 个子字段中包含的值和输出规范表的 40--51 列（空行和跳行登记项）中的值相同。如果输出规范表的空行/跳行项（40--51 列）是空格，而且子字段 1--4 也是空格，缺省值是打印后空一行。如果说明了 PRTCTL 键字，它仅用于在 40--51 列上是空格的输出记录。可以在程序运行时修改这些 PRTCTL 数据结构子字段的值来控制 PRINTER 文件的空行和跳行值（子字段 1--4）。

子字段 5 含有当前行数值。编译程序在打印了第一个输出行以后初始化子字段 5 的值。编译程序在对该文件的每个输出处理以后修改子字段 5 的值。

16.2.4.1 修改格式控制信息的例子

图 114 给出用 PRTCTL 键字修改格式控制信息必须的编码。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords++++++++++++++++++++++++++++++
FPRINT     0    F  132        PRINTER PRTCTL(LINE)

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
DName++++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++++++
DLINE             DS
D SpBefore                1      3
D SpAfter                 4      6
D SkBefore                7      9
```

```
D SkAfter                 10      12
D CurLine                 13      15  0

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
CLONO1Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....
C                   EXCEPT
C   01CurLine       COMP       10                                    49
C   01
CAN 49              MOVE       '3'           SpAfter

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
OFilename++DF..NO1NO2NO3Excnam++++B++A++Sb+Sa+.........................
OPRINT     E    01
O.............NO1NO2NO3Field+++++++++YB.End++PConstant/editword/DTformat++
O                       DATA                25
```

图 114　PRTCTL 选项的例子

在文件描述规范表上，为 PRINT 文件说明了 PRTCTL 键字，相应的数据结构名为 LINE。

在输入规范表上定义 LINE 数据结构，它仅有 PRTCTL 数据结构里前面定义的子字段。在 1--12 列的头 4 个子字段用来提供空行和跳行的资料，它们一般是在输出规范表的 40 到 51 列上说明。PRTCTL 键字允许你修改程序内的这些说明。

在这个例子里，当 Curline（当前行数值）子字段的值等于 10 时，修改 SpAfter 子字段的值为 3。（假设记录标识指示器 01 置为 on）。

16.3 访问磁带设备

往磁带文件上写记录时，使用 SEQ 设备描述。要往带文件上写变长记录，使用 CL 命令 CRTTAPF 或 OVRTAPF 的 RCDBLKFMT 参数。当使用 RCDBLKFMT 参数时，写到磁带上的每个记录长是由下列条件决定的：

• 该记录输出规范表上最后的结束位置。

• 如果没说明一个结束位置，编译程序从字段的长度来计算该记录的长度。

从带上读变长带记录与读任一顺序组织文件的记录是一样的。要保证在文件描述规范表上说明的记录长度，提供了该文件中最长的记录。

16.4 访问显示设备

用显示文件在程序和显示设备（象终端）间交流信息。显示文件用来定义显

示在终端上的信息格式，并且定义系统处理进出终端信息的方法。如何使用 WORKSTN 文件的资料见 17 章 4.5 中的“使用 WORKSTN 文件”。

16.5 使用顺序文件

ILE　RPG/400 程序中的顺序文件与 AS/400 系统上中任何顺序组织文件相联系，象：

- 数据库文件
- 软盘文件
- 打印机文件
- 磁带文件

文件描述规范表中 SEQ 文件名向 AS/400 说明顺序文件。AS/400 的文件描述规定了实际的 I/O 设备，象磁带、打印机和磁盘。

程序运行时也可以用 CL 的覆盖命令（象 OVRDBF，OVRDKTF，OVRTAPF）来指出实际的 I/O 设备。

16.5.1 顺序文件的指定

在文件描述规范表上 36--42 列说明的顺序设备(SEQ)指出了和一个顺序组织文件相联系的输入或输出。参考 4.4.5.1.1 中的图 116。在运行一个程序期间与文件相联系的实际设备可以用一个 OS/400 覆盖命令来规定，或者用文件名指出的文件描述来说明。如果在程序中说明 SEQ，不能指定设备相关功能，象空行/跳行，或者 CHAIN 等功能。

下图表示了 SEQ 文件所允许的操作码。

文件描述规范表		计算规范表
17 列	18 列	26--35 列
I	P/S	CLOSE，FEOD
I	F	READ，OPEN，CLOSE，FEOD
O		WRITE，OPEN，CLOSE，FEOD

注：对一个 SEQ 不允许打印控制说明。

图 115　SEQ 文件有效的文件操作码

16.5.1.1 规定顺序文件的例子

图 116 是一个在 ILE　RPG/400 原成员中如何指定 SEQ 文件的例子。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+ ...*
FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords++++++++++++++++++++++++++++++
FTIMECDS   IP   E             DISK
FPAYOTIME  O    F  132        SEQ
 *
```

图 116　SEQ 设备

PAYOTIME 文件说明为一个 SEQ 设备。当程序运行时，可用 OS/400 覆盖命令规定程序运行期间和文件相联系的实际设备（象打印机，带或软盘）。例如，某个程序运行使用软盘，另些程序运行需用打印机。用文件名指出的文件描述可以指出实际设备，在这种情况不需要使用覆盖命令。

16.6 使用特殊文件

RPG 设备名 SPECIAL（文件描述规范表的 36--42 列）允许指定一个输入和/或输出设备，这个设备不直接由 ILE　RPG/400 支持。文件的输入和输出操作由用户编写的例程来控制，例程名必须在文件描述规范表内用 PGMNAME（程序名）键字指定。

ILE　RPG/400 调这个用户编写的例程来打开该文件，读写记录并关闭该文件。例程 ILE　RPG/400 生成一个参数列表由用户写的例程来使用。这个参数列表包含：

- 可选编码参数(option)
- 返回状态参数(status)
- 错误查找参数(error)
- 记录域参数(area)

这个参数列表由 ILE　RPG/400 编译程序和用户编写的例程来访问；它不能由含有 SPECIAL 文件的程序来访问。

下面描述 RPG 生成的参数列表的参数：

•option：option 参数是一个一位长字符型字段，它指出用户编写的例程处理的动作。根据在 SPECIAL 文件上处理的操作(OPEN，CLOSE，FEOD，READ，WRITE，DELETE，UPDATE)，从 ILE　RPG/400 给用户编写的例程传递下列值：

传递的值	描　　述
O	打开文件
C	关闭文件
R	读一个记录并把它放到由区域参数定义的区域中
F	强迫结束文件
W	ILE　RPG/400 已经把一个记录放到由区域参数定义的区域

中，

	记录要被写出
D	删除该记录
U	该记录是最后读进记录的更新

•status：status 参数是一个 1 位长的字符型字段，它指出，当控制返回到 ILE RPG/400 时用户编写的例程的状态。当用户写的例程把控制返回到 ILE RPG/400 时，STATUS 必须含有下列的一个返回值：

返回值	描　述
0	正常返回，执行了所请求的活动。
1	输入文件是在文件末，没有记录返回。如果文件是一个输出文件，这个返回值是一个错误。
2	所请求的活动设执行；存在错误条件。

• error：error 参数是一个没有小数位的 5 位数字的区十进位制数字字段。如果用户写的例程发现一个错误，则error参数含有一个表示这类错误的值。当status参数是 2 时，该值放在 IWFDS 的*RECORD 的头 5 列上。

• area：区域参数是字符型字段，它的长度等于和 SPECIAL 文件相联系的记录长度，这个字段用来从 ILE　RPG/400 程序传送或接收记录。

可以增加其它参数到 RPG 生成的参数列表上，在文件描述规范表上为 SPECIAL 文件指定键字 PLIST（参数列表名）。见 4.4.6.1 中的图 118，然后在计算规范表上用 PLIST 操作去定义其它的参数。

用户编写的例程，在 SPECIAL 文件的文件描述规范表上用 PGMNAME 键字指定，必须包含一个输入参数列表，包括 RPG 生成的参数和用户说明的参数。

如果把 SPECIAL 文件规定为主文件，用户说明的参数必须在第一次读入主文件之前初始化。这可以用 PARM 参数的因子 2 来规定，也可用编译时数组，或用作为参数的一个数组元素来做。

图 117 表示 SPECIAL 文件有效的文件操作码。

```
* 文件描述规范表            计算规范表              *
*  17    18                 26-35                   *
----------------------------------------------------
*  I     P/S                CLOSE, FEOD             *
*                                                   *
*  C     P/S                WRITE, CLOSE, FEOD      *
*                                                   *
```

```
*   U      P/S              UPDAT, DELETE, CLOSE       *
*                           FEOD                       *
*                                                      *
*   O                       WRITE, OPEN, CLOSE,        *
*                           FEOD                       *
*                                                      *
*   I      F                READ, WRITE, OPEN,         *
*                           CLOSE, FEOD                *
*                                                      *
*   C      F                READ, WRITE, OPEN,         *
*                           CLOSE, FEOD                *
*                                                      *
*   U      F                READ, UPDATE, DELETE,      *
*                           OPEN, CLOSE, FEOD          *
```

图 117　对一个 SPECIAL 文件有效的文件操作

16.6.1 使用特殊文件的例子

图 118 表示如何在程序中运用 RPG 设备名。
SPECIAL 在这个例子中，文件 EXCPTN 是和 SPECIAL 设备相关的。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords+++++++++++++++++++++++++++++
FEXCPTN    O    F   20        SPECIAL PGMNAME('USERIO')
F                                    PLIST(SPCL)

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
DName++++++++++++ETDsFrom+++To/L+++IDc.Functions+++++++++++++++++++++++++++++
D OUTBUF          DS
D  FLD                     1     20

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....
C     SPCL          PLIST
C                   PARM                    FLD1
C                   MOVEL     'HELLO'       FLD
C                   MOVE      '1'           FLD1              1
```

```
C                   WRITE     EXCPTN        OUTBUF
C                   MOVE      '2'           FLD1            1
C                   WRITE     EXCPTN        OUTBUF
C                   SETON                                     LR
```

图 118　SPECIAL 设备

图 119 表示用户编写的程序 USER10。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....

 *----------------------------------------------------------------*
 * 头 4 个参数是 ILE RPG/400 生成的参数，其余的是程序员定义的 PLIST  *
 * 定义的。                                                       *
 *----------------------------------------------------------------*
C     *ENTRY        PLIST
C                   PARM                    OPTION          1
C                   PARM                    STATUS          1
C                   PARM                    ERROR           5 0
C                   PARM                    AREA           20
C                   PARM                    FLD1            1

 *----------------------------------------------------------------*
 * 用户写的程序完成依进入选项而定的文件 I/O 操作。                *
 *----------------------------------------------------------------*
C                   SELECT
C                   WHEN      OPTION = 'O'
C*  perform OPEN operation
C                   WHEN      OPTION = 'W'
C*  perform WRITE operation
C                   WHEN      OPTION = 'C'
C*  perform CLOSE operation
C                   ENDSL
C                   RETURN
```

图 119　用户编写的程序 USER10

SPECIAL 设备的 I/O 操作是由用户写的程序 USER10 来控制的。由程序员定义的 PLIST（SPCL）所说明的参数加到 SPECIAL 设备的 RPG 生成的参数列表末尾，程序员定义的参数可以由用户 ILE　RPG/400 程序和用户写的例程 USER10 来访问；而 RPG 生成的参数列表却仅能由内部 ILE　RPG/400 逻辑和用户写的例程访问。

第十七章　使用 WORKSTN 文件

AS/400 上的交互应用通常包括和下列对象通信：

- 通过显示文件的一个或多个终端用户
- 通过 ICF 文件的一个或多个远程系统程序
- 通过 ICF 文件的一个或多个远程系统上的设备

显示文件是 AS/400 系统上具有 DSPF 属性的*FILE 类型的目标。可以用显示文件与终端用户进行交互式通信。象数据库文件一样，显示文件既可以是外部描述的又可以是程序描述的。

ICF 文件是 AS/400 系统上具有 ICFF 属性的*FILE 类型的目标。可以用 ICF 文件与在远程系统上（AS/400 或非 AS/400）的其它应用程序进行通信（传递和接收数据）。ICF 文件含有系统间传递和接收数据所需要的通信格式。可以写一个使用 ICF 的程序，与在远程系统上的其它应用程序通信（传递和接收数据）。

当 RPG 程序中一个文件用 WORKSTN 设备名说明时，程序可以和终端用户交互通信或者用 ICF 文件与其它程序进行通信。这一章介绍如何使用：

- 系统间通信功能(ICF)
- 外部描述 WORKSTN 文件
- 程序描述 WORKSTN 文件
- 多设备文件

16.1 系统间通信功能

为了使用 ICF，要在程序中定义一 WORKSTN 文件，引用做 ICF 设备文件。可以使用系统提供的文件 QICDMF 或者使用 OS/400 命令 CRTICFF 建立的文件。

为了使用 ICF，可在程序中把 ICF 作为一个文件编码，ICF 文件类似于显示文件。它包括系统之间传送和接收数据所需的通信格式。

关于 ICF 的进一上描述，参考《ICF 编程》。

16.2 使用外部描述的 WORKSTN 文件

RPG　WORKSTN 文件可以用外部描述的显示设备文件或 ICF 设备文件，它包括文件信息和要写出的记录中字段的描述。用得最多的外部描述 WORKSTN 文件是显示文件。（关于描述和生成显示文件的资料，请看《DDS 参考》）。

除了对于字段描述外（象字段名和属性），显示设备文件的 DDS 可以用来：

• 指出每个字段和常量所在行数和列数来确定记录放置在屏幕上的格式。

• 指出有关的功能，象下划线、高亮度显示字段、反象显示、闪烁光标等。

• 指定对在显示工作站上录入的数据进行有效性检查。有效性检查功能包括检查字段在哪儿需要数据，命令填充字段，错误数据类型，数据的范围，录入的数据是否有效，以及检查模 10 或 11 的数据的正确性。

• 控制屏幕管理功能，象显示新数据时字段是否擦除，复盖或保持等。

• 把指示器 01--99 与命令注意键和命令功能键联系起来。如果把一个功能键描述成一个命令功能键(CF)，则把响应指示器和数据记录（在屏幕上录入的修改的数据）都返给到程序。如果把一个功能键描述成一个命令注意键(A)，响应指示器返给程序而数据记录保持不变。这样，仅输入的字符型字段是用空格充填而仅输入数字型字段用零充填，除非这些字段用另外的内容初始化。

• 给一个字段分配一个编辑码(DETCDE)或编辑字。规定字段的值在屏幕上如何显示。

• 指出子文件

显示设备记录格式包括下列三种类型的字段：

• 输入字段。当程序读一个记录时，从设备中把输入字段传送到程序中。输入字段可用缺省值初始化，如果缺省值没改变，把缺省值传送到程序中。没有初始化的输入字段作为空格显示，工作站用户可以往那里录入数据。

• 输出字段。当程序写一个记录显示时，输出字段是从程序传送到设备上。输出字段可以由程序提供或者由设备文件中的记录格式提供。

• 输出/输入字段。输出/输入字段是一个可以修改的输出字段。如果它被修改就变成一个输入字段。当程序从显示屏上读一个记录时，输出/输入字段传送到程序中，而当程序要往显示上写一个记录时，输出/输入字段从程序传送到显示屏上。输出/输入字段在用户要修 改或更新，从程序写到显示屏上的数据时使用。

如果在 DDS 中为一个 WORKSTN 文件指定了关键字 INDARA，RPG 程序用指示器区把指示器传送到 WORKSTN 文件，而不是用输入/输出缓冲器。

有关外部描述显示设备文件和有效的 DDS 键字方面的详细介绍，见《DDS 参考》。

图 120 是显示设备文件的 DDS 的例子。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
AAN01N02N03T.Name++++++RLen++TDpBLinPosFunctions++++++++++++++++++*
A** ITEM MASTER INQUIRY
A                                       REF(DSTREF)  1
A          R PROMPT                     TEXT('Item Prompt Format')
A  73N61                                OVERLAY  2
A                                       CA03(98 'End of Program')  3
A                                   1  2'Item Inquiry'
A                                   3  2'Item Number'
A            ITEM      R        I  3 15PUTRETAIN  4
A  61                                   ERRMSG('Invalid Item Number' 61) 5
A          R RESPONSE                   TEXT('Response Format')
A                                       OVERLAY  2
A                                       LOCK  6
A                                   5  2'Description'
A            DESCRP    R           5 15
A                                   5 37'Price'
A            PRICE     R           5 44
A                                   7  2'Warehouse Location'  7
A            WHSLOC    R           7 22
A                                   9  2'On Hand'
A            ONHAND    R           9 10
A                                   9 19'Allocated'  8
A            ALLOC     R           9 30
A                                   9 40'Available'
A            AVAIL     R           9 51
A*
```

图 120　显示设备文件的数据描述规范示例

这个显示设备文件有两个记录格式：PROMPT 和 RESPONSE。

1.这个文件中字段的属性在 DSTREE 字段引用文件中定义。

2.使用键字 OVERLAY 这样两个记录格式可以使用同一个显示。

3.功能键 3 和指示器 98 相联，程序员用它结束程序。

4.PUTRETAIN 键字允许在 ITEM 字段中录入的值在显示上保持。另外，用 38 列上的 I 把字段 ITEM 定义为一个输入字段，在这些记录格式中 ITEM 是仅输

入字段。其他所有字段因为在 38 列上是空格，所以都是输出字段。

5.ERRMSG 键字标识错误信息，如果在使用这个记录格式的程序中指示器 61 置 on 时，显示该错误信息。

6.LOCK 键字在初始显示 RESPONSE 记录格式时防止用户从键盘上录入。

7.象“Description”，“Price”和“Warehouse　Location”这些常量描述由程序写出的字段。

8.给出行数和列数，字段或常量写在显示器上的位置。

16.2.1 在显示设备文件中指定功能键指示器

如果在 DDS 中说明了关联的功能键，对于 WORKSTN 文件的程序，功能键指示器 KA--KN，KP--KY 是有效的。

功能键指示器和功能键这样对应：功能键指示器 KA 对应功能键 1，KB 对应功能键 2，… …，KX 对应功能键 23，KY 对应功能键 24。

在 DDS 中用键字 CFxx（命令功能）或 CAxx（命令注意）说明功能键。例如，键字 CF01 允许使用功能键 1。当按功能键 1 时，功能键指示器 KA 在 RPG 程序里置 on。如果规定了功能键 CF01(99)，那么功能键 KA 和指示器 99 在 RPG 程序中都设为 on。如果工作站用户按了一个在 DDS 中没说明的功能键，OS/400 告诉用户按了一个无效键。

如果工作站用户按了一个说明了的功能键，当从记录中抽取字段（传送字段逻辑）时，在 RPG 程序中相应的功能键指示器置为 on 而其他的功能键指示器置 off。如果没按功能键，所有的功能键指示器在传送字段时都置为 off。如果用户按 Enter 键，功能键指示器置 off。

16.2.2 在显示设备文件上指定命令键

可以在 DDS 中用键字 HELP，ROLLUP，ROLLDOWN，PRINT，CLEAR 和 HOME 为一个显示设备文件说明命令键 Help，Rollup，RollDown，Prine，Clear 和 Home。

命令键是由 RPG 程序处理的。RPG 编译程序处理在 DDS 中指出了相应键字的记录格式的 READ 或 EXFMT 操作，它就处理命令键。当命令键有效且按一个命令键时，OS/400 将控制返回给程序。如果在 DDS 中为所选择的命令规定了响应指示器，则那个指示器置 on，而对该记录格式有影响的所有其他的响应指示器和文件指示器置 off。

如果在 DDS 中没有为命令键指定响应指示器，将出现下列情况：

• 不用*PGM 的 Print 键，执行打印功能。

• 与子文件联用的 ROLLUP 和 ROLL　DOWN 键，在显示的子文件内上下翻页。如果企图翻的页超出子文件的头或尾，将得到一个运行时错误。

•用*PGM 说明的打印键，没有和子文件联用的 ROLL　UP 和 ROLL　DOWN

键，Clear，Help，Home 键，如果*STATUS 的值置为 1121--1126 之一，则继续执行。

16.2.3 处理外部描述的 WORKSTN 文件

当处理一个外部描述的 WORKSTN 文件时，OS/400 系统把数据从程序中转换成文件规定的格式并显示该数据。当把数据传送给程序时，把数据转换成程序所用的格式。

OS/400 系统为设备执行输入/输出操作提供设备控制信息。当从设备上请求一个输入记录时，OS/400 发出请求，然后在把数据传送给程序之前从数据中取消设备控制信息。另外，OS/400 可以把指示器传送到程序，以指出记录中的哪个字段已做了修改。

当程序请求一个输出操作时，它把输出记录传给 OS/400。OS/400 提供显示该记录必需的设备控制信息。当显示记录时 OS/400 也加上记录格式规定的常量信息。

当把一个记录传给程序时，字段按照它们在 DDS 中说明的顺序安排。字段显示的位置是根据在 DDS 中给字段定的位置（行数和列数）确定的。字段在 DDS 中说明的顺序和它们在屏幕上出现的顺序没必要一样。

关于处理 WORKSTN 文件的资料见 4.5.4 中的“有效的 WORKSTN 文件操作”。

16.2.4 使用子文件

子文件是用 DDS 在显示文件中说明的，能在显示上处理多个相同类型的记录。（见图 121）一个子文件是从显示设备文件上读或写的一组记录。例如，程序从数据库文件读记录，生成一个输出记录的子文件。当写出整个子文件时，程序用一个写操作把整个子文件送到显示设备。工作站用户可以在子文件上修改数据或录入其它数据；然后程序从显示设备把整个子文件读到程序里面再分别处理子文件中的每个记录。

要放在子文件中的记录要在 DDS 中说明。能放在一个子文件中的记录数也必须在 DDS 中说明。一个文件可以有多个子文件，最多可有 12 个子文件同时活动，可以同时显示两个子文件。

一个子文件的 DDS 由两个记录格式构成：子文件记录格式和子文件控制记录格式。子文件记录格式在子文件控制格式控制下与显示文件传输字段信息。子文件控制记录格式控制子文件所做的物理读写或控制操作。图 122 给出的子文件记录格式的 DDS 例子，图 123 是子文件控制记录格式的 DDS 的例子。

如何使用子文件键字的介绍，见《DDS 参考》。

```
*                                                                                *
* Customer Name Search                                                           *
*                                                                                *
* Search Code _______                                                            *
*                                                                                *
* Number  Name                  Address               City                State  *
*                                                                                *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
* XXXX    XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX     *
*                                                                                *
```

图 121　子文件显示

要在 RPG 程序中使用显示设备文件的子文件，必须在 WORKSTN 的文件描述规范表上指定 SFILE 键字。SFILE 键字的形式是：SFILE(记录格式名：RECNO 字段名)。WORKSTN 文件必须是一个外部描述文件（22 列上是 E）。

必须为 SFILE 键字指定子文件记录格式名（不是控制记录格式）和子文件处理中所用的含有相对记录号的字段名。

在 RPG 程序中，相对记录号处理是 SFILE 定义的一部分。SFILE 定义隐含对子文件定义为有 ADD 选项的全过程更新文件。所以，对子文件有效的文件操作不依赖于主 WORKSTN 文件的定义，即 WORKSTN 文件可以定义为一个主文

件或一个全过程文件。

对子文件记录格式使用 CHAIN，READC，UPDATE 或者 WRITE 操作码，在程序和子文件之间做数据传输。

对子文件控制记录格式使用 READ，WRITE 或 EXFMT 操作码在程序和显示设备之间做数据传输或执行子文件控制操作。

子文件处理遵循相对记录号处理的原则。RPG 程序把由 READC 操作取得的记录的相对记录号放到 SFILE 键字第二位命名的字段里。这个字段也用来指出 RPG 程序用 WRITE 操作或者往子文件中写的记录号执行用 ADD 的输出操作。SFILE 键字中的 RECNO 字段名必须定义为无小数位的数字型字段。该字段必须有足够的位置来存放文件的最大记录号（见《DDS 参考中的 SFILE 键字》），在输出规范表上指出 WRITE 和 ADD 操作码需要的相对记录号字段在文件描述规范表的 SFILE 键字的第二个位置说明。

如果 WORKSTN 文件有一个相关的子文件，所有引用这个文件的隐式的输入操作和明确的计算操作都不在主 WORKSTN 文件上执行。任意一个不作子文件记录格式名的操作都在主 WORKSTN 文件上执行。

如果在读一个非子文件记录期间按下指定的功能键，随后的读取子文件记录会导致响应功能键指示器又一次被置 on，即使在两次读之间功能键指示器已被置 off，这种情况一直会继续直到从 WORKSTN 文件中读非子文件记录。

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
AAN01N02N03T.Name++++++RLen++TDpBLinPosFunctions++++++++++++++++++++*
A** CUSTOMER NAME SEARCH
A                                       REF(DSTREF)  1
A          R SUBFIL                     SFL  2
A                                       TEXT('Subfile Record')
A            CUST      R          7  3
A            NAME      R          7 10
A            ADDR      R          7 32  3
A            CITY      R          7 54
A            STATE     R          7 77
```

图 122　子文件记录格式的 DDS

一个子文件记录格式的 DDS 描述了子文件中的记录：

1.记录格式中字段的属性包括在由 REF 键字所指出的字段引用文件 DSTREF 中。

2.SFL 关键字标识该记录格式是一个子文件的记录格式。

3.行和列项定义了显示上字段的位置。

16.2.4.1 子文件的使用

应用子文件的几种典型方法是：

• 仅显示。工作站用户查询显示。

• 有选择的显示。用户请求一个题目多种信息的显示。

• 修改。用户修改一个或多个记录。

• 仅输入，不做有效性检查。用子文件完成数据录入功能。

• 仅输入，进行有效性检查。用子文件完成数据录入功能，但要检查记录。

• 综合任务。用子文件作显示同时可进行修改以及增加新记录。

下图是子文件控制记录格式的 DDS 例子。在 RPG 程序中用子文件的例子见 4.6.4。

```
 *.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
AAN01N02N03T.Name++++++RLen++TDpBLinPosFunctions++++++++++++++++++*
A          R FILCTL                    SFLCTL(SUBFIL)
A N70                                  SFLCLR
A  70                                  SFLDSPCTL
A  71                                  SFLDSP
A                                      SFLSIZ(15)
A                                      SFLPAG(15)
A                                      TEXT('Subfile Control Record')
A                                      OVERLAY
A  71                                  ROLLUP(97 'Continue Search')
A                                      CA01(98 'End of Program')
A                                      HELP(99 'Help Key')
A                                   1  2'Customer Name Search'
A                                   3  2'Search Code'
A            SRHCOD    R         I  3 14PUTRETAIN
A                                   5  2'Number'
A                                   5 10'Name'
A                                   5 32'Address'
A                                   5 54'City'
A                                   5 76'State'
A*
```

图 123　子文件控制记录格式的 DDS

子文件控制记录格式定义了子文件的属性，检索输入字段，常量和功能键。

可以使用的键字如下所示：

- SFLCTL 命名相关子文件(SUBFIL)
- SFLCLR 指出什么时候清除子文件(当指示器 70 为 off 时)
- SFLDSPCTL 指出什么时候显示子文件控制记录(当指示器 70 是 on 时)
- SFLSIZ 指出包括在子文件中的记录总数(15)
- SFLPAG 指出每页上的记录总数(15)
- ROLL　UP 指出当用户按 ROLL　UP 键时程序中的指示器 99 是 on
- HELP 允许用户按 Help 键显示有效功能键的信息
- PUTRFTAIN 允许在 SRHCOD 字段中录入的值保持在显示上

另外，对控制信息，子文件控制记录格式还定义了用作子文件记录格式列标题的常量。

16.3 使用程序描述的 WORKSTN 文件

在输出规范表上用或不用记录格式名，来使用程序描述的 WORKSTN 文件，如果指出了格式名，引用 DDS 中的记录格式名。这个记录格式描述了：

- 在屏幕上怎样显示从 RPG 程序发送的数据流
- 发送什么数据
- 完成了什么 ICF 功能

如果用一个格式名，必须用输入或输出规范表来描述输入和输出记录。

可以在文件描述规范表上为一个程序描述的 WORKSTN 文件指定 PASS(*NOIND)。PASS(*NOIND)指出，RPG 程序不另外把指示器传递到输出或输入数据中。你的任务是，通过把指示器描述为输入或输出记录中的字段来传送指示器（用格式*INxx，*IN 或*IN(x)）。它们必须用(DDS)中所需要的顺序说明。可以用 DDS 列表来检测这个顺序。

16.3.1 使用有格式名的程序描述 WORKSTN 文件

下列规范表适用于使用格式名的程序描述的 WORKSTN 文件。

输出规范表：

在输出规范表上，必须在 7--16 列上指出 WROEKSTN 文件名。DDS 的记录格式名要在字段描述行的 53--80 列作为一个文字或命名常量来说明。在格式名行的 47--51 列必须指定（右对齐）K1--K10。K 标识该项是作为长度，而不是结束位置，数字指出格式名的长度。例如，如果格式名是 CUSPMT，在 47--51 列上的登记是 K6，（K 后面允许有前置零）。格式名不能是条件限制的（21--29 列上的指示器无效）。

输出字段必须用 DDS 中定义的顺序放置在输出记录中，但字段名不必相同。字段的结束位置项参考从 RPG 程序传送到数据管理的输出记录的结束位置，而

不是字段在屏幕上的位置。

要在输出上传递指示器，做下面之一：

• 在 WORKSTN 文件的 DDS 中指定 INDARA 键字。不要在文件描述规范表上用 PASS(*NOIND)键字也不要在输出规范表上指定指示器。程序和文件用单独的指示器区来传递指示器。

• 在文件描述规范表上指定 PASS(*NOIND)键字。在输出规范表上用格式 *INxx 作为字段说明指示器。指示器字段必须在输出记录中其他的字段的前面，而且必须以 WORKSTN 文件的 DDS 中说明的顺序出现。可以用 DDS 列表检测它们的顺序。

输入规范表：

输入规范表描述 RPG 程序从显示设备或 ICF 设备接收的记录。WORKSTN 文件名必须在 7--16 列上说明。输入字段必须用 DDS 中定义的同样顺序安排在输入记录中。但字段名不必相同。字段位置项参考输入记录中字段的位置。

要在输入上接收指示器，要按以下做下面之一：

• 在 WORKSTN 文件的 DDS 上指定 INDARA 键字。在文件描述规范表上不使用 PASS(*NOIND)键字，而且在输入规范表上不说明指示器。程序和文件用单独的指示器域来传递指示器。

• 在文件描述规范表上指定 PASS(*NOIND)键字。在输入规范表上用格式 *INxx 把指示器作为字段来指定。它们必须以 WORKSTN 文件的 DDS 中说明的顺序出现在输入记录上。可以用 DDS 列表中检测这个顺序。

给文件中的每个记录配置一个记录标识指示器，它指出从 WORKSTN 文件已经读进的记录。在 DDS 中可以为记录标识码说明一个有缺省值的隐蔽字段。

计算规范表：

对定义为组合全过程文件的程序描述的 WORKSTN 文件，READ 操作码是有效的。见 4.5.4 中的图 124。必须在这个操作的因子 2 上说明文件名。在设备执行输入操作之前，格式必须存在。这个需要可以在显示设备通过用 1P 条件限制一个输出记录，或者往其它程序（例如 CL 程序）中的设备上写第一个格式。EXFMT 操作无效。可用 EXCPT 操作往 WORKSTN 文件中写记录。

附加的考虑：

当用一个有格式名的程序描述 WORKSTN 文件时，你也必须要考虑以下几点：

• 在输出规范表的 53--80 列上所指出的名是用来生成该文件的 DDS 中记录格式名。

• 如果对输出记录说明了 Kn，它文件的其他输出记录也必须使用它。如果 Kn 不是为文件的所有输出记录所用，将出现一个运行时错误。

16.3.2 使用没有格式名的程序描述的 WORKSTN 文件

当不使用记录格式名时，程序描述的显示设备文件描述了包含字段描述的一个记录格式的文件。记录中的字段必须在使用该文件的程序内描述。

当用生成显示文件命令生成显示文件时，文件具有下列属性：

• 可以说明一个变长的记录；所以，实际的记录长度必须在应用程序中指出（允许的最大记录长度是屏幕尺寸减 1）。

• 没有指示器在程序和文件中传输。

• 不定义功能键指示器

• 写到显示上的记录在第一个有效行的第 2 列开始。

输入文件：

对一个输入文件，OS/400 设备支持把输入记录看作是一个单个的输入字段，当文件打开时它被初始化成空格。光标指向字段头，是显示上的第 2 列。

输出文件：

对一个输出文件，OS/400 设备支持把输出记录看作是发往显示的一个字符串。每个输出记录作为文件的下一个顺序记录写出；即显示的每个记录覆盖前面显示的记录。

组合文件：

对一个组合文件，OS/400 设备支持把记录看作是一个单独的字段。出现在屏幕上，即输出又输入记录。设备支持初始化输入记录为空格，而光标放在第 2 列上。

有关程序描述显示设备文件的详细资料，见《数据管理》。

16.4 有效的 WORKSTN 文件操作

图 124 表示的是 WORKSTN 文件的有效文件操作码。

文件描述规范表中位置		计算规范表中位置
17	18	26-35
I	P/S	CLOSE, ACQ, REL, NEXT, POST, FORCE
I	P/S	WRITE(1), CLOSE, ACQ, REL, NEXT, POST, FORCE
I	F	READ, OPEN, CLOSE, ACQ, REL, NEXT, POST
C	F	READ, WRITE(1), EXFMT(2), OPEN, CLOSE, ACQ, REL, NEXT, POST, UPDATE(3), CHAIN(3), READC(3)
O	Blank	WRITE(1), OPEN, CLOSE, ACQ, REL, POST

注：⑴WRITE 操作对用格式名的程序描述文件无效
注：⑵ 如用 EXFMT 操作，文件必须是外部描述文件
注：⑶对子文件记录格式，UPDATE、CHAIN、READC 操作也是有效的。

图 124　WORKSTN 有效的文件操作码

下面进一步解释操作码 EXFMT，READ 以及 WRITE 在处理 WORKSTN 文件时的用法。

16.4.1　EXFMT 操作

EXFMT 操作是对同一个记录格式先写后读的组合操作(相当于数据管理的写-读操作)，如果把一个 WORKSTN 文件在文件描述规范表中定义为一个全过程(18 列是 F)组合文件(17 列是 C)，而使用外部描述的数据(22 列是 E)，EXFMT(执行格式)操作码用来从显示上读和往显示上写数据。

16.4.2　READ 操作

对一个全过程组合文件或一个全过程输入文件，不管是使用外部描述的数据，还是使用程序描述的数据，READ 操作都是有效的，READ 操作从显示上接收一条记录，但任何输入操作之前，记录格式必须在设备上存在，这种要求可以用下面方法实现：

在显示设备上用 1P 指示器做输出条件，通过从另一个程序往设备上写第一个格式或者如果是用记录格式名来读，在 DDS 的记录格式描述上使用键字 INZRCD。

16.4.3　WRITE 操作

WRITE 操作对一个组合文件或一个输出文件是有效的，它把一条新的记录写到显示上，输出规范表和 EXCPT 操作也能往一个 WORKSTN 文件上写记录，关于这些操作码的完整介绍，请见《ILE RPG/400 参考》。

16.5 多设备文件

在文件描述规范表中规定了至少有 DEVID、SAVEIND、MAXDEV(*FILE)或 SAVEDS 键字之一的 WORKSTN 文件是一个多设备文件，通过一个多设备文件，程序可以访问多个设备。

RPG 通过程序设备访问设备，程序设备对实际设备来说是一个符号结构，当生成一个文件时（用 DDS 描述和生成文件命令），要考虑：哪一个设备和程序设备相关联，文件是否得请求程序设备，用哪个记录格式来请求设备响应一个用文件名的读操作，以及 READ 操作要多长响应时间，关于生成一个多设备文件的可选项以及要求的详细资料，请见数据管理有关 ICF 文件和 ICF 程序设计，请参见相应的手册。

使用多设备文件，特别要用到下列操作码：

• 除打开一个文件外，OPEN 操作隐含得到在生成文件时指定的设备。

• ACQ(得到)操作得到多设备文件的其他设备。

• REL(释放)操作从文件中释放设备。

• 与 DDS 键字 INVITE 一起用 WRITE 操作时，程序设备响应后续的。从请求的程序设备的读操作，详见《数据管理》和《ICF 程序设计》。

• READ 操作，执行一个从请求程序设备的读操作或者执行一个从程序设备的读操作，当设有 NEXT 操作时，程序周期读或者用文件名的读操作等待来自响应请求设备上的输入操作（从请求的程序设备中读）。其他的输入和输出操作，包括 NEXT 操作之后的用文件名的读操作，以及用格式名的读操作，是用一个特定字段指出的程序设备处理从一个程序设备的读操作(这个字段在文件描述规范表中用 DEVID 键字给出)。

•这个设备可以是最后一次输入操作所用的设备。一个指定的设备或者是请求的程序设备，有关内容请见《ICF 程序设计》和《数据管理》。

• NEXT 操作规定哪一个设备用在由文件名读或程序周期读的下一次读中。

•POST 操作把信息放在信息数据结构 INFDS 中，这些信息可以是有关某个设备或者文件的信息(POST 操作使用多设备文件不受限制)。

有关 RPG 操作码的详细介绍，请见《ILE RPG/400 参考》。

在文件描述规范表中，可以指定若干个键字来控制多设备文件的处理。

• MAXDEV 键字指出是单设备文件还是多设备文件。

• MAXDEV(*FILE)处理有文件处理时定义的最多个设备数目的多设备文件，MAXDEV(*ONLY)处理仅有一个设备的文件。

• DEVID 键字允许指定一个可直接进行输入输出操作的程序设备名。

• 当执行从一个程序设备读或者 WRITE 操作时，操作所使用的设备是键字 DEVID 的参数规定的设备，这个字段初始化为空格，并且用最后一次成功输入操作的设备名来更新，它还能通过赋值明确的设置。ACQ 操作码对此字段不产生影响，如果没有规定 DEVID 键字，那么输入操作将在最后一次成功输入操作所发生的设备上执行，如果从某一设备的读操作没有成功地执行，使用空格作为设备名。

• 当对空格的设备名的设备做从程序设备读或者 WRITE 操作时，RPG 编译程序隐含使用程序请求的设备名，如果交互的调用它不是你想完成的某个操作，RPG 程序并且获得一个 ICF 设备，那么你必须明确地把 ICF 设备名传给以前完成过这个操作的 DEVID 键字指定的字段名。

如果不这样做，使用的设备名将为空格(在这种情况下，使用交互请求的设备名)，或者使用的设备是最后一次成功的输入操作所用的设备，一旦在 ICF 设备上执行一个 I/O 操作，就不必修改这个值，除非对不同的设备完成了输入操作。

• SAVEDS 键字指出用于存贮和恢复获得设备数据结构，SAVEIND 键字指出了指示器的设置，在输入操作之前，保存指示器的当前设置和数据结构，在输入操作之后，RPG 编译程序恢复与操作有关设备的指示器和数据结构，这和输入操作之前所用的指示器或者数据结构可能有所不同。

• INFDS 键字为 WORKSTN 文件指定了文件信息数据结构，RPG STATUS 字段和 I/O 操作的 majer/minor 返回码可以通过这个数据结构来访问，特别是用 ICF 时，这两个字段对

检测多设备文件的 I/O 操作期间发生的错误是很有用的。

注：当规定这些控制选项时，必须在 DEVID、SAVEIND 或者 SAVEDS 选项之前规定 MAXDEV 选项。

第十八章　交互应用的例子

这一章给出一些一般工作站应用开发以及 ILE　RPG/400 的编码。

本章提供的应用程序由四个模块组成，每个模块解释了 WORKSTN 文件的一般用法，第一个模块(CUSMAIN)给出程序的主菜单，根据用户的选择，它调用提供功能请求相应模块中的过程。

每个模块使用 WORKSTN 文件来提示用户在屏幕上输入和显示信息。除了主模块(CUSMAIN)之外，每个模块还使用了表示主数据库视图的逻辑文件，这个视图仅由这个模块处理所需要的主文件字段组成。

注：除 CUSMAIN 之外，每个模块可编译成一个独立的程序，也就是说，它们中的每一个都可以当作独立的程序来使用。

表 18：交互式应用例子中各模块的说明

模块	说明
4.6.2 主菜单查询	一个基本的菜单查询程序的例子 它用 WORKSTN 文件来显示菜单选择并操作输入
4.6.3 文件维护	一个维护程序的例子 它允许更新、删除、增加和显示主文件中的客户记录
4.6.4 由 ZIP 码检索	一个用 WORKSTN 子文件来处理显示所有与指定的 ZIP 码相匹配记录的例子
4.6.5 由名字检索及查询	一个用 WORKSTN 子文件处理显示所有与指定的客户名相匹配记录的例子，它允许用户从子文件中选择一条记录来显示客户的完整信息。

18.1 数据库物理文件

图 125 显示了这个客户文件的 DDS 描述，这个文件包括了每个客户的重要信息，例如，姓名、地址、帐号和客户号。要获得客户信息的每个模块都使用这个数据库文件，（或者它

的一个逻辑视图）。

```
A*****************************************************************
A* 文 件 名：CUSMST                                               *
A* 有关程序：CUSMNT，SCHZIP，SCHNAM                              *
A* 有关文件：CUSMSTL1，CUSMTSL2，CUSMSTL3（逻辑文件）            *
A* 说     明：这是物理文件 CUSMST，它的记录格式名为 CUSREC。      *
A*****************************************************************
A* CUSTOMER MASTER FILE -- CUSMST
A          R CUSREC
A            CUST           5  0       TEXT('CUSTOMER NUMBER')
A            NAME          20          TEXT('CUSTOMER NAME')
A            ADDR1         20          TEXT('CUSTOMER ADDRESS')
A            ADDR2         20          TEXT('CUSTOMER ADDRESS')
A            CITY          20          TEXT('CUSTOMER CITY')
A            STATE          2          TEXT('CUSTOMER STATE')
A            ZIP            5  0       TEXT('CUSTOMER ZIP CODE')
A            ARBAL         10  2       TEXT('ACCOUNTS RECEIVABLE
BALANCE')
```

图 125　主数据物理文件 CUSMST 的 DDS

18.2 主菜单查询

下面是一个简单的查询程序，它使用一个 WORKSTN 文件来显示菜单选项并接收输入。

18.2.1　MAINMENU：显示设备文件的 DDS

显示文件 MAINMENU 的 DDS 指定了文件层项和一个记录格式：HDRSCN 的描述，文件层定义了屏幕的大小(DSPSIZ)输入，缺省值(CHGINPDFT)，打印键(PRINT)和一个指示器区(INDARA)。

记录格式 HDRSCN 包含常量“CUSTOMER MAIN INQUIRY”，它标识了该显示，也有键字 TIME 和 DATE，将在屏幕显示当前的日期和时间，CA 键字定义了使用的功能键，及在 RPG 程序里与功能键相关联的指示器。

```
A*****************************************************************
A*  文 件 名：MAINMENU                                            *
```

```
A*  有关程序：CUSMAIN                                                *
A*  说      明：这是一个显示文件 MAINMENU，记录格式名为 HDRSCN。     *
A*********************************************************************
A                                      DSPSIZ(24 80 *DS3)
A                                      CHGINPDFT(CS)
A                                      PRINT(QSYSPRT)
A                                      INDARA
A           R HDRSCN
A                                      CA03(03 'END OF INQUIRY')
A                                      CA05(05 'MAINTENANCE MODE')
A                                      CA06(06 'SEARCH BY ZIP MODE')
A                                      CA07(07 'SEARCH BY NAME MODE')
A                                    2  4TIME
A                                      DSPATR(HI)
A                                    2 28'CUSTOMER MAIN INQUIRY'
A                                      DSPATR(HI)
A                                      DSPATR(RI)
A                                    2 70DATE
A                                      EDTCDE(Y)
A                                      DSPATR(HI)
A                                    6  5'Press one of the following'
A                                    6 32'PF keys.'
A                                    8 22'F3 End Job'
A                                    9 22'F5 Maintain Customer File'
A                                   10 22'F6 Search Customer by Zip Code'
A                                   11 22'F7 Search Customer by Name'
```

图 126　显示设备文件 MAINMENU 的 DDS

除了描述常量、字段、行数和屏幕的行列位置之外，记录格式还定义了这些项的显示属性。

注：通常，字段属性定义在一个字段引用文件中而不是在 DDS 文件中，这些属性在 DDS 中给出，这样你可以看出它们的内容。

18.2.2　CUSMAIN：RPG 源码

```
*****************************************************************
* 程 序 名：CUSMAIN                                             *
* 有关文件：MAINMENU（显示文件）                                 *
* 有关程序：CUSMNT   （ILE RPG 程序）                            *
*           SCHZIP  （ILE RPG 程序）                            *
*           SCHNAM  （ILE RPG 程序）                            *
*  说     明：   这是客户查询主程序，它提示用户选择下列之一：     *
*              1.管理客户记录（加更新、删除和显示）              *
*              2.用 ZIP 码检索客户                               *
*              3.用名字检索客户                                  *
*****************************************************************

FMAINMENU  CF   E             WORKSTN

C                   EXFMT     HDRSCN
C*
C                   DOW       NOT *IN03
C                   SELECT
C                   WHEN      *IN05
C                   CALLB     'CUSMNT'
C                   WHEN      *IN06
C                   CALLB     'SCHZIP'
C                   WHEN      *IN07
C                   CALLB     'SCHNAM'
C                   ENDSL
C                   EXFMT     HDRSCN
C                   ENDDO
C*
C                   SETON                                        LR
```

图 127　模块 CUSMAIN 的源码

这一模块解释了操作码 CALLB 的用法，根据用户菜单选项的选择由 CUSMAIN 调用相应的模块(CUSMNT、SCHZIP 或 SCHNAM)，为了生成程序目标：

1、使用命令 CRTRPGMOD 对每个源成员(CUSMAIN、CUSMNT、SCHZIP 和 SCHNAM)生成模块。

2、下列命令生成程序

CRTPGM　PGM(MYPROG)　MODULE(CUSMAIN CUSMNT SCHZIP SCHNAM)

ENTMOD(*FIRST)

注：*FIRST 选项指定了在列表中的第一个模块 CUSMAIN，作为程序的入口过程。

3、调用程序：

CALL MYPROG

出现如图 128 的主菜单

```
*                                                                    *
*   22:30:05              CUSTOMER MAIN INQUIRY              9/30/94   *
*                                                                    *
*    Press one of the following PF keys.                             *
*                                                                    *
*                    F3 End Job                                      *
*                    F5 Maintain Customer File                       *
*                    F6 Search Customer by Zip Code                  *
*                    F7 Search Customer by Name                      *
```

图 128　客户查询操作屏幕。

18.3 文件维护

下面说明了一个使用 WORKSTN 文件的维护程序。它允许增加、删除、更新和显示主客户数据文件中记录。

18.3.1　CUSMSTL1:逻辑文件的 DDS

```
A*********************************************************************
A* 文 件 名：CUSMSTL1                                                 *
A* 有关程序：CUSMNT                                                   *
A* 有关文件：CUSMST（物理文件）                                       *
A* 说　　明：  这是逻辑文件 CUSMSTL1，记录格式名为 CMLREC。是由       *
A*             客户号（CUST）对 CUSMST 的视图。                       *
A*********************************************************************
A          R CMLREC1                   PFILE(CUSMST)
A            CUST
A            NAME
A            ADDR1
A            ADDR2
A            CITY
```

```
A            STATE
A            ZIP
A          K CUST
```

图 129　逻辑文件 CUSMSTL1 的 DDS。

程序使用的数据库文件的 DDS 描述了一个记录格式：CMLREC1 给出在记录格式中的字段，CUST 作为该格式的键字字段。

18.3.2　MNTMENU：显示文件的 DDS

```
A*********************************************************************
A*  文 件 名：MNTMENU                                                *
A*  有关程序：CUSMNT                                                 *
A*  有关文件：CUSMSTL1（物理文件）                                    *
A*  说     明：这是一个显示文件，有 3 个记录格式。                    *
A*********************************************************************
A                                      REF(CUSMSTL1)
A                                      CHGINPDFT(CS)
A                                      PRINT(QSYSPRT)
A                                      INDARA
A          R HDRSCN
A                                      TEXT('PROMPT FOR CUST NUMBER')
A                                      CA03(03 'END MAINTENANCE')
A                                      CA05(05 'ADD MODE')
A                                      CA06(06 'UPDATE MODE')
A                                      CA07(07 'DELETE MODE')
A                                      CA08(08 'DISPLAY MODE')
A            MODE           8A  O  1   4DSPATR(HI)
A                                   1 13'MODE'
A                                      DSPATR(HI)
A                                   2   4TIME
A                                      DSPATR(HI)
A                                   2 28'CUSTOMER FILE MAINTENANCE'
A                                      DSPATR(HI RI)
A                                   2 70DATE
A                                      EDTCDE(Y)
A                                      DSPATR(HI)
A            CUST      R     Y  I 10 25DSPATR(CS)
```

```
A                                      CHECK(RZ)
A  51                                  ERRMSG('CUSTOMER ALREADY ON +
A                                      FILE' 51)
A  52                                  ERRMSG('CUSTOMER NOT ON FILE'+
A                                      52)
A                                   10 33'<--Enter Customer Number'
A                                      DSPATR(HI)
A                                   23  4'F3 End Job'
A                                   23 21'F5 Add'
A                                   23 34'F6 Update'
A                                   23 50'F7 Delete'
A                                   23 66'F8 Display'
A          R CSTINQ
A                                      TEXT('DISPLAY CUST INFO')
A                                      CA12(12 'PREVIOUS SCREEN')
A            MODE           8A  O  1  4DSPATR(HI)
A                                    1 13'MODE'
A                                      DSPATR(HI)
A                                    2  4TIME
A                                      DSPATR(HI)
A                                    2 28'CUSTOMER FILE MAINTENANCE'
A                                      DSPATR(HI)
A                                      DSPATR(RI)
A                                    2 70DATE
A                                      EDTCDE(Y)
A                                      DSPATR(HI)
A                                    4 14'Customer:'
A                                      DSPATR(HI)
A                                      DSPATR(UL)
A            CUST      R        O  4 25DSPATR(HI)
A            NAME      R        B  6 25DSPATR(CS)
A  04                                  DSPATR(PR)
A            ADDR1     R        B  7 25DSPATR(CS)
A  04                                  DSPATR(PR)
A            ADDR2     R        B  8 25DSPATR(CS)
A  04                                  DSPATR(PR)
A            CITY      R        B  9 25DSPATR(CS)
A  04                                  DSPATR(PR)
```

```
A            STATE     R        B 10 25DSPATR(CS)
A  04                                DSPATR(PR)
A            ZIP       R        B 10 40DSPATR(CS)
A                                    EDTCDE(Z)
A  04                                DSPATR(PR)
A                                  23  2'F12 Cancel'
A            MODE1          8   0 23 13
A          R CSTBLD                  TEXT('ADD CUST RECORD')
A                                    CA12(12 'PREVIOUS SCREEN')
A            MODE           8   0  1  4DSPATR(HI)
A                                   1 13'MODE'      DSPATR(HI)
A                                   2  4TIME
A                                    DSPATR(HI)
A                                   2 28'CUSTOMER FILE MAINTENANCE'
A                                    DSPATR(HI RI)
A                                   2 70DATE
A                                    EDTCDE(Y)
A                                    DSPATR(HI)
A                                   4 14'Customer:' DSPATR(HI UL)
A            CUST      R        O  4 25DSPATR(HI)
A                                   6 20'Name'      DSPATR(HI)
A            NAME      R        I  6 25
A                                   7 17'Address'   DSPATR(HI)
A            ADDR1     R        I  7 25
A                                   8 17'Address'   DSPATR(HI)
A            ADDR2     R        I  8 25
A                                   9 20'City'      DSPATR(HI)
A            CITY      R        I  9 25
A                                  10 19'State'     DSPATR(HI)
A            STATE     R        I 10 25
A                                  10 36'Zip'       DSPATR(HI)
A            ZIP       R     Y  I 10 40
A                                  23  2'F12 Cancel Addition'
```

图 130　显示文件 MNTMENU 的 DDS。

显示文件 MNTMENU 的 DDS 包含三个记录格式：HDRSCN、CSTINQ 和 CSTBLD。记录格式 HDRSCN 提示了客户号和处理方式，记录 CSTINQ 被用来更新、删除和显示模式。字段被定义成输出/输入(38 列上的 B)，当选择显示或删除方式时，字段被保护

(DSPATR(PR))。记录 CSTBLD 对新记录提供了输入字段(38 列的 I)。

记录格式 HDRSCN 包括常量‘Customer File Maintenance’。ERRMSG 键字定义了如果发生错误时显示的信息。CA 键字定义了与程序指示器相关联的功能键。

18.3.3 CUSMNT：RPG 源码

```
     ****************************************************************
     * 程 序 名：CUSMNT                                             *
     * 有关文件：CUSMSTL1（逻辑文件）                               *
     *           MNTMENU  （显示文件）                              *
     * 说    明：  这个程序用工作站文件做客户管理，允许加更新、删  *
     *             除和显示客户记录，按 F3 结程序。                 *
     ****************************************************************
     FCUSMSTL1  UF A E           K DISK
     FMNTMENU   CF   E             WORKSTN

     C     CSTKEY        KLIST
     C                   KFLD                    CUST
 ****************************************************************
*       主程序行                                                  *
****************************************************************
     C                   MOVE      'DISPLAY '    MODE
     C                   EXFMT     HDRSCN
     C*
     C                   DOW       NOT *IN03
     C                   EXSR      SETMOD
     C*
     C     CUST          IFNE      *ZERO
     C     MODE          CASEQ     'ADD'         ADDSUB
     C     MODE          CASEQ     'UPDATE'      UPDSUB
     C     MODE          CASEQ     'DELETE'      DELSUB
     C     MODE          CASEQ     'DISPLAY'     INQSUB
     C                   ENDCS
     C                   ENDIF
     C*
     C                   EXFMT     HDRSCN
     C                   ENDDO
     C                   MOVE      *ON           *INLR
```

```
*****************************************************************
*     子程序—ADDSUB                                            *
*     用  途—往文件中加新客户                                   *
*****************************************************************
     C     ADDSUB      BEGSR
     C     CSTKEY      CHAIN     CMLREC1                        50
     C                 IF        NOT *IN50
     C                 MOVE      *ON           *IN51
     C                 ELSE
     C                 MOVE      *OFF          *IN51
     C                 MOVE      *BLANK        NAME
     C                 MOVE      *BLANK        ADDR1
     C                 MOVE      *BLANK        ADDR2
     C                 MOVE      *BLANK        CITY
     C                 MOVE      *BLANK        STATE
     C                 Z-ADD     *ZERO         ZIP
     C                 EXFMT     CSTBLD
     C                 IF        NOT *IN12
     C                 WRITE     CMLREC1
     C                 ENDIF
     C                 ENDIF
     C                 ENDSR
*****************************************************************
      *     子程序—UPDSUB                                       *
      *     用  途—更新客户主文件记录                            *
*****************************************************************
     C     UPDSUB      BEGSR
     C                 MOVE      *OFF          *IN04
     C     CSTKEY      CHAIN     CMLREC1                        52
     C                 IF        NOT *IN52
     C                 EXFMT     CSTINQ
     C                 IF        NOT *IN12
     C                 UPDATE    CUSMSTL1
     C                 ELSE
     C                 UNLOCK    CUSMSTL1
     C                 ENDIF
     C                 ENDIF
     C                 ENDSR
```

```
*****************************************************************
*     子程序—DELSUB                                             *
*     用  途—删客户主文件记录                                   *
*****************************************************************
          C     DELSUB      BEGSR
          C                 MOVE      *ON           *IN04
          C     CSTKEY      CHAIN     CMLREC1                        52
          C                 IF        NOT *IN52
          C                 EXFMT     CSTINQ
          C                 IF        NOT *IN12
          C                 DELETE    CMLREC1
          C                 ELSE
          C                 UNLOCK    CUSMSTL1
          C                 ENDIF
          C                 ENDIF
          C                 ENDSR
 *****************************************************************
*     子程序—INQSUB                                             *
*     用  途—显示客户主文件记录                                 *
*****************************************************************
          C     INQSUB      BEGSR
          C                 MOVE      *ON           *IN04
          C     CSTKEY      CHAIN     CMLREC1                        52
          C                 IF        NOT *IN52
          C                 EXFMT     CSTINQ
          C                 UNLOCK    CUSMSTL1
          C                 ENDIF
          C                 ENDSR
 *****************************************************************
*     子程序—SETMOD                                             *
*     用  途—设置管理方式                                       *
*****************************************************************
          C     SETMOD      BEGSR
          C                 SELECT
          C                 WHEN      *IN05
          C                 MOVE      'ADD    '     MODE
          C                 WHEN      *IN06
          C                 MOVE      'UPDATE '     MODE
```

```
C                   WHEN      *IN07
C                   MOVE      'DELETE  '     MODE
C                   WHEN      *IN08
C                   MOVE      'DISPLAY '     MODE
C                   ENDSL
C*
C                   MOVE      MODE           MODE1
C                   ENDSR
```

图 131　模块 CUSMNT 的源码。

这个程序对客户主文件进行增加、修改和删除，也用来查询。

程序首先设置处理的缺省方式（显示）并显示客户维护提示屏幕，工作站用户可按 F3 键，它置 03 指示器为 ON 来结束作业，否则，处理客户信息，用户输入一个客户号并按执行键，用户可以用，F5(ADD)，F6(UPDATE)，F7(DELETE)或 F8(DISPLAY)来改变处理方式。

为了往文件中增加新记录，程序使用客户号作为检索自变量与主文件相连，如果文件中不存在记录，程序显示 CSTBLD 屏，允许用户输入一条新记录，如果记录已经存在，将显示错误信息，用户可按 F12 键，它置指示器 12 为 ON，取消增加操作，并释放记录，否则，处理增加操作，用户在输入字段进入新的客户记录信息并写入文件中。

为了更新、删除或显示已有的记录，程序使用客户号做检索自变量来连接文件，如果记录存在，则程序显示客户查询屏幕 CSTINQ，如果记录不存在，将显示一个出错信息，如果处理方式为显示或删除，则输入字段被保护不能修改，否则，处理客户记录，用户可以在输入字段中输入新的信息，可按 F12 键指示器 12 为 ON 来取消更新和删除操作并释放记录。当按执行键时显示方式自动释放记录。

在图 132 工作站用户输入客户号 00007 来显示记录。

```
*                                                                    *
*    DISPLAY  MODE                                                   *
*    22:30:21              CUSTOMER FILE MAINTENANCE          9/30/94   *
*                                                                    *
*                          00007   <--Enter Customer Number          *
*                                                                    *
*    F3 End Job      F5 Add     F6 Update     F7 Delete     F8 Display    *
*                                                                    *
```

图 132　客户文件维护显示方式显示屏

因在主文件客户号为 00007 的记录存在，所以数据象图 133 那样显示：

```
*                                                                    *
*   DISPLAY  MODE                                                    *
*   22:31:06              CUSTOMER FILE MAINTENANCE          9/30/94   *
*                                                                    *
*            Customer:  00007                                        *
*                                                                    *
*                       Mikhail Yuri                                 *
*                       1001 Bay Street                              *
*                       Suite 1702                                   *
*                       Livonia                                      *
*                       MI            11201                          *
*                                                                    *
* F12 Cancel DISPLAY                                                 *
*                                                                    *
*                                                                    *
```

图 133 客户文件维护显示方式屏

工作站用户输入一个新的客户号来响应增加的提示，如图 134：

```
*                                                                    *
*   ADD      MODE                                                    *
*   22:31:43              CUSTOMER FILE MAINTENANCE          9/30/94   *
*                                                                    *
*                       00012   <--Enter Customer Number             *
*                                                                    *
*   F3 End Job     F5 Add     F6 Update      F7 Delete      F8 Display   *
*                                                                    *
```

图 134 客户文件维护增加方式提示屏

在图 135 一条新的客户记录增加到客户主文件中去。

```
*                                                                    *
*   ADD      MODE                                                    *
*   22:32:04              CUSTOMER FILE MAINTENANCE          9/30/94   *
*                                                                    *
*            Customer:  00012                                        *
```

```
*                                                                    *
*                    Name JUDAH GOULD                                *
*                 Address 2074 BATHURST AVENUE                       *
*                 Address                                            *
*                    City YORKTOWN                                   *
*                   State NY         Zip 70068                       *
*                                                                    *
*  F12 Cancel Addition                                               *
*                                                                    *
```

图 135 客户文件维护的增加方式提示屏

工作站用户输入客户号来删除一条记录，如图 136：

```
*                                                                    *
*    DELETE   MODE                                                   *
*    22:32:55                CUSTOMER FILE MAINTENANCE          9/30/94    *
*                                                                    *
*                         00011   <--Enter Customer Number           *
*                                                                    *
*    F3 End Job     F5 Add     F6 Update     F7 Delete      F8 Display      *
*                                                                    *
```

图 136 客户文件维护的删除方式提示屏

工作站用户输入客户号来更新一条记录，如图 137：

```
*                                                                    *
*    UPDATE   MODE                                                   *
*    22:33:17                CUSTOMER FILE MAINTENANCE          9/30/94    *
*                                                                    *
*                         00010   <--Enter Customer Number           *
*                                                                    *
*    F3 End Job     F5 Add     F6 Update     F7 Delete      F8 Display      *
```

图 137 客户文件维护的更新方式提示屏

18.4 用 ZIP 码查找

下面解释了 WORKSTN 子文件的处理过程（仅显示）。

子文件被用来显示所有与指定的 ZIP 码相匹配的记录。

18.4.1 CUSMSTL2：逻辑文件的 DDS

```
A*********************************************************************
A* 文 件 名：CUSMSTL2                                                 *
A* 有关程序：SCHZIP                                                  *
A* 有关文件：CUSMST（物理文件）                                       *
A* 说     明：  这是逻辑文件 CUSMSTL2，它的记录格式为 CMLREC2，       *
A*              它是用 ZIP 做的 CUSMST 的视图。                       *
A*********************************************************************
A          R CMLREC2                   PFILE(CUSMST)
A            ZIP
A            NAME
A            ARBAL
A          K ZIP
```

图 138　逻辑文件 CUSMSTL2 的 DDS

该 DDS 描述了一个记录格式：CMLREC2，由 ZIP 做键字的逻辑文件 CUSMSTL2 是基于物理文件 CUSMST，它由 PFILE 键字指出，由逻辑文件建立的记录格式仅包括在逻辑文件 DDS 中指定的那些字段，其他的字段除外。

18.4.2 SZIPMENU：显示文件的 DDS

```
A*********************************************************************
A* 文 件 名：SZIPMENU                                                 *
A* 有关程序：SCHZIP                                                  *
A* 有关文件：CUSMSTL2（逻辑文件）                                     *
A* 说     明：这是一个显示文件，它有 6 个记录格式。                   *
A*********************************************************************
A                                      REF(CUSMSTL2)
A                                      CHGINPDFT(CS)
A                                      PRINT(QSYSPRT)
```

```
A                                      INDARA
A                                      CA03(03 'END OF JOB')
A          R HEAD
A                                      OVERLAY
A                                    2  4TIME
A                                      DSPATR(HI)
A                                    2 28'CUSTOMER SEARCH BY ZIP'
A                                      DSPATR(HI RI)
A                                    2 70DATE
A                                      EDTCDE(Y)
A                                      DSPATR(HI)
A          R FOOT1
A                                   23  6'ENTER - Continue'
A                                      DSPATR(HI)
A                                   23 29'F3 - End Job'
A                                      DSPATR(HI)
A          R FOOT2
A                                   23  6'ENTER - Continue'
A                                      DSPATR(HI)
A                                   23 29'F3 - End Job'
A                                      DSPATR(HI)
A                                   23 47'F4 - RESTART ZIP CODE'
A                                      DSPATR(HI)
A          R PROMPT
A                                      OVERLAY
A                                    4  4'Enter Zip Code'
A                                      DSPATR(HI)
A            ZIP       R     Y  I  4 19DSPATR(CS)
A                                      CHECK(RZ)
A  61                                  ERRMSG('ZIP CODE NOT FOUND' +
A                                      61)
A          R SUBFILE                   SFL
A            NAME      R           9  4
A            ARBAL     R           9 27EDTCDE(J)
A          R SUBCTL                    SFLCTL(SUBFILE)
A  55                                  SFLCLR
A N55                                  SFLDSPCTL
A N55                                  SFLDSP
```

```
A                                       SFLSIZ(13)
A                                       SFLPAG(13)
A                                       ROLLUP(95 'ROLL UP')
A                                       OVERLAY
A                                       CA04(04 'RESTART ZIP CDE')
A                                   4  4'Zip Code'
A            ZIP        R        O  4 14DSPATR(HI)
A                                   7  4'Customer Name'
A                                       DSPATR(HI UL)
A                                   7 27'A/R Balance'
A                                       DSPATR(HI UL)
```

图 139　显示文件 SZIPMENU 的 DDS

显示文件 SZIPMENU 的 DDS 包含六个记录格式：HEAD、FOOT1、FOOT2、PROMPT、SUBFILE 和 SUBCTL。

记录格式 PROMPT 请求用户输入一个 ZIP 码，如果这个 ZIP 码在文件中没找到，将给出一个出错信息，用户可按 F3 键，它置 03 指示器为 ON，结束程序。

记录格式 SUBFILE 必须在子文件控制记录格式 SUBCTL 的紧前面，用键字 SFL 定义的子文件记录格式描述了记录中的每个字段，并且指定了第一条记录在显示屏上出现的位置（这里，在第 9 行）。

子文件控制记录格式包括下面特别的键字：

- SFLCTL 标识了这个格式是一个控制记录格式并且命名了相关的子文件记录格式。
- SFLCLR 描述了子文件什么时候要清除已存在的记录（当指示器 55 为 ON 时），这个键字需要增加附加的显示。
- SFLDSPCTL 指出什么时候显示子文件控制记录格式（当指示器 55 为 OFF 时）。
- SFLDSP 指示什么时候显示子文件（55 指示器为 OFF 时）。
- SFLSIZ 指定了子文件总的大小，在这个例子中，子文件的大小为 13 个记录，在第 9～21 行显示。
- SFLPAG 定义了一页上的记录数，在本例中，页大小与子文件大小相同。
- ROLLUP 指出当用屏幕上翻功能时程序中的指示器 95 设为 ON。键字 OVERLAY 定义了这个子文件控制记录格式是一个复盖格式，该格式在不用 OS/400 系统清屏时可以写入。当重复使用相同的 ZIP 码进行查找时，可用 F4 键，（F4 这一用法允许下翻格式）。

18.4.3　SCHZIP：RPG 源码

```
*****************************************************************
* 程 序 名：  SCHZIP                                            *
```

```
            * 有关文件： CUSMSTL2（逻辑文件）                              *
            *           SZIPMENU（工作站文件）                             *
            * 说     明： 这个程序给出用工作站子文件处理客户文件的查询例子， *
            *             它提示用户输入 ZIP 码，由它来显示有关客户的记录，  *
            *           Rool up 键用来开始另一页。F3 结束程序。             *
            ****************************************************************
           FCUSMSTL2  IF   E           K DISK
           FSZIPMENU  CF   E             WORKSTN SFILE(SUBFILE:RECNUM)

           C     CSTKEY        KLIST
           C                   KFLD                    ZIP
 ****************************************************************
*    主程序行                                                    *
****************************************************************
           C                   WRITE     FOOT1
           C                   WRITE     HEAD
           C                   EXFMT     PROMPT
           C*
           C                   DOW       NOT *IN03
           C                   MOVE      *OFF          *IN61
           C     CSTKEY        SETLL     CMLREC2                          20
           C                   IF        NOT *IN20
           C                   MOVE      *ON           *IN61
           C                   ELSE
           C                   EXSR      SFLPRC
           C                   END
           C                   IF        (NOT *IN03) AND (NOT *IN04)
           C                   IF        NOT *IN61
           C                   WRITE     FOOT1
           C                   WRITE     HEAD
           C                   ENDIF
           C                   EXFMT     PROMPT
           C                   ENDIF
           C                   ENDDO
           C*
           C                   SETON                                      LR
 ****************************************************************
            *    子程序—SFLPRC                                          *
```

```
      *    用  途—处理子文件和显示                                               *
 *********************************************************************
          C     SFLPRC         BEGSR
          C     NXTPAG         TAG
          C                    EXSR       SFLCLR
          C                    EXSR       SFLFIL
          C     SAMPAG         TAG
          C                    WRITE      FOOT2
          C                    WRITE      HEAD
          C                    EXFMT      SUBCTL
          C                    IF         *IN95
          C                    IF         NOT *IN71
          C                    GOTO       NXTPAG
          C                    ELSE
          C                    GOTO       SAMPAG
          C                    ENDIF
          C                    ENDIF
          C                    ENDSR
*********************************************************************
*    子程序—SFLFIL                                    *
*    用  途—文件的子文件                                       *
*********************************************************************
          C     SFLFIL         BEGSR
          C                    DOW        NOT *IN21
          C     ZIP            READE      CMLREC2                        71
          C                    IF         *IN71
          C                    MOVE       *ON           *IN21
          C                    ELSE
          C                    ADD        1             RECNUM
          C                    WRITE      SUBFILE                        21
          C                    ENDIF
          C                    ENDDO
          C                    ENDSR
*********************************************************************
*    子程序—SFLCLR                                            *
*    用  途—清除子文件记录                                 *
*********************************************************************
          C     SFLCLR         BEGSR
```

```
C                   MOVE      *ON           *IN55
C                   WRITE     SUBCTL
C                   MOVE      *OFF          *IN55
C                   MOVE      *OFF          *IN21
C                   Z-ADD     *ZERO         RECNUM          5 0
C                   ENDSR
```

图 140　模块 SCHZIP 的源码

文件描述规范表中指出了用来查找的磁盘文件及所用的显示文件(SZIPMENU)。WORKSTN 文件的 SFILE 键字指出记录格式(SUBFILE)是文件使用的，相关的记录号字段(RECNUM)控制访问子文件中哪一个记录。

程序显示 PROMPT 记录格式并等待工作站用户的请求，按 F3 置 03 指示器为 ON，结束程序，ZIP 码(ZIP)用操作码 SETLL 来定位 CUSMSTL2 文件，注意：SETLL 操作中用记录格式名 CMLREC2，而不用文件名 CUSMSTL2。如果没有找到记录，将出现错误信息。

SFLPRC 子程序控制子文件的执行。清除、填充和显示，在子程序 SFLCLR 中准备子文件实现另外请求，如果指示器 55 为 ON，在显示上没有动作需求，但清除了子文件记录的主存贮区，SFLFIL 子程序用记录写满子文件，从 CUSMSTL2 文件中读一个记录，如果 ZIP 码相同，记录数(RECNUM)增加并把该记录写到子文件中去，这个子程序重复执行直到写满子文件(WRITE 操作上的指示器 21 为 ON 时)或者文件 CUSMSTL2 出现了文件末(READ 操作的指示器 71 为 ON)，当子文件写满或出现文件末时，子文件 EXPMT 操作按子文件控制记录格式写到显示屏幕上，用户查阅该显示并决定：

- 按 F3 键结束程序；
- 按 F4 重新键入 ZIP 码，PROMPT 记录格式不显示，有相同的 ZIP 码的子文件显示。
- 按 ROLLUP 键看另一页，如果 CUSMSTL2 文件出现文件末，当前面重新显示，否则清除子文件并显示下一页；
- 按 ENTER 键可用另一个 ZIP 码继续进行，显示 PROMPT 记录格式，用户可以键入一个 ZIP 码或结束程序。

在图 141 中，用户在 PROMPT 格式中输入 ZIP 码。

```
*                                                                *
*                                                                *
*    22:34:38              CUSTOMER SEARCH BY ZIP          9/30/94    *
*                                                                *
*    Enter Zip Code 11201                                        *
*                                                                *
*      ENTER - Continue       F3 - End Job                       *
*                                                                *
```

图 141　用 ZIP 码检索的提示屏幕

子文件写到屏幕上如图 142。

```
*                                                                    *
*                                                                    *
*    22:34:45              CUSTOMER SEARCH BY ZIP            9/30/94   *
*                                                                    *
*    Zip Code  11201                                                 *
*                                                                    *
*    Customer Name          A/R Balance                              *
*                                                                    *
*    Rick Coupland               300.00                              *
*    Mikhail Yuri                150.00                              *
*    Karyn Sanders                 5.00                              *
*                                                                    *
*      ENTER - Continue      F3 - End Job      F4 - RESTART ZIP CODE   *
*                                                                    *
```

图 142　用 ZIP 码检索屏幕

18.5 用姓名检索和查询

下面解释了 WORKSTN 子文件的处理过程（按选择显示），子文件被用来显示所有与指定的客户名相匹配的记录，然后允许用户从子文件中选择，这样可以显示客户的其它信息。

18.5.1　CUSMSTL3 逻辑文件的 DDS

```
A*********************************************************************
A* 文 件 名：CUSMSTL3                                              *
A* 有关程序：SCHNAM                                                *
A* 有关文件：CUSMST                                                *
A* 说　　明：这是逻辑文件 CUSMSTL3，记录格式名为 CUSREC，它是     *
A*           由 NAME 对 CUSMST 的视图。                            *
A*********************************************************************
A          R CUSREC                    PFILE(CUSMST)
A          K NAME
```

```
A*
```

图 143　逻辑文件 CUSMSTL3 的 DDS

用在这个程序中的数据库文件的 DDS 定义了一个名为 CUSREC 的记录格式，并定义 NAME 字段为键字字段。

18.5.2　SNAMMENU：显示文件的 DDS

```
A*********************************************************************
A* 文 件 名：SNAMMENU                                              *
A* 有关程序：SCHNAM                                                *
A* 有关文件：CUSMSTL3（逻辑文件）                                          *
A* 说　　明：这是一个显示文件，它有 7 个记录格式。                          *
A*********************************************************************
A                                      REF(CUSMSTL3)
A                                      CHGINPDFT(CS)
A                                      PRINT(QSYSPRT)
A                                      INDARA
A                                      CA03(03 'END OF JOB')
A          R HEAD
A                                      OVERLAY
A                                    2  4TIME
A                                      DSPATR(HI)
A                                    2 25'CUSTOMER SEARCH & INQUIRY BY
                                         ANAME'
A                                      DSPATR(HI UL)
A                                    2 70DATE
A                                      EDTCDE(Y)
A                                      DSPATR(HI)
A          R FOOT1
A                                   23  6'ENTER - Continue'
A                                      DSPATR(HI)
A                                   23 29'F3 - End Job'
A                                      DSPATR(HI)
A          R FOOT2
A                                   23  6'ENTER - Continue'
A                                      DSPATR(HI)
```

```
A                                      23 29'F3 - End Job'
A                                         DSPATR(HI)
A                                      23 47'F4 - Restart Name'
A                                         DSPATR(HI)
A          R PROMPT
A                                         OVERLAY
A                                       5  4'Enter Search Name'
A                                         DSPATR(HI)
A            SRCNAM    R        I  5 23REFFLD(NAME CUSMSTL3)
A                                         DSPATR(CS)
A          R SUBFILE                      SFL
A                                         CHANGE(99 'FIELD CHANGED')
A            SEL           1A  B  9  8DSPATR(CS)
A                                         VALUES(' ' 'X')
A            ZIP       R        O  9 54
A            CUST      R        O  9 43
A            NAME      R        O  9 17
A          R SUBCTL                       SFLCTL(SUBFILE)
A                                         SFLSIZ(0013)
A                                         SFLPAG(0013)
A  55                                     SFLCLR
A N55                                     SFLDSPCTL
A N55                                     SFLDSP
A                                         ROLLUP(95 'ROLL UP')
A                                         OVERLAY
A                                         CA04(04 'RESTART SEARCH NAME')
A                                       5  4'Search Name'
A            SRCNAM    R        O  5 17REFFLD(NAME CUSMSTL3)
A                                         DSPATR(HI)
A                                       7  6'Select'
A                                         DSPATR(HI)
A                                       8  6' "X"       Customer Name '
A                                         DSPATR(HI)
A                                         DSPATR(UL)
A                                       8 42' Number     Zip Code   '
A                                         DSPATR(HI)
A                                         DSPATR(UL)
A          R CUSDSP
```

```
A                                      OVERLAY
A                                      CA04(04 'RESTART ZIP CDE')
A                                   6 25'Customer'
A            CUST           5S 00   6 35DSPATR(HI)
A                                   8 25'Name'
A            NAME           20A  0  8 35DSPATR(HI)
A                                  10 25'Address'
A            ADDR1          20A  0 10 35DSPATR(HI)
A            ADDR2          20A  0 11 35DSPATR(HI)
A                                  13 25'City'
A            CITY           20A  0 13 35DSPATR(HI)
A                                  15 25'State'
A            STATE           2A  0 15 35DSPATR(HI)
A                                  15 41'Zip Code'
A            ZIP             5S 00 15 50DSPATR(HI)
A                                  17 25'A/R Balance'
A            ARBAL          10Y 20 17 42DSPATR(HI)
A                                      EDTCDE(J)
```

图 144　显示文件 SNAMMENU 的 DDS

显示文件 SNAMMENU 的 DDS 包括七个记录格式：HEAD、FOOT1、FOOT2、PROMPT、SUWBFILE、SUBCTL 和 CUSDSP。

PROMPT 记录格式要用户键入一个 ZIP 码和检索的名字，如果没有任何键入内容，显示从文件就开始，用户可按 F3 键，相应指示器 03 为 ON，程序结束。

子文件记录格式 SUBFILE 后必须紧跟子文件控制记录格式 SUBCTL，子文件记录格式使用键字 SFL，它描述了记录中的每个字段，并且规定了出现在显示上第一个记录的位置（在第 9 行），子文件控制记录格式包括下面一些特别的键字：

- SFLCTL 标识了这个格式是一个控制记录格式并且命名了相关的子文件记录格式。
- SFLCLR 描述了子文件什么时候要清除已存在的记录（当指示器 55 为 ON 时），这个键字需要增加附加的显示。
- SFLDSPCTL 指出什么时候显示子文件控制记录格式（当指示器 55 为 OFF 时）。
- SFLDSP 指示什么时候显示子文件（55 指示器为 OFF 时）。
- SFLSIZ 指定了子文件总的大小，在这个例子中，子文件的大小为 13 个记录，在第 9～21 行显示。
- SFLPAG 定义了一页上的记录数，在本例中，页大小与子文件大小相同。
- ROLLUP 指出当用屏幕上翻功能时程序中的指示器 95 设为 ON。键字 OVERLAY 定义了这个子文件控制记录格式是一个复盖格式，该格式在没用 OS/400 系统清屏时可以写

入。当重复使用相同的名字进行查找时，可用 F4 键，（F4 这一用法允许下翻格式）。

CUSDSP 记录格式显示选择了的客户的信息。

18.5.3 SCHNAM：RPG 源码

```
     ******************************************************************
      * 程 序 名：SCHNAM                                              *
     * 有关文件：CUSMSTL3（逻辑文件）                                  *
     *          SNAMMENU（工作站文件）                                *
     * 说     明：这个程序给出用工作站子文件处理客户主文件查询的例子 ，  *
     *           它提示用户输入客户名，并以它用 SETLL 定位 CUSMST3 文   *
     *           件，然后用子文件显示记录，要显示另一页，用 Rool up     *
     *           键。要显示详细的客户信息，在 CUSTOMER 边上输入'X',     *
     *           按执行键。用 F3 结束程序。                            *
     ******************************************************************
         FCUSMSTL3  IF   E           K DISK
         FSNAMMENU  CF   E             WORKSTN SFILE(SUBFILE:RECNUM)

         C     CSTKEY      KLIST
         C                 KFLD                   SRCNAM
         C     ZIPKEY      KLIST
         C                 KFLD                   NAME
 ******************************************************************
*     主程序行                                                     *
******************************************************************
         C                 WRITE     FOOT1
         C                 WRITE     HEAD
         C                 EXFMT     PROMPT
         C                 DOW       NOT *IN03
         C     CSTKEY      SETLL     CUSREC
         C                 EXSR      SFLPRC
         C                 EXSR      SFLCHG
         C                 IF        (NOT *IN03) AND (NOT *IN04)
         C                 WRITE     FOOT1
         C                 WRITE     HEAD
         C                 EXFMT     PROMPT
         C                 ENDIF
         C                 ENDDO
```

```
C*
C                   SETON                                          LR
*****************************************************************
 *    子程序—SFLPRC                                              *
 *    用  途—处理子文件和显示                                      *
*****************************************************************
C     SFLPRC        BEGSR
C     NXTPAG        TAG
C                   EXSR      SFLCLR
C                   EXSR      SFLFIL
C     SAMPAG        TAG
C                   WRITE     FOOT2
C                   WRITE     HEAD
C                   EXFMT     SUBCTL
C                   IF        *IN95
C                   IF        NOT *IN71
C                   GOTO      NXTPAG
C                   ELSE
C                   GOTO      SAMPAG
C                   ENDIF
C                   ENDIF
C                   ENDSR
*****************************************************************
 *    子程序—SFLFIL                                              *
 *    用  途—子文件                                              *
*****************************************************************
C     SFLFIL        BEGSR
C                   DOW       NOT *IN21
C                   READ      CUSREC                             71
C                   IF        *IN71
C                   MOVE      *ON           *IN21
C                   ELSE
C                   ADD       1             RECNUM
C                   MOVE      *BLANK        SEL
C                   WRITE     SUBFILE                            21
C                   ENDIF
C                   ENDDO
C                   ENDSR
```

```
**********************************************************************
*     子程序—SFLCLR                                          *
*     用  途—清子文件                                                *
**********************************************************************
          C     SFLCLR      BEGSR
          C                 MOVE      *ON           *IN55
          C                 WRITE     SUBCTL
          C                 MOVE      *OFF          *IN55
          C                 MOVE      *OFF          *IN21
          C                 Z-ADD     *ZERO         RECNUM          5 0
          C                 ENDSR
 **********************************************************************
           *     子程序—SFLCHG                                        *
           *     用  途—选择的客户记录                                    *
**********************************************************************
          C     SFLCHG      BEGSR
          C                 READC     SUBFILE                         98
          C                 IF        NOT *IN98
          C     ZIPKEY      CHAIN     CUSREC                          71
          C                 EXFMT     CUSDSP
          C                 ENDIF
          C                 ENDSR
```

图 145　SCHNAM 的源码

文件描述规范表指出要检索的磁盘文件和所用的显示文件(SNAMMENU)WORKSTN 文件所用的 SFILE 键字，指出记录格式 SUBFILE 做为子文件用，相关记录号字段 RECNUM 指出要检索子文件的哪个记录。

程序显示 PROMPT 记录格式并等待终端用户响应，F3 键设指示器 03 为 ON。它控制程序的结束，名字(NAME)做为键字由 SETLL 操作来定位 CUSMSTL3。注意，用在 SETLL 操作中的是记录格式名(CUSREC)而不是文件名(CUSMSTL3)。

SFLPRC 过程控制子文件的处理：清除、填充和显示，子文件准备好要用子例程 SFLCLR 做其它请求来用，如果指示器 55 为 ON，显示上没有其他的动作，但子文件记录的主存储域要被清除，SFLFIL 用记录来填充子文件。当从 CUSMSTL3 中读一个记录时，记录计数(RECNUM)要增加，且把记录写到子文件中，这个过程要一直做下去直到子文件满(WRITE 操作的指示器 21 为 ON)或者 CUSMSTL3 到了文件末，(READ 操作的指示器 71 为 ON)，子文件由 EXFMT 操作用子文件控制记录规定的格式写到屏幕上，用户接收这个显示并进行选择。

按 F3 键字结束程序。

要更新启动子程序按 F4 键，不显示 PROMPT 记录格式子文件用同一个名字再显示。

为显示另外一页用 ROLL UP 键，假如 CUSMSTL3 文件结束，当前页再次显示，否则，清除子文件并显示下一页。

要显示客户的详细信息要键入 X，并按执行键，用户也可按执行键返回到 PROMPT 屏，用 F4 再次显示子文件，或用 F3 结束程序。

图 146 用户键入一个客户名来回答初始提示。

```
*                                                                    *
*                                                                    *
*    22:35:26          CUSTOMER SEARCH & INQUIRY BY NAME       9/30/94    *
*                                                                    *
*    Enter Search Name  JUDAH GOULD                                  *
*                                                                    *
*      ENTER - Continue       F3 - End Job                           *
*                                                                    *
```

图 146　用名字检索和查询的提示屏幕

用户键入 X 来请求显示详细信息，如图 147。

```
*                                                                    *
*                                                                    *
*    22:35:43          CUSTOMER SEARCH & INQUIRY BY NAME       9/30/94    *
*                                                                    *
*    Search Name  JUDAH GOULD                                        *
*                                                                    *
*      Select                                                        *
*       "X"       Customer Name            Number     Zip Code       *
*        X        JUDAH GOULD              00012      70068          *
*                 JUDAH GOULD              00209      31088          *
*                                                                    *
*      ENTER - Continue       F3 - End Job      F4 - Restart Name    *
*                                                                    *
```

图 147　用名字检索和查询的信息屏幕

选择的客户的详细信息如图 148 所示，这时，用户可以选择相应的功能键来继续或结束查询。

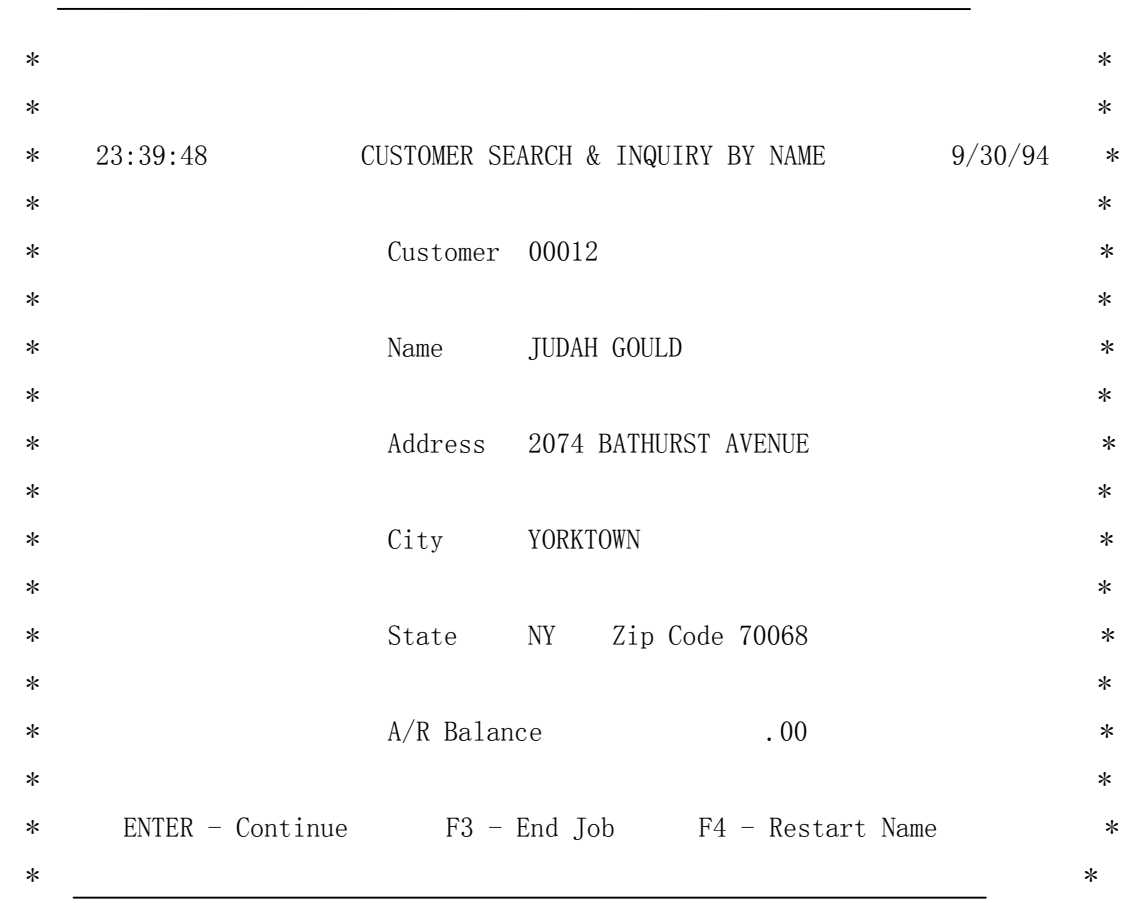


图 148　用名字检索和查询的详细信息。

附录 1.1 附录 A　OPM　RPG/400 和 ILE　RPG/400 之间的主要区别

下面列出了 OPM　RPG/400 与 ILE　RPG/400 编译程序的不同之处。
它们在语言定义上的不同处，参看《ILE RPG/400 参考》有关附录。

附录 1.1.1 编译

1.在 RPGⅣ中控制规范表中的 DATEDIT 与 DECEDIT 之间设有从属关系。
2.注意 ILE　RPG/400 的生成命令(CRTBNDRPG 和 CTRPGMOD)：
・指定 CVTOPT(*NONE)意思是忽略不被 RPG 支持的类型或属性的所有的外部描述的字段（变长字符及变长图形），这也意味着那些类型为图型，日期、时间和时间标记和在外部描述中指定的相同类型的字段带到程序中。
・在 CRTRPGPGM 命令上的参数 IGNDECERR 已被 ILE RPG/400 编译命令的参数 FIXNBR 所代替。
・有一个新的参数 TRUNCNBR，它控制是否允许数字溢出。
・在 RPGⅣ中没有自动报表特性或命令。
・不能从编译程序中请求一个 MI 列表。
3.在编译清单中，行号从 1 开始源语句或者生成规范表，每行增量为 1，源语句的 ID 是数字，即对于/COPY 和扩展加 DDS，行数不超过 AA000100。
4.RPGⅣ要求所有的编译指令在编译时数据之前出现，包括/TITLE，当 RPGⅣ遇到一个/TITLE 指令时，把它当作数据来处理，(RPGⅢ无论在源语句的什么地方遇到/TITLE 都把它当作编译指令处理)。
换算工具将移去它在编译时数据中遇到的/TITLE 规定。
5.RPGⅣ在数据结构中测试字段覆盖更精确，在某些情况下，ILE RPG/400 将发布一条信息，但同样情况 OPM 编译程序就可能不发布信息。

附录 1.1.2 运行

1.RPGⅣ不支持 FREE 操作。
2.某些 MCH 信息，可以出现在不在 OPM 之下的作业日志中，例如，MCH1202，这些信息的外表显示不出程序行为的变化。
3.如果使用非连接 API　QMHSNDPM 从程序发送信息，需要往堆栈偏移参数加 1 以便能表示堆栈中程序入口例程，这仅指 ILE 过程是一个用户入口过程，并且使用特别的值（*）来调用信息队列，且堆栈偏移量的值比 0 大。
4.ILE RPG/400 不对非例外结束的非 RPG 程序或过程解释非 0 或 1 的返回码。
5.当测试到递归时，OPM　RPG/400 显示询问信息 RPG8888，ILE　RPG 用逃逸信息 RNX8888，表示这种情况下不显示询问信息。

附录 1.1.3 调试和异常处理

1.在 RPGⅣ中不支持 DEBUG 操作；
2.当使用 ILE 源码调试设置断点时，不能用 RPG 标识符，子例程名或周期中的*GETIN 和*DETC 做断点；
3.OPM　RPG/400 和 ILE　RPG/400 的功能检查正常地写入作业日志，但在 ILE RPG/400 中，如果编写了一个错误指示器或者*PSSR 错误例程，那么不会出现功能检验。

既然*PSSR 能避免发生功能检查，那应当去掉所有删除功能检查的代码。

4.由于要得到 PSDS 中 80 个字节之后的信息开销较大，那么不用 PSDS 或 PSDS 少于 80 字节会大大改变 LR 为 ON 的调用性能。但这种情况下，如果没有编写 PSDS 或者 PSDS 字段长度没有包括这些字段，一个格式化转储就不包括在 PSDS 中程序开始运行的系统时间和日期有关的信息。

5.ILE　RPG/400 与 OPM　RPG/400 异常处理的比较看 3.2.1.1.1 内容。

附录 1.1.4　I/O

1.在 ILE　RPG/400 中你可以读一个打开文件中的一条记录来更新，用 SHARE(*YES)生成或覆盖时，能用打开同一文件做更新另一程序更新锁定的记录。

2.不能使用 MOVE 或 SETON 操作来修改指示器 MR(RPG 仅阻止对 MR 使用 SETON)。

3.文件描述规范表中的文件类型项不再指出那些必须在计算规范表中出现的 I/O 操作的类型。

例如，在 RPGⅢ中，如果定义一个更新文件，在程序中必定有 UPDAT 操作，这在 RPGⅣ中已不再必要，然而文件定义仍然必须与程序中出现的 I/O 操作相一致，所以如果在源码中使用了 UPDAT 操作，则文件必须定义为更新文件。

4.既使在文件描述规范表中指定了 COMMIT 键字。ILE RPG/400 也允许记录分块。

5.在 RPGⅣ中，一个为更新打开的文件也作为能删除的文件，不必用 DELETE 操作来使它有删除能力。

6.在 RPGⅣ 中，不必为用于多设备文件设备上编写实际用的数目，如果在文件描述规范表中指定了 MAXDEV(*FILE)，那么为 SAVEDS 和 SAVEIND 建立的存贮区域数根据文件能够处理的设备数来确定。(RPGⅣ文件描述规范表中的键字 SAVEDS、SAVEIND 和 MAXDEV 分别与 RPGⅢ的文件描述规范表中续行上的 SAVDS、IND 和 NUM 相对应。)

在 ILE RPG/400 中，不能指定能由程序得到的程序设备数与在设备文件中定义的最大设备数不同，OPM RPG/400 通过 NUM 选项允许这样。

7.在 ILE RPG/400 中，ACQ 和 REL 操作码能用于单设备文件。

8.在 ILE RPG/400 中，INFDS 中数据库相关反馈段中的相关的记录号和键字字段被每个做分块读的输入操作来更新。

9.在 OPM RPG/400 中，当发生引用限制错误时，状态码设为“01299”(I/O 错)，在 ILE RPG/400 中，状态码设为“01022”，“01222”或“01299”，它赖于所发生的引用限制类型：

• 如果数据管理不能分配一条记录，导致上面的错误发生，CPF502D 通知信息，ILE RPG/400 设状态码为“01222”，而 OPM RPG/400 设状态码为“01299”。

• 如果没有错误指示器或 INFSR 错误子例程，ILE RPG/400 将发生 RNQ1222 询问信息，而 OPM RPG/400 发生 RPG1299 询问信息，这两个信息的主要区别是 RNQ1222 允许你重作。

• 如果数据管理检测一个上述错误，导致发出 CPF503A 或 CPF50ZD 通知信息，ILE RPG/400 置状态码为“01022”，而 OPM RPG/400 置状态码为“01299”。

• 如果没有错误指示器或 INFSR 错误子例程，ILE RPG/400 将发出 RNQ1022 询问信息，而 OPM RPG 将发出 RPG1299 询问信息。

• 所有被数据管理测试的这些错误，导致数据管理发出一个逃逸信息使 OPM 和 ILE RPG/400 都设置状态码为“01299”。

10.ILE RPG/400 比 OPM RPG/400 更多地依靠数据管理的错误处理手段，这意味着在某些情况下将在 ILE RPG/400 程序的作业日志中发现某些错误信息，但是在 OPM RPG/400 就不一定有注意到在处理错误中的一些不同：

• 当对一个没有被前一次输入操作锁定的数据库文件作记录的更新操作时，ILE RPG/40 和 OPM RPG/400 都置状态码为“01211”，ILE RPG/400 测试到这种情况，数据管理发出一个 CPF501B 的通知信息并把它放在作业

日志中。

• 处理 WORKSTN 文件，试图对一个没有得到的或没定义的设备进行 I/O 操作时，ILE 和 OPM RPG/400 都置状态码为“01281”，ILE RPG/400 测试到这种情况，数据管理发出一个 CPF5068 逃逸信息并放到作业日志中。

11.当数据库文件作 READE、REDP 时，在 ILE 里是 READPE、SETLL 时或者由记录地址文件作有限顺序处理时，OPM RPG/400 使用*HEX 排列次序来做键字比较，这在访问路径上的字段内容与字段的实际内容不同时会产生非予期结果。

• ILE RPG/400 只对 3.1 版本之前的 DDM 文件使用中*HEX 排列次序做键字比较。详细内容见 4.3.7.1。

附录 1.1.5 字符字段中的 DBCS 数据

1.在 OPM RPG/400 中，控制规范表的 57 列允许你指定 RPG/400 编译程序是否扫瞄 DBCS 字符文字和常量，如果规定编译程序扫瞄透明文字，且如果一个由撇号开始跟着移出的字符文字使透明检测失败，这个文字表示一个不透明的文字。

ILE RPG/400 中，在控制规范表中没有选项来指定编译程序是否在字符文字上执行透明性校验。如果一个字符文字包括一个移出控制字符，不管移出字符在字符文字中的位置，移出字符就表示 DBCS 数据的开始，编译程序将做如下校验：

有没有与移出相匹配的移入字符(即，移出与移入控制字符要成对)。在移出与移入字符之间的字符是偶数个(最小为 2)。在 DBCS 数据中没有嵌入的移出。

如果不满足上面的条件，编译程序将发出一个诊断信息，并且这个文字将不被再解析，作为一个结果，如果在 OPM RPG 程序中没有由其编译程序完成和失败透明检查的字符文字，在 ILE RPG 中这样的程序将收到一个编译错误。

2.在 OPM RPG/400 中，如果在一个字符文字中有两个连续的撇号，在移出和移入控制字符内，如果字符文字不是透明文字，则这两个撇号被认为是一个撇号，事实上，字符文字不是一个透明的文字是由于：

• 字符文字不是以撇号跟一个移出开始。

• 字符文字使由编译程序执行透明检验失败。

• 用户没有规定由编译程序执行的透明性检验。

在 ILE RPG/400 中，如果在一个字符文字中有两个连续的撇号，在移出和移入控制字符内，这两个撇号不被认为是一个单独的撇号，在字符文字里的一副撇号，如果没有括在移出和移入中，则认为是一个单独的撇号。

如果有一个包含移出字符的字符文字，你不想解释为移出字符，那么，应指定该文字作为十六进制的文字，例如，如果有一个文字’A0B’，这里’0’表示移出，这个文字的编码为 x’C1OEC2’。

附录 1.2 附录 B 使用 RPGⅢ到 RPG Ⅳ的转换工具。

RPG Ⅳ的源码的样子明显不同于 SYS/38 环境的 RPGⅢ和 OPM RPG/400，例如，表中的输入项的位置已经改变并且表的可用类型也已经改变，RPG Ⅳ规范表与先前的格式是不兼容，为了使用 RPG Ⅳ的特色，要把 RPGⅢ和 RPG/400 应用程序中的源成员转换为 RPG Ⅳ源格式。

注：可以转换的源成员的有效类型是：RPG，RPT，RPG38，RPT38，SQLRPG，和空白，转换工具不支持 RPG36 和 RPT36，以及其它非 RPG 源成员的类型。

如果你想马上开始转换，请直接看附录 1.2.2“转换源码”，按一般用法说明来做。

附录 1.2.1 转换概述

通过 CL 命令(CVTRPGSRC)来调用辅助工具把源程序转换为 RPG Ⅳ的源码格式。

转换工具可以转换：

- 一个单独的成员；
- 在源物理文件中的所有成员；
- 在同一个文件中的有共同的成员名前缀的所有成员；

为了使转换发生问题的可能性最小，可以转换源码中包括的/COPY 成员。为了方便阅读源码，可有选择地用包括在被转换源码中的规范样板。

转换工具一行一行来转换每个源成员，每个成员转换之后，如果用命令指定了一个日志文件，它要用转换状态来更新日志文件，还可以获得一个转换报表，它包括象转换错误，/COPY 语句，CALL 操作和转换状态等信息。

转换工具假设源码对任何编译错误是自由的，这种情况下，它将成功地转换大多数源码，但在某些情况下，可能有一小部分代码，要手工转换，这些情况有些由转换工具标识出来，其他的要到编译转换好的源码时，才能检测到。看哪些是由转换工具标识的，可以用没转换的成员作为输入运行转换工具，并且规定转换报表且输出成员，有关内容请参考附录 1.2.5。

附录 1.2.1.1 文件考虑

转换辅助工具对文件成员进行操作，这部分介绍了使用转换工具时必须考虑有关文件方面的内容。

附录 1.2.1.1.1 源成员的类型

表 19 列出了各种源成员的类型，指出了什么样类型的成员能被转换，并且指出输出源成员的类型。

表 19　源成员类型以及它们的转换状态

源成员类型	转换？	转换的成员类型
RPG	Yes	RPGLE
RPG38	Yes	RPGLE
RPT	Yes	RPGLE
RPT38	Yes	RPGLE
‘空白’	Yes	RPGLE
RPG36	No	N/A
RPT36	No	N/A
SQLRPG	Yes	SQLRPGLE
Any Other Type	No	N/A

如果源成员的类型为‘空白’，转换工具假定它有一个 RPG 类型的成员，如一个自动报表源成员的类型为空，那么应当在转换它之前给成员设置正确的类型(RPT 或 RPT38)，这样，转换工具将自动地扩充自动报表源成员，使它能够被正确地转换，因为 ILE　RPG/400 不支持自动报表源成员，这个扩充是必要的。

关于自动报表源成员的转换详细资料，见附录 1.2.2.7 中的“转换自动报表源成员”。

附录 1.2.1.1.2 文件记录长度

我们建立转换源物理文件的记录长度为 112 个字符，这个记录按图 149 给出的 RPG Ⅳ的结构。安排这个长度的记录也适合编译清单中一行上的最大信息量。

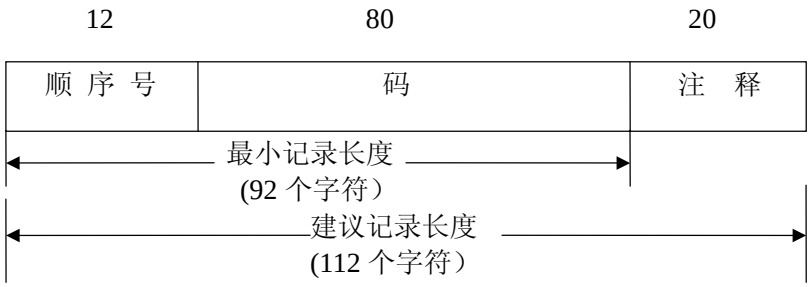


图 149　RPG Ⅳ记录长度分解

如果被转换的源文件的记录长度小于 92 个字符，那将发出错误信息并且停止转换，这是因为记录长度是不足以装下源码允许的 80 个字符，这样一些码很可能丢失。

附录 1.2.1.1.3 文件和成员名字

未转换的成员和转换后输出的成员如果它们在不同的文件或库中，仅能有相同的名字。

被转换的源成员的名称依赖于你是否转换一个或者几个成员，如果转换一个成员，转换后成员的名字缺省值是转换前的成员名，你当然可以指定一个不同的名字，如果转换一个文件中所有的源成员，或它们使用类属名中的一组成员，那么这些成员的名字自动地被与转换前的源成员名字相同。

规定转换后的文件、库和成员名是可选的，如果没指定这些名字，转换后的输出放在文件 QRPGLESRC 中，成员名与转换前的成员名相同。(将在库列表中检索文件 QRPGLESRC)。

附录 1.2.1.2 日志文件

转换工具用一个日志文件为每个源成员的转换提供状态跟踪报告，通过浏览日志文件，能够确定先前转换的状态，可以由用户编写的程序来访问日志文件，来执行进一步的处理，例如，编译和联接程序。

如果你规定一个日志文件是能更新的，那么它的记录格式必须与提供的 model 数据库文件 QARNCVTLG(在库 QRPGLE 中)的格式相匹配，附录 1.2.4.2 中的图 156 给出了这个文件的 DDS。使用 CRTDUPOBJ 命令在自己的库中(如 MYLIB)建立这个模块的拷贝。日志文件的名为 QRNCVTLG，是转换工具缺省的日志文件名。

```
CRTDUPOBJ       OBJ(QARNCVTLG)    FROMLIB(QRPGLE)
                OBJTYPE(*FILE)    TOLIB(MYLIB)
                NEWOBJ(QRNCVTLG)
```

对转换工具访问的日志文件，你必须有目标管理、操作和增加的授权。

详细的资料见附录 1.2.4.2 中的“使用日志文件”。

附录 1.2.1.3 转换辅助工具的要求

要使用转换工具，下列授权：

- 对命令 CVTRPGSRC 有*USE 权利；
- 对有源文件和源成员的库有*USE 权利；
- 对给将要放源文件和转换后源成员的新库有*CHANGE 权利；
- 对转换工具使用的日志文件有目标管理、操作和增加的权利。

除了目标授权要求之外，还可能有另外的存储区要求，一般来说，每个转换后源程序，比原来成员大约大25%。为了使用转换工具要有足够的存贮区来存贮转换的源文件。

附录 1.2.1.4 转换工具不能做什么

转换工具不支持从 RPGⅣ格式转回到 RPGⅢ或 RPG/400 格式。

RPG Ⅳ编译器不支持在编译时成员的自动转换。

转换工具不支持 RPGⅡ源程序到 RPG Ⅳ源格式的转换，但可先用由 RPGⅡ转到 RPGⅢ的转换工具，后进行由 RPGⅢ到 RPGⅣ的转换。

转换工具不更新设计源码，除非有需要，(例如，条件指示器的数量)。

转换工具不建立文件，日志文件和输出文件在运行前必须存在。

附录 1.2.2 转换源码

这一部分讨论怎样把源程序转换为 RPGⅣ格式，解释命令 CVTRPGSRC 的用法，它启动转换工具，并使用做转换。转换源码为 RPGⅣ格式，按下面步骤做：

1.如果在控制规范表中使用数据域，必须用 RPGⅣ格式创建新的一个，详细资料参考《ILE RPG/400 参考》中控制规范表部分。

2.如果需要，建立一个日志文件。

除非指定 LOGFILE(*NONE)，必定有一个日志文件让转换工具访问。如果没有，可以通过 CRTDUPOBJ 命令建立一个，详细资料见附录 1.2.1.2 中的“日志文件”和附录 1.2.4.2 中的“使用日志文件”。

3.为转换源成员建立文件。

转换工具不建立任何文件，在运行 CVTRPGSRC 命令前，必须为转换源文件建立输出文件，建立输出文件名为 QRPGLESRC 记录长度 112 个字符，详细文件信息见附录 1.2.1.1 中的“文件考虑”。

4.用命令 CVTRPGSRC 转换源码。

必须输入要转换的文件和成员名，如果使用缺省值，将在文件 QRPGLESRC 中有一个转换的成员，它的名将与未转换的源成员的名字一致。

转换后源成员中不扩充/COPY 成员，除非它的类型为 RPT 或 RPT38，也生成一个转换报表。

详细信息见附录 1.2.2.1。

5.检测日志文件或错误报表，详细资料见附录 1.2.4。

6.如果有错误，改正它并返回步骤 4.。

7.如果没有错误，建立程序，关于建立 ILE RPG/400 程序的详细资料，见第 5 章 2 节，“用 BNDRPG 命令建立程序”。

8.如果转换的源成员仍然有编译问题，大部分由于源成员含有/COPY 编译指令，有两种选择来改正这种问题：

a.指定 EXPCPY(*YES)重新转换源成员，把拷贝成员扩充进转换源成员中。

b.依据编译清单，手工改正其余的错误。

其他考虑参考附录 1.2.5。

9.一旦转换的源成员编译成功，在包装成产品之前要重新测试。

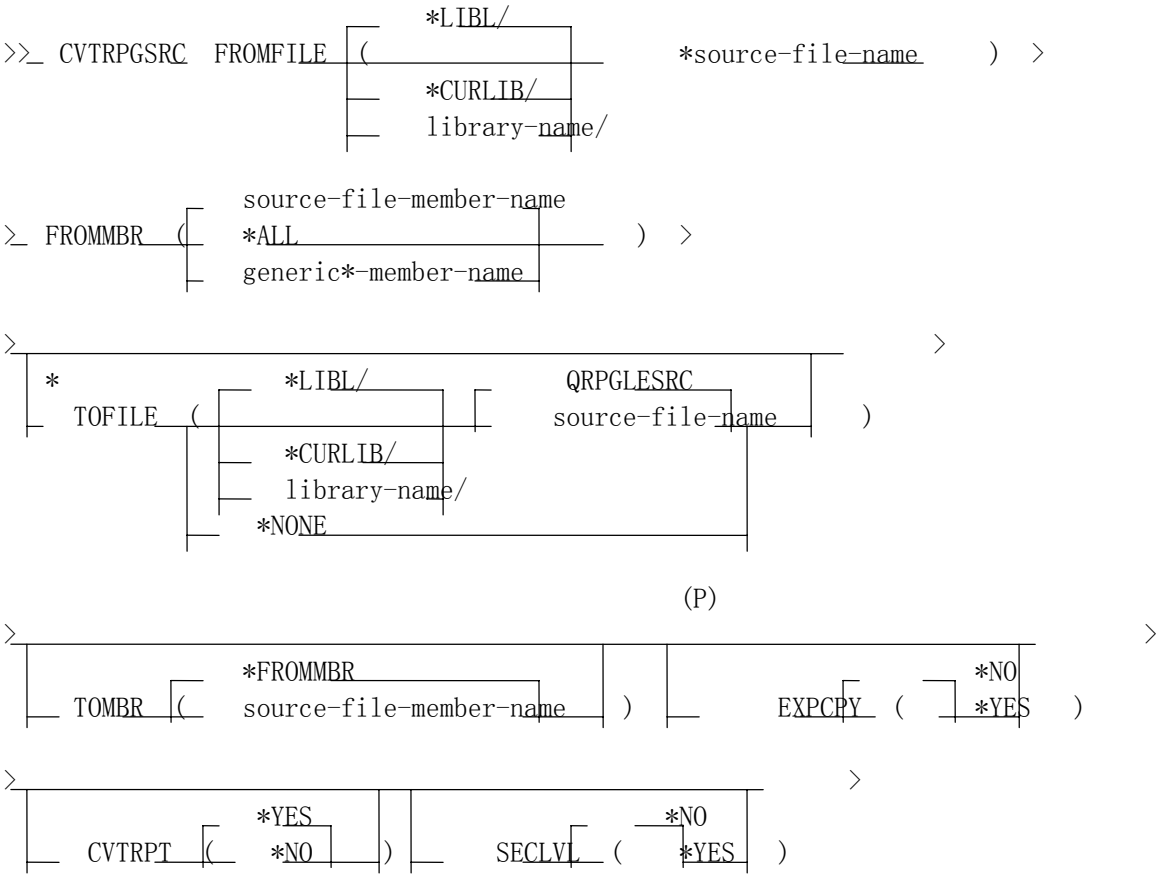
附录 1.2.2.1 CVTRPGSRC 命令

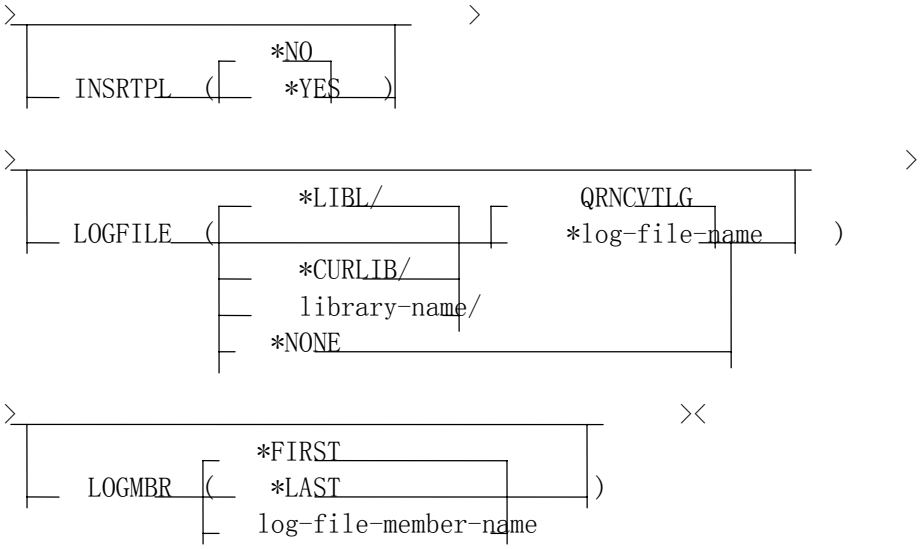
为了把 RPGⅢ和 RPG/400 源码转换成新的 RPGⅣ格式，把使用命令 CVTRPGSRC 来启动转换工具。表 20 显示了命令及按功能给出的参数。

表 20　CVTRPGSRC 参数和功能的缺省值

程序标识		
	FROMFILE	指出要被转换的 RPG 源码所在的库和文件名
	FROMMBR	指出要被转换的源成员
	TOFILE(*LIBL/QRPGLESRC)	指出转换输出的库的文件名
	TOMBR(*FROMMBR)	标识转换源码的文件成员名
转换处理		
	TOMBR	如果指定*NONE，则不保存文件成员
	EXPCPY(*NO)	确定/COPY 语句是否包括在转换输出中
	INSRTPL(*NO)	显示规范的样板是否包括在转换输出中
转换反馈		
	CVTRPG(*YES)	确定是否产生转换报表
	SECLVV(*NO)	确定是否包括二级的信息文本
	LOGFILE(*LIBL/QRNCVTLG)	指出审查报表的日志文件
	LOGMBR(*FIRST)	指出日志文件的哪一成员使用审查报表

CVTRPGSRC 命令的语法如下所示：





注：(P)这一点之前的参数都是位置参数。语法图中给出了参数和它们的可能值，如果需要提示，键入 CVTRPGSRC 并按 F4 键，出现 CVTRPGSRC 屏，列出了参数，并提供缺省值，对显示参数的解释，把光标定在此参数上，按 F1 键，对参数的扩展帮助按 F1 键后再按后 F2 键。

FROMFILE

指定包含要转换的 RPGⅢ或 RPG/400 源码源文件名，及所在库名，这是一个必需的参数；没有缺省的文件名。

source-file-name(源文件名)

输入包含要转换的源成员的源文件名。

*LIBL(库列表)

系统在库列表中寻找存有源文件的库。

*CVRLIB(当前库)

用当前库查找源文件，如果你没指定一个当前库，那么库 QGPL 被使用。

library-name(库名)

输入存有源文件的库名。

FROMMBR

规定要转换的成员的名字，这是一个必需的参数，没有缺省的成员名。

能被转换的有效的原成员类型是：RPG、RPT、RPG38、RPT38、SQLRPG 和空类型，转换 源码命令不支持 RPT36、RPG36 类型和其他的非 RPG 源成员类型(例如，CLP 和 TXT)。

source-file-member-name(源文件成员名)

输入要转换的源成员名。

*ALL

命令转换指定的源文件中的所有成员

generic*-member-name(类属成员名)

输入有相同前缀(在前缀后面加‘*’(星号)，表示的成员的类属名)，命令转换规定源文件中的有这个类属名的所有成员，例如，规定 FORMMBR(PR*)将转换名用‘PR’开始的所有成员。

有关类属名的详细资料见《CL 程序员指南》。

TOFILE

指定它包含转换过的源成员的源文件名及所在的库名，转换进的源文件必须存在它的记录长度为 112 个字符：其中 12 个位置给顺序号和日期，80 个给编码，20 个给注释。

QRPGLESRC

缺省的源文件 QRPGLESRC 包含转换后源成员。

*NONE

设有转换的成员生成，TOMBR 参数值可以忽略，也可规定 CVTRPT(*YES)或者转换将立即结束。这个参数允许不建立转换后源成员而找某些潜在的问题。

source-member-name(源文件名)

输入包含转换后的源成员的源文件名。

如果 TOFILE 库名与 FROMFILE 的库名相同，TOFILE 源文件名必须不同于 FROMFILE 的源文件名。

*LIBL(库列表)

系统检索找库列表来查找存储转换后的源文件的库。

*CURLIB(当前库)

当前库被用来寻找转换后的源文件，如果没指定一个当前库，则使用库 QGPL。

library-name(库名)

输入放转换后源文件的库名。

TOMBR

指定转换后源成员的名字，如果在参数 FROMMBR 上指定为*ALL 或 generic*，那么 TOMBR 必须等于*FROMMBR。

*FROMMBR

用在 FROMMBR 参数中指定的成员名作为转换后的源成员名，如果规定了 FROMMBR(*ALL)，那么 FROMFILE 中的所有源成员都被转换，转换后的源成员与那些先前的源成员有相同的名字，如果在 FROMMBR 参数中指定类属名，那么转换所有名字有相同的前缀的成员，转换后的源成员与那原先类属名的源成员有相同的名字。

sruce-file-member-name(源文件成员名)

输入转换后的源成员名，如果成员不存在，它被建立。

EXPCPY

指定/COPY 成员是否要扩充到转换后的源成员中，仅当有与/COPY 成员有关的转换问题时，指定 EXPCPY(*YES)。

注：如果成员的类型为 RPT 或 RPT38，EXPCPY(*YES)或 EXPCPY(*NO) 没有影响，因为自动报表程序将总是扩充/COPY 成员。

*NO

不把/COPY 文件成员扩充到转换后的源文件中。

*YES

把/COPY 文件成员扩充到转换后的源文件中。

CVTRPT

指定是否打印转换报表。

*YES

打印转换报表。

*NO

不打印转换报表。

SECLVL

规定在信息概括部分中的转换报表里是否打印二级信息文本。

*NO

在转换报表里不打印二级信息文本。

*YES

在转换报表里打印二级信息文本。

INSRTPL

指定 ILE　RPG/400 的规范样板(H-，F-，D-，I-，C-和或者 0-规范表)，是否插入到转换后的源成员中，缺省值显*NO。

*NO

规范样板不插入到转换后的源成员中去。

*YES

规范样板插入到转换后的源成员中去，每个样板插入到相应表的开始部分。

LOGFILE

指定用来跟踪转换的信息的日志文件名，除非规定了*NONE，否则必须有一个日志文件，文件必须已经存在，并且它必须是一个数据物理文件，使用 CRTDUPOBJ 命令，其中参数“form object”用库 QRPGLE 中文件 QARNCVTLG，参数“New Object”用你自己的库，文件用 QRNCVTLG。

QRNCVTLG

缺省的日志文件 QRNCVTLG，用来收集转换的信息。

*NONE

转换信息不写入日志文件中。

log-file-name(日志文件名)

输入用来做跟踪转换信息的日志文件名。

*LIBL

系统检索库列表来查找存在日志文件的库。

library-name(库名)

输入放日志文件的库名。

LOGMBR

指定了用来做跟踪转换信息的日志文件的成员，新的信息加到规定的日志文件成员中已有的数据中去。如果日志文件没有成员，那么生成一个与日志同名的成员。

*FIRST

使用在指定的日志文件中的第一个成员。

*LAST

使用在指定的日志文件中的最后一个成员。

log-file-member-name(日志文件成员名)

输入用来做跟踪转换信息的日志文件成员名。

附录 1.2.2.2 使用缺省方式转换一个成员

能够用 CVTRPGSRC 支持的缺省值来转换成员，例如：

CVTRPGSRC　FROMFILE(文件名)FROMMBR(成员名)

它将对规定的源成员进行转换，输出放在文件 QRPGLESRC 中，使用库列表中含有此文件的那个库，不扩充/COPY 成员，不插入规范样板，产生转换报表，更新日志文件 QRNCVTLG。

注：文件 QRPGLESRC 和 QRNCVTLG 必须已经存在。

附录 1.2.2.3 转换文件中的所有的成员

在命令 CVTRPGSRC 中指定 FROMMBR(*ALL)和 TOMBR(*FROMMBR)，就能转换一个源物理文件中的所有成员。转换工具要转换指定文件中的所有成员。如果其中一个成员转换失败，转换过程仍然继续进行。例如，如果把文件 QRGPSRC 中的所有成员转换到文件 QRPGLESRC 中，输入：

```
CVTRPGSRC       FROMFIEL (OLDRPG/QRPGSRC)
                FROMMBR(*ALL)
                TOFILE(NEWRPG/QRPGLESRC)
                TOMBR(*FROMMBR)
```

这个命令转换库 OLDRPG 中文件 QRPGSRC 中的所有成员，在库 NEWRPG 中的源物理文件 QRPGLESRC 中建立新成员。如果愿意在同一个文件中保留所有源码(DDS 源码，RPG 源码等)，通过指定 FROMMBR(*ALL)，用一步转换 RPG 源成员，转换工具仅转换有效的 RPG 类型的成员(见表 19，附录 1.2..1.1.1 中)。

附录 1.2.2.4 转换一个文件中的某些成员

如果仅需要转换源物理文件中的某些成员，它们的成员名有一个共同的前缀，那么成员名使用类属名。例如，如果跟着想转换所有的‘PAY’前缀的成员，应输入：

```
CVTRPGSRC       FROMFILE(OLDRPG/QRPGSRC)
                FROMMBR(PAY*)
                TOFILE(NEWRPG/QRPGLESRC)
                TOMBR(*FROMMBR)
```

这个命令转换了库 OLDRPG 中源物理文件 QRPGSRC 中的所有名字以‘PAY’打头的成员，新的成员建立在库 NEWRPG 中的源物理文件 QRPGLESRC 中。

附录 1.2.2.5 做一个尝试性转换

可以对猜测到有转换问题的源成员做尝试性转换，将会得到一个转换报表，它标识出某些转换错误。

例如，对源成员 PAYROLL 做尝试性的转换，输入：

```
CVTRPGSRC       FROMFILE(OLDRPG/QRPGSRC)
                FROMMBR(PAYROLL)
                TOFILE(*NONE)
```

TOMBR 参数指定为*FROMMBR，既然它是缺省值，就不必去指定它，除非要改变缺省值，参数 CVTRPT 应指定为*YES，这也是缺省值，如果不为*YES，那么转换将立即停止。

使用 TOFILE(*NONE)参数告诉转换工具不生成转换的成员，但仍然产生转换报表，见附录 1.2.4。

附录 1.2.2.6 得到转换报表

转换工具通常在每次发出命令时产生一个转换报表，假脱机文件名与在参数 TOFILE 中指定的文件名一样，

如果你试图转换已经存在的成员，或者一个不支持的成员类型，那么在工作日志中打印一个信息，指出这些成员没有被转换，如果需要日志文件，也会更新来反映这些问题，然后报表中没有这些成员的信息。

转换报表包括下面的信息：

CVTRPGSRC 命令选项

源码部分，包括：

转换错误或警告

CALL 操作

/COPY 指令

信息总结

最后总结

转换错误信息向你提供如何改正的建议，另外，在未转换的源码中的 CALL 操作和/COPY 指令，被标识出来，帮助识别转换的各部分，一般情况下，一个应用程序里的所有 RPG 成份，在同一时刻被转换。

如果不想要转换报表，那指定 CVTRPT(*NO)。

附录 1.2.2.7 转换自动报表源成员

在一个 RPGⅢ或 OPM RPG/400 源程序中检测到一个自动报表源成员(RPT 或 RPG38)，转换工具调用 CRTRPTPGM 命令来扩充源成员并且转换它，(这是因为 ILE RPG/400 不支持自动报表)。

自动报表程序每次由转换工具调用时都产生一个假脱机文件，可以检查这个文件，看看是否出现自动报表扩充错误，因为这些错误，在转换报表找不到。

在自动报表假脱机文件检查到的一个特别的错误指出没有找到/COPY 成员，转换工具不知道这些文件是否已丢失，但没有这些文件，是不能够成功地转换你的源码。

注：如果转换的源成员类型不是 RPG 或 RPT38，并且成员是一个自动报表源成员，那么在转换之前给出正确的源成员类型(RPT 或 RPT38)，否则可能出现转换错误。

附录 1.2.2.8 插入规范样板

因为 RPGⅣ的规范表是新的，要把规范校板插入到转换后的源码中，在命令 CVTRPGSRC 中指定 INSRTPL(*YES)，缺省值为 INSRTPL(*NO)。

附录 1.2.2.9 从一个数据文件转换源码

转换工具能从一个数据文件转换源码，因为数据文件通常没有顺序号，所以转换后输出文件的最小记录长度为 80 个字符。(见图 149)，建立的数据文件的记录长度为 100 个字符。

注：如果数据文件有顺序号，应该在运行转换工具之前去掉它们。

附录 1.2.3 源码转换的例子

例子中给出简单的 RPGⅢ的源码。

```
     H                                                              TSTPGM
     FFILE1   IF  E                    DISK                        COMM1
```

```
     FQSYSPRT O   F     132     OF    LPRINTER
     LQSYSPRT  60FL 56OL
     E                    ARR1    3   3  1              COMM2
     E                    ARR2    3   3  1
     IFORMAT1
     I              OLDNAME                          NAME
     I* DATA STRUCTURE COMMENT
     IDS1         DS
     I                                          1   3 FIELD1
     I* NAMED CONSTANT COMMENT
     I              'XYZ'                  C         CONST1                COMM3
     I                                          4   6 ARR1
     C           ARR1,3    DSPLY
     C                     READ FORMAT1                 01
     C           NAME      DSPLY
     C                     SETON                     LR
     C                     EXCPTOUTPUT
     OQSYSPRT E   01           OUTPUT
     O                         ARR2,3    10
**
123
**
456
```

图 150　RPGIII的 TEST1 源码

要转换这个源码，输入：

```
CVTRPGSRC  FROMFILE(MYLIB/QRPGSRC)      FROMMBR(TEST1)
           TOFILE(MYLIB/QRPGLESRC)  INSRTPL(*YES)
```

转换后的源码如图 151。

```
1 .....H*unctions++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Comments+++++++
2      H DFTNAME(TSTPGM)
3 .....F*ilename++IPEASFRlen+LKlen+AIDevice+.Functions++++++++++++++++++++++++++++Comments+++++++
4      FFILE1     IF   E            DISK                                      COMM1
5      FQSYSPRT   O    F  132       PRINTER OFLIND(*INOF)
6      F                                 FORMLEN(60)
7      F                                 FORMOFL(56)
```

```
8 .....D*ame++++++++++ETDsFrom+++To/L+++IDc.Functions++++++++++++++++++++++++++++Comments+++++++
9       D ARR2            S               1    DIM(3) CTDATA PERRCD(3)
10      D* DATA STRUCTURE COMMENT
11      D DS1             DS
12      D  FIELD1                 1       3
13      D  ARR1                   4       6
14      D                                      DIM(3) CTDATA PERRCD(3)              COMM2
15      D* NAMED CONSTANT COMMENT
16      D CONST1          C                    CONST('XYZ')                         COMM3

17 .....I*ilename++SqNORiPos1+NCCPos2+NCCPos3+NCC.............................Comments++++++

18 .....I*.............Ext_field+Fmt+SPFrom+To+++DcField+++++++++L1M1FrP1MnZr......Comments++++++
19      IFORMAT1
20      I              OLDNAME                      NAME

21 .....C*0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....Comments++++++
22      C     ARR1(3)       DSPLY
23      C                   READ      FORMAT1                                  01
24      C     NAME          DSPLY
25      C                   SETON                                        LR
26      C                   EXCEPT    OUTPUT
27      OQSYSPRT   E            OUTPUT           01
28      O                       ARR2(3)             10
29 **CTDATA ARR1
30 123
31 **CTDATA ARR2
32 456
```

图 151　转换后的(RPGⅣ)TEST1 源码

关于转换后的源码注意以下几点：

- 新表的类型是 H(控制)，F(文件)，D(定义)，I(输入)，C(计算)和 O(输出)；它们必须按这个顺序输入。
- 转换的源码包含新类型的规范样板，由于在 CVTRPGSRC 上指定了 INSRTPL(*YES)。
- 控制文件和定义规范表是可定向的键字，见第 2，4-7 和 9-16 行。
- ILE 成员有一个新的规范类型，定义规范表用它来定义独立的字段，数组和常量命名以及数据结构。
- 在这个例子里：
  --ARR2 被定义为一个独立的数组(第 9 行)
  --数据结构 DS1 被定义为一个带有两个子字段 FIELD1 和 ARR1 的数据结构(第 11-14 行)。
  --CONST1 被定义为一个常量(第 16 行)。
- I 表现在仅用来定义文件的记录和字段，见第 19-20 行。
- E 表已不用，数组和表现在用定义规范表定义。
- 在扩展规范表上的记录地址文件(RAF)，已由在 F 表中的键字 RAFDATA 所代替。

• 行计数规范已不用，它们被 F 表中的键字 FORMLEN 和 FORMOFL 代替，见第 6-7　　行。
• 所有规范的类型都扩充成允许文件名和字段名用 10 个字符。
• 在 RPGⅣ中，数据结构(用定义规范表定义的)必须在 I 表之前。
• 在 RPGⅢ中，命名常量能够出现在数据结构的中间，这在 RPGⅣ中是不允许的。
• 在转换后的源码中，CONST1(16 行)已经移到数据结构 DS1(11 行)之后。
• 如果一个规范表被移动，在它之前的注释也被移动。
• 在转换后的源码中，CONST1 和 DS1 上面的注释被移到规范表之后。
• 在 RPGⅢ中，为了把数组定义为数据结构的子字段，要用相同的名字来既定义数组又定义数据结构子字段，这种双重定义在 RPGⅣ中是不允许的，代替的办法是使用新的键字语法定义子字段时，定义数组属性。
• 在这个例子中，ARR1 在 OPM 版本中被定义两次，但是在转换后的源码中被合并成为一个单独的定义，见 13，14 行。
• 合并 RPGⅢ数组定义可能导致数组定义的重排序，如果重排序数组是一个编译时数组，那么数组数据装入时可能受到影响，为了克服这个问题，RPGⅣ提供了一个键字格式为**记录，在**之后输入一个键字 FTRANS，ALTSEQ 或 CTDATA，如果键字是 CTDATA，则在 10-19 列输入数组名或者表名。
• 在这个例子中，由于 ARR2 的两个规范的合并，导致数组 ARR2 在数组 ARR1 之前，转换工具用转换的**记录，插入了键字数组名，这样来保证编译时数据的正确装入，见 29，31 行。
• 注意数组句法已经改变，数组 ARR1 标志在 RPGⅢ中是 3，在 RPGⅣ中是 ARR1(3)，见 28 行。

附录 1.2.4 分析转换信息

转换工具提供两条分析转换结果的方法，它们是：
• 转换错误报表
• 日志文件

附录 1.2.4.1 使用转换报表

如果在命令 CVTRPGSRC 中指定了 CVTRPT(*YES)参数转换工具产生一个转换报表，假脱机文件名与在参数 TOFILE 上指定的文件名相同。
转换报表包括四个部分：
1.CVTRPGSRC 命令选项；
2.源码部分；
3.信息总结；
4.最后总结。
报表清单的第一部分包括了 CVTRPGSRC 命令使用的选项总结，图 152 给出了为转换的命令总结的例子。

```
     5763RG1 V3R1M0  940909RN          IBM ILE RPG/400                          AS400S01
09/30/94 20:41:35        Page        1

        Command  . . . . . . . . . . . . :   CVTRPGSRC
          Issued by  . . . . . . . . . . :     DAVE

        From file  . . . . . . . . . . . :   QRPGSRC
```

```
  Library  . . . . . :     MYLIB
From member  . . . . . :   REPORT

To file. . . . . :   QRPGLESRC
  Library  . . . . . :     MYLIB
To member  . . . . . :   *FROMMBR

Log file . . . . . :   *NONE
  Library  . . . . . :
Log member . . . . . :   *FIRST

Expand copy members. . . . . :   *NO
Print conversion report  . . . . :   *YES
Include second level text. . . . :   *YES
Insert specification template. . :   *YES
```

图 152　转换报表的命令总结

源码部分包括有警告或者是错误信息的行，当用 SEU 显示时，这些行的第一列有一个星号(*)，信息总结包括所有三种类型的信息。要特别注意两种信息：

- RNM0508--flags/copy 语句；
- RNM0511--flags　Call 操作。

一个程序里的所有/COPY 成员必须按符合 ILE　RPG/400 程序的顺序去转换，这样编译时才不出错。同样在同一时间转换与 CALL 相关的所有成员，用报表的这部分帮助你识别这些成员，图 153 给出了转换的源码部分。

```
     5763RG1 V3R1M0  940909RN          IBM ILE RPG/400                                TORASD80
06/06/94 20:41:35        Page       2

        From file  . . . . . :   MYLIB/QRPGSRC(REPORT)
        To file. . . . . :   MYLIB/QRPGLESRC(REPORT)
        Log file . . . . . :   *NONE

                          C o n v e r s i o n    R e p o r t

     Sequence             <-----------------------            Source             Specifications
--------------------------><-------------- Comments --------------> Page

Number    ....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9...
.+...10....+...11....+...12 Line

        000002 C                   CALL      PROG1
      *RNM0511 00 CALL operation code found.
```

```
  000003 C/COPY COPYCODE
*RNM0508 00 /COPY compiler directive found.

  000004 C                   FREE      PROG2
*RNM0506 30 FREE operation code is not supported in RPG IV.

       * * * * *  E N D  O F  S O U R C E  * * * * *
```

图 153　转换报表的源码部分

报表清单中的信息总结部分显示发出的不同信息，如果指定了 SECLVL(*YES)，二级信息将在信息总结中。图 154 给出了转换的信息部分，包括二级信息。

```
     5763RG1 V3R1M0  940909RN          IBM ILE RPG/400                          AS400S01
09/30/94 20:41:35        Page       2

                          M e s s a g e   S u m m a r y

 Msg id  Sv Number Message text

*RNM0508 00      1 /COPY compiler directive found.
                   Cause . . . . . :   In order for this RPG IV source to
                     compile correctly, ensure that all /COPY source members
                     included in this source member have also been converted to
                     RPG IV.
                   Recovery  . . . :   Ensure that all /COPY source
                     members are converted prior to compiling in RPG IV. In some
                     cases, problems may result when attempting to convert and
                     compile source members that make use of the /COPY compiler
                     directive.  If this situation results, specify *YES for the
                     EXPCPY parameter on the CVTRPGSRC command to expand the
                     /COPY member(s) into the converted source.  For further
                     information see the ILE RPG/400 Programmers Guide.

*RNM0511 00      1 CALL operation code found.
                   Cause . . . . . :   RPG specifications that contain CALL
                     operation codes have been identified because the user may
                     wish to:
                       -- change the CALL operation code to CALLB to take
                     advantage of static binding
                       -- convert all programs in an application to RPG IV.
                   Recovery  . . . :   Convert the CALL
                     operation code to a CALLB if you wish to take advantage of
                     static binding or convert the called program to RPG IV if
```

```
                you wish to convert all programs in an application.

*RNM0506 30      1 FREE operation code is not supported in RPG IV.
                   Cause . . . . . :   The RPG III or RPG/400 program contains
                     the FREE operation code which is not supported in RPG IV.
                   Recovery  . . . :   Remove the FREE operation and replace
                     it with alternative code so that the programming logic is
                     not affected prior to compiling the converted source.

       * * * * *  E N D  O F  M E S S A G E  S U M M A R Y  * * * * *
```

图 154　转换报表的信息总结

报表的最后总结提供了信息和记录统计，最后的状态信息也放在作业日志中，图 155 给出了转换的信息部分。

```
                        F i n a l  S u m m a r y
  Message Totals:
    Information  (00) . . . . . :       2
    Warning      (10) . . . . . :       0
    Severe Error (30+)  . . . . :       1
    -------------------------------  -------
    Total . . . . . :       3

  Source Totals:
    Original Records Read . . . . . :       3
    Converted Records Written . . . . :       4
    Highest Severity Message Issued . :      30

        * * * * *  E N D  O F  F I N A L  S U M M A R Y  * * * * *

        * * * * *  E N D  O F  C O N V E R S I O N  * * * * *
```

图 155　转换报表的最后总结

附录 1.2.4.2 使用日志文件

通过浏览日志文件，能看到转换的结果，日志文件在每次转换操作之后被更新，它跟踪记录了：

- 源成员和它们的库名；
- 转换后的源文件名和它们的库名；
- 发现的最高级别的错误。

例如，如果没有发现错误，转换状态设为 0，如果发现严重错误，则该状态设为 30。

如果试图去转换一个不支持的成员类型或转换已转换过一个成员，那么因为严重错误(级别为 40 或更高)转

换将不会发生，一条转换状态设为 40 的记录加到日志文件中，因为没能做转换，把 TOFILE，TOMBR 和 TOLIBRARY 设为空，来表示没有生成一个 TOMBR。

日志文件是一个外部描述的物理数据库文件，在库 QRPGLE 的文件 QARNCVTLG 中提供了这个文件的一个“样本”，它有一个记录格式叫 QRNCVTLG，所有的字段名都是 6 个字符长度，形如 LGxxxx，这里 xxxx 指出字段，图 156 显示了这个文件的 DDS。

```
A          R QRNCVTFM
A            LGCENT         1A         COLHDG('CVT' 'CENT')
A                                      TEXT('Conversion Century: 0-20th 1-+
A                                      21st')
A            LGDATE         6A         COLHDG('CVT' 'DATE')
A                                      TEXT('Conversion Date : format is Y+
A                                      YMMDD')
A            LGTIME         6A         COLHDG('CVT' 'TIME')
A                                      TEXT('Conversion Time : format is H+
A                                      HMMSS')
A            LGSYST         8A         COLHDG('CVT' 'SYST')
A                                      TEXT('Name of the system running co+
A                                      nversion')
A            LGUSER        10A         COLHDG('CVT' 'USER')
A                                      TEXT('User Profile name of the user+
A                                      running conversion')
A            LGFRFL        10A         COLHDG('FROM' 'FILE')
A                                      TEXT('From File')
A            LGFRLB        10A         COLHDG('FROM' 'LIB')
A                                      TEXT('From Library')
A            LGFRMR        10A         COLHDG('FROM' 'MBR')
A                                      TEXT('From Member')
A            LGFRMT        10A         COLHDG('FMBR' 'TYPE')
A                                      TEXT('From Member Type')
A            LGTOFL        10A         COLHDG('TO' 'FILE')
A                                      TEXT('To File')
A            LGTOLB        10A         COLHDG('TO' 'LIB')
A                                      TEXT('To Library')
A            LGTOMR        10A         COLHDG('TO' 'MBR')
A                                      TEXT('To Member')
A            LGTOMT        10A         COLHDG('TMBR' 'TYPE')
A                                      TEXT('To Member Type')
A            LGLGFL        10A         COLHDG('LOG' 'FILE')
A                                      TEXT('Log File')
A            LGLGLB        10A         COLHDG('LOG' 'LIB')
A                                      TEXT('Log Library')
A            LGLGMR        10A         COLHDG('LOG' 'MBR')
```

```
A                                      TEXT('Log Member')
A            LGCEXP         1A         COLHDG('CPY' 'EXP')
A                                      TEXT('Copy Member Expanded: Y=Yes, +
A                                      N=No')
A            LGERRL         1A         COLHDG('CVT' 'RPT')
A                                      TEXT('Conversion Report Printed: Y=+
A                                      Yes, N=No')
A            LGSECL         1A         COLHDG('SEC' 'LVL')
A                                      TEXT('Second Level Text Printed: Y=+
A                                      Yes, N=No')
A            LGINSR         1A         COLHDG('INSR' 'TPL')
A                                      TEXT('Template Inserted: Y=Yes, N=N+
A                                      o')
A            LGSTAT         2A         COLHDG('CVT' 'STAT')
A                                      TEXT('Conversion Status')
A            LGMRDS        50A         COLHDG('MBR' 'DESC')
A                                      TEXT('Member Description')
```

图 156 库 QRPGLE 中日志文件 QARNCVTLG 的 DDS

附录 1.2.5 解决转换发生的问题

转换问题是由下列一种或多种原因造成：

- RPGⅢ源码有编译错误；
- RPGⅢ的某些语句不被 RPGⅣ支持；
- 在 RPGⅢ源码中有一个或多个/COPY 编译指令；
- 使用了外部描述的数据结构；
- OPM 和 ILE 在运行时多行为不同。

下面的部分讨论以上列出的内容。

附录 1.2.5.1 在于 RPGⅢ编码中有编译错误

转换工具假设转换的是一个有效的 RPGⅢ程序，即，设有编译错误的程序，如果不是这样，那么在转换期间可能发生难以预料的结果，如果你的程序有编译错误，先使用 RPGⅢ编译程序去编译它，并改正所有的错误，之后执行转换。

附录 1.2.5.2 不支持的 RPGⅢ特性

有几个 RPGⅢ语言特性不被 RPGⅣ支持，最应引起注意的是：

- 自动报表功能；
- FREE 操作码；
- DEBUG 操作码。

由于不支持自动报表功能，转换工具将自动地扩展类型是 RPT 或 38RTR 的程序(即调用自动报表)。

必须在转换前或转换后用等价逻辑来替换 FREE 或 DEBUG 操作码。

在命令 CVTRPGSRC 指定 CVTRPT(*YES)，将得到一个转换报表，该报表将指出问题的，关于转换自动报表成员的详细的信息，请见附录 1.2.2.7，关于 RPGⅢ和 RPGⅣ之间不同之处的详细资料，请见附录 1.1。

附录 1.2.5.3 使用/COPY 编译指令

在一些情况下，在实际编译转换过的 RPGⅣ源码之前，不能发现某些错误，这种类型的转换错误通常与使用/COPY 编译指令有关，它分为两类：合并问题和前后关系敏感问题，下面讨论问题发生的原因及解决的办法。

附录 1.2.5.3.1 合并问题

因为 RPGⅢ与 RPGⅣ语言之间的不同，转换工具对某些源语句，要重排序，例子见附录 1.2.3，TEST1 的 RPGⅢ源成员，比较一下附录 1.2.3 的图 150 和图 151 中的数据结构 DS1，看到数据结构 DS1 被移到记录格式 FORMAT1 之前。

现在假设 RPGⅢ的成员(TEST1 的)分为两个成员 TEST2 和 COPYDS1，其中数据结构 DS1 和命名常量 CONST1 是在一个拷贝成员 COPYDS1 中并且这个拷贝成员包括在成员 TEST2 中，图 157 和图 158 给出 TEST2 的源码和 COPYDS1 的源码。

```
     H                                                                  TSTPGM
     FFILE1   IF  E                    DISK                            COMM1
     FQSYSPRT O   F     132     OF     LPRINTER
     LQSYSPRT  60FL 560L
     E                    ARR1    3   3  1               COMM2
     E                    ARR2    3   3  1
     IFORMAT1
     I              OLDNAME                          NAME
     D/COPY COPYDS1
     C           ARR1,3    DSPLY
     C                     READ FORMAT1                  01
     C           NAME      DSPLY
     C                     SETON                     LR
     C                     EXCPTOUTPUT
     OQSYSPRT E   01           OUTPUT
     O                         ARR2,3    10
**
123
**
456
```

图 157　TEST2 的源码

```
I* DATA STRUCTURE COMMENT
IDS1          DS
I                                        1    3 FIELD1
I* NAMED CONSTANT COMMENT
I               'XYZ'                 C          CONST1                 COMM3
I                                        4    6 ARR1
```

图 158　COPYDS1 的源码

在这个情况下转换工具将对 TEST2 和 COPYDS1 进行正确的转换，因而，在编译时包括拷贝成员，由于这里有/COPY 指令，它将被插入到 FORMAT1 之下，其结果是，在拷贝成员 COPYDS1 的所有源码行中将得到一个“源码记录超序”的错误，这是因为在 RPGⅣ中，定义规范表必须在输入规范表之前。

注意：转换工具由于不知道/COPY 成员的内容，所以不能够把/COPY 指令移到 FORMAT1 之前。

有两种方法来改正这类问题：

1.(在 CVTRPGSRC 命令中)使用 EXPCPY(*YES)选项，在转换后的 RPGⅣ源成员中指出包括所有的/COPY 成员。

这个方法很简单并在大多数时间里能用，但，在每个源成员中包括/COPY 成员，要降低应用程序的维护能力。

2.在转换后使用 ILE　RPG/400 编译清单中的信息和《ILE　RPG/400 参考》人工改正后再编译 。

这类问题的其他例子包括：

・行计数规范和记录地址文件：

在 RPGⅢ中，在 E 表中定义的行计数规范和记录地址文件改成在 F 表中的键字(REAFDATA，FORMLEN 和 FORMOFL)，如果/COPY 成员的内容仅包含在 E 的行规范和/或者记录地址，而没有相应的 F 表，转换工具就不知道在哪插入键字。

・E 表数组和数据结构子字段

在附 1.2.3“源码转换的例子”中，在 RPGⅣ中，不能用同一名字定义独立的数组和数据结构子字段，因此象在 TEST1 例子中给出的那样(附录 1.2.3 中图 151)，转换工具必须合并两个定义，然而，数组和数据结构子字段不在同一个源成员中(即，一个或都在/COPY 成员中)，不能发生合并并且产生编译时错误。

・合并编译时数组和编译时数据(**)记录

如例子 TEST1 所示(附录 1.2.3 中图 151)，如果编译时数组同数据结构子字段合并，将影响数组数据装入。为了避免这个问题在至少合并一个编译时数组时，把编译时数组数据改为新的**CTDATA 格式，然而，如果数组和数据不在于同一个源文件(即一个或两个在一个/COPY 成员中)使用**CTDATA 格式命名的，编译时数据记录不能正确的处理。

附录 1.2.5.3.2 前后关系敏感问题

在 RPGⅢ，有这种情况：不用主源成员的规范表中上下文的内容就不可能确定/COPY 成员中规范表的类型，有两个这种问题的例子：

・数据结构子字段或者程序描述文件字段

```
I* 如果 RPGⅣ源码只有下面二个字段描述的源语句，转换工具就不知道
```

```
I* 怎样转换它们。这些语句可能是数据结构字段（它用定义规范转换）。
I* 也可以是程序描述文件的字段。（它用输入规范转换）。
I                                          1    3 FIELD1
I                                          4    6 FIELD2
```

图 159　RPGⅢ源码

重命名一个外部描述数据结构字段或者外部描述文件的字段。

```
I* 如果 RPGⅢ源码只包括下面的 CHAR 语句，转换工具就不知怎样转换它，
I* 这个语句可能是对外部描述数据结构字段的重命名（用定义规范转换），
I* 也可能是外部描述文件字段的重命名（要用输入规范转换）。
I              CHARACTER                   CHAR
```

图 160　RPGⅢ源码

在上面两个例子中，假定一个数据结构并且产生一个定义规范表，也产生了一个包含 I 表编码的注释块，例如，转换工具将把图 159 中的源码转换为图 161 的编码显示，如果要求 I 表的编码要删除定义规范表，并且从相应的 I 表中把星号变空白。

```
D* 如果 RPGⅢ源码只有下面二个字段描述的源语句，转换工具就不知道
D* 怎样转换它们。这些语句可能是数据结构字段（它用定义规范转换）。
D* 也可以是程序描述文件的字段。（它用输入规范转换）。
D FIELD1                  1       3
D FIELD2                  4       6
I*                                  1    3  FIELD1
I*                                  4    6  FIELD2
```

图 161　RPGⅣ源码

记住，有两个选择来改正这类错误，一是用命令 CVTRPGSRC 的选项 EXPCPY(*YES)，一是用人工方式在转换后改正编码。

附录 1.2.5.4 使用外部描述的数据结构

即使在 CVTRPGSRC 命令有两个问题，仍要手动处理它们。指定了 EXPCPY(*YES)，与使用外部描述的数据结构有关。

由于这些问题将产生编译时错误，所以可使用 ILE　RPG/400 编译清单的信息和《ILE　RPG/400 参考》来改正它们。

附录 1.2.5.4.1 合并用外部描述的 DS 子字段的数组

前面讲过，在 RPGⅣ中不允许用相同的名字去定义一个独立的数组和数据结构子字段，一般地讲，转换工

具将合并这两个定义，但，如果子字段是在外部描述的数据结构中，不处理这种合并你要手动改正转换后的源成员。例如，在图 162 中的字段 ARRAY，在图 163 中被包括两次，一次是作为一个独立的数组，另一次是在外部描述的数据结构 EXTREC 中，转换后的 RPGⅣ源码如图 164 中所示，由于 ARRAY 被定义两次，这个编码不能编译，为了改正这个问题，删除独立的数组并且把一个带有键字的子字段加到数据结构 DSONE 中，如图 165 所示。

```
A            R RECORD
A              CHARACTER      10
A              ARRAY          10
```

图 162　DDS

```
E                       ARRAY       10   1
IDSONE      E DSEXTREC
C             CHAR        DSPLY
C                         SETON                        LR
```

图 163　RPGⅢ源码

```
D ARRAY             S                 1    DIM(10)
D DSONE           E DS                     EXTNAME(EXTREC)
C      CHAR             DSPLY
C                       SETON                                  LR
```

图 164　RPGⅣ源码

```
D DSONE           E DS                     EXTNAME(EXTREC)
D  ARRAY          E                        DIM(10)
C      CHAR             DSPLY
C                       SETON                                  LR
```

图 165　改正的 RPGⅣ源码

附录 1.2.5.4.2 一个外部描述的 DS 子字段的重命名和初始化

在 RPGⅢ，当重命名和初始化一个在外部描述的数据结构中的字段时，你要用两行源码，如图 166 所示的 CHAR 字段转换后的源码也有两行，如图 167 所示，因字段 CHAR 被定义两次时，将导致一个编译时错误，要改正编码必须合并字段 CHAR 的键字成为一个单独的行，如图 168 所示，这里键字字段 INZ 和 EXTFLD 已被结合，字段 CHAR 也仅有一种情况。

```
IDSONE      E DSEXTREC
I               CHARACTER                       CHAR
I I             'XYZ'                           CHAR
C            CHAR       DSPLY
C                       SETON                    LR
```

图 166　RPGⅢ源码

```
D DSONE          E DS                   EXTNAME(EXTREC)
D  CHAR          E                      EXTFLD(CHARACTER)
D  CHAR          E                      INZ('XYZ')
C      CHAR           DSPLY
C                     SETON                                  LR
```

图 167　RPGⅣ源码

```
D DSONE          E DS                   EXTNAME(EXTREC)
D  CHAR          E                      EXTFLD(CHARACTER)  INZ('XYZ')
C      CHAR           DSPLY
C                     SETON                                  LR
```

图 168　改正的 RPGⅣ源码

附录 1.2.5.5 运行时的区别

如果有运行前数组，该数组用数据结构复盖，在运行时，装入这些数组的顺序在 RPGⅢ和 RPGⅣ中可能不同，这导致覆盖部分的数据不同，数组装入的顺序是它们在源码中的顺序，在转换期间因为数组子字段的合并，顺序可能要改变。

一般地，你应该避免这种局面，当应用程序由 OPM 和 ILE 程序组成，且跨越 OPM 缺省的活动组和一个命名的活动组，当跨越这两个活动组时，要混合 ILE 行为和 OPM 行为所以结果很难预料，详细情况请参考 1.3 或《ILE 概念》。

附录 1.4　附录 D　编译清单

编译清单提供程序编码中要改正的有关 RPGⅣ语言的语法和语义方面的信息。这个清单帮助你用源码编辑来改错，当调试一个模块时，也可用它做为助手，这一部分告诉你如何理解 ILE　RPG/400 编译清单，怎么使用这个清单，请看 2.2.2。

要得到编译清单，在命令 CRTRPGMOD 或 CRTBNDRPG 中规定 OUTPUT(*PRINT)(这是缺省设置)。如果规定 OUTPUT(*NONE)则不产生编译清单。

表 21 总结了这些规则及有关的编译清单信息。

表 21　编译清单各部分

清单部分	选项(1)	解　释
序		命令选项概括
源码列表		源码规定
联机诊断信息		源码中一行包括的错误信息
/COPY 成员	*SHOWCPY	/COPY 成员源记录
外部描述文件	*EXPDDS	生成规范表
匹配字段		与匹配字段区配的长度
附加的诊断信息		源码中多于一行的错误
输出缓冲区的字段位置		程序描述的输出字段的起始、结束位置
/COPY 成员表		/COPY 成员列表和它们的外部名
编译时数据		编译源记录
交替分配顺序		ALTSEQ 记录和表或 NLSS 信息和表
文件传输		文件传输记录
数组		数组记录
表		表记录
键字字段信息	*EXPDDS	键字段属性
交叉引用	*XREF	文件和记录、字段和指示器引用
外部引用	*EXT	在编译期间引用的外部过程和字段清单
信息总结		信息列表及发生的次数
二级文本	*SECLVL	信息的二级文本
最后总结		信息和源记录总计及最后编译信息
码生成错误(2)		在编码生成部分的错误
连接部分(2)		在 CRTBNDRPG 命令连接期间发生的错误

(1)OPTION 列指出，在 OPTION 参数上指定这个值才能得到相应信息，空列表示如果规定 OUTPUT(*PRINT)总是出现这个信息。

(2)仅当有这方面错误时才有这部分信息，没有选项能禁止出现这部分内容。

附录 1.4.1 读编译清单

下面的内容解释编译清单的各部分，按它们在清单中出现的顺序介绍。

附录 1.4.1.1 序

这部分总结了由 CL 命令分析处理的命令参数和它们的值。如果规定*LIBL 和*CURLIB，列出实际用的库，在序中也指出复盖的效果。图 170 解释了如何看程序 MYSRC 清单的序言部分，程序是用 CRTBNDRPG 编译的。

```
      5763RG1 V3R1M0  940909RN          IBM ILE RPG/400          MYLIB/MYSRC    1          AS400S01
09/30/94 09:43:20       Page     1

         Command  . . . . . . . . . . . . :   CRTBNDRPG
```

```
  Issued by  . . . . . :     MYUSERID

Program  . . . . . :   MYSRC     2
  Library  . . . . . :     MYLIB
Text 'description' . . . . . :   *SRCMBRTXT

Source Member  . . . . . :   MYSRC     3
Source File  . . . . . :   QRPGLESRC   4
  Library  . . . . . :     MYLIB
  CCSID  . . . . . :     37
Text 'description' . . . . . :
Last Change  . . . . . :   09/30/94  11:54:44

Generation severity level  . . . :   10
Default activation group . . . . :   *YES
Compiler options . . . . . :   *XREF      *GEN       *NOSECLVL  *SHOWCPY      5
                               *EXPDDS    *EXT       *NOEVENT
Debugging views  . . . . . :   *NONE
Output . . . . . :   *PRINT
Optimization level . . . . . :   *NONE
Source listing indentation . . . :   '|'   6
Type conversion options  . . . . :   *NONE
Sort sequence  . . . . . :   *HEX
Language identifier  . . . . . :   *JOBRUN
Replace program  . . . . . :   *NO
User profile . . . . . :   *USER
Authority  . . . . . :   *LIBCRTAUT
Truncate numeric . . . . . :   *YES
Fix numeric  . . . . . :   *NONE
Target release . . . . . :   V3R1M0
Allow null values  . . . . . :   *NO
```

图 170　CRTBNDRPG 的序部分

1.页标题：页标题信息包括产品信息行及由 TITLE 指令提供的注释，详细信息请看　　2.2.2.1.1。
2.模块或程序：生成的模块目标名(如果用 CRTRPGMOD)或生成的程序目标名(如　　果用 CRTBNDRPG)。
3.源成员：源记录所在的源成员名(如果用复盖命令，这个名可能与 2 给出的名不同)。
4.源文件：实际上用来提供源记录的文件名，如果文件是被复盖的，使用复盖的源文件名。
5.编译选项：编译时起作用的编译选项，它们是在命令 CRTRPGMOD 或 CRTBNDRPG

中规定的。
6.标识符：这个字符用来标识清单中源码部分的结构操作。

附录 1.4.1.2 源码部分

这部分给出组成 ILE RPG/400 记录的源码规范定义，它总是给出最根本的源成员记录，如果也规定了OPTION(*EXPDDS)，那么源码部分也给出从外部描述文件生成的记录，用行号边上的‘=’指出它们。如果规定*NOEXPDDS，就不显示这些记录。如果规定 OPTION(*SHOWCPY)，也显示来自源码中规定的/COPY 成员的记录，并用行号边上的‘+’号表示，如果规定*NOSHOWCPY 就不显示它们。

这部分也给出源码中的语法错误，相应的也包括匹配字段表。

注意：清单中的行号反映的是编译后源码的行号，例如：程序 MYSRC 在 26 行上有/COPY 语句。在没编译的源文件中，下一行是 DOWEQ，但在清单中，这个操作是在 30 行，中间三行是从/COPY 成员中得到的。

图 171 给出 MYSRC 的源码部分。

```
      5763RG1 V3R1M0  940909RN          IBM ILE RPG/400          MYLIB/MYSRC                AS400S01
09/30/94 10:03:55        Page       2
       1
      Line     <---------------------  Source  Specifications  -------------------------->
Comments ----> Do  Page  Change  Src  Seq

Number ....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+
...10 Num Line  Date    Id   Number
                            S o u r c e   L i s t i n g
          1                                  H                                    ALTSEQ(*SRC)
940909       000100

          2   FInFile             IF         E                                          DISK
940909       000200

*-------------------------------------------------------------------------------------------* 2
*                             RPG name         External name                              *
* File name. . . . . . . . . :  INFILE          MYLIB/INFILE                               *
* Record format(s) . . . . . :  INREC           INREC                                      *
*-------------------------------------------------------------------------------------------*
          3   FKEYL6                IF          E                                  K    DISK
940909       000300

*-------------------------------------------------------------------------------------------*
*                             RPG name         External name                              *
* File name. . . . . . . . . :  KEYL6           MYLIB/KEYL6                                *
* Record format(s) . . . . . :  REC1            REC1                                       *
*                             REC2            REC2                                       *

*-------------------------------------------------------------------------------------------*
          4   FOutFile          O             E                                          DISK
```

```
940909       000400

*-------------------------------------------------------------------------------------------------*
*                               RPG name         External name                                    *
* File name. . . . . . . . . :  OUTFILE          MYLIB/OUTFILE                                    *
* Record format(s) . . . . . :  OUTREC           OUTREC                                           *
*-------------------------------------------------------------------------------------------------*

          5  D  Blue                    S                       4       DIM(5)  CTDATA  PERRCD(1)
940908       000500
          6  D  Green                     S                          2        DIM(5)  ALT(Blue)
940908       000600
          7  D  Red                     S                       4       DIM(2)  CTDATA  PERRCD(1)
940908       000700
          8   D   DSEXT1                   E   DS                        100          PREFIX(BI_)
940909       000800

*-------------------------------------------------------------------------------------------------*  3
1
* Data structure . . . . . :  DSEXT1                                                              *
                          1
* Prefix . . . . . . . . . :  BI_                                                                 *
                          1
* External format  . . . . . :  REC1 : MYLIB/DSEXT1                                               *
                          1
* Format text  . . . . . . . :  BUFFER INFO                                                       *
                          1

*-------------------------------------------------------------------------------------------------*
1

4
          9=D BI_FLD1                          5A    EXTFLD (FLD1)                          FORMAT
1      1
         10=D BI_FLD2                         10A    EXTFLD (FLD2)                         MAPPING
1      2
         11=D BI_FLD3                         18A    EXTFLD (FLD3)                          TABLE
ENTRY                              1      3

         12=IINREC
2      1

*-------------------------------------------------------------------------------------------------*
```

```
2
* RPG record format  . . . . :  INREC                                                *
2
* External format  . . . . . :  INREC : MYLIB/INFILE                                  *
2

*-------------------------------------------------------------------------------------*
2
          13=I                                                    1      25    FLDA
2      2
          14=I                                                   26      90    FLDB
2      3
          15=IREC1
3      1

*-------------------------------------------------------------------------------------*
3
* RPG record format  . . . . :  REC1                                                 *
3
* External format  . . . . . :  REC1 : MYLIB/KEYL6                                   *
3

*-------------------------------------------------------------------------------------*
3
          16=I                                          *ISO-D    1      10    FLD12
3      2
          17=I                                                   11      13    FLD13
3      3
          18=I                                                   14      17    FLD14
3      4
          19=I                                                   18      22    FLD15
3      5
          20=IREC2
4      1

*-------------------------------------------------------------------------------------*
4
* RPG record format  . . . . :  REC2                                                 *
4
* External format  . . . . . :  REC2 : MYLIB/KEYL6                                   *
4

*-------------------------------------------------------------------------------------*
```

```
4
         21=I                                              *ISO-D       1       10     FLD22
4      2
         22=I                                                           11      13     FLD23
4      3
         23=I                                                           14      17     FLD24
4      4
         24=I                                                           18      22     FLD25
4      5

      Line                    <----------------------        Source        Specifications
----------------------------------------------><---- Comments ----> Src  Seq
      Number           ....1....+....2....+<--------           26            -            35
--------->....4....+....5....+....6....+....7....+....8....+....9....+...10 Id   Number
         25 C                       MOVE                          '123'            BI_FLD1
000900
         26                                                  /COPY                     MYCPY
940909      001000

*---------------------------------------------------------------------------------------------* 5
* RPG member name  . . . . . :  MYCPY                                                       *
5
* External name  . . . . . . :  MYLIB/QRPGLESRC(MYCPY)                                      *
5
* Last change  . . . . . . . :  09/30/94  11:55:20                                          *
5
* Text 'description' . . . . :                                                              *
5

*---------------------------------------------------------------------------------------------*
        6
         27+C                            Blue(1)                                       DSPLY
5 000100
         28+C                              Green(4)                                    DSPLY
5 000200
         29+C                          Red(2)                                          DSPLY
5 000300
        7
         30  C         *in20                doweq                                       *OFF
001100
         31  C                                |  READ                                  InRec
----20                            001200
         32  C                              |  if                                 NOT  *in20
```

```
001300
          33    C                     FLDA                                        |     |     DSPLY
001400
          34    C                                                                 |     |     endif
001500
          35    C                                                                       enddo
001600
          36  C                                 write                                 outrec
001700

8
          37    C                                                                       SETON
LR----                                 001800

      Line     <---------------------- Source Specifications --------------------------><----
Comments ----> Do  Page  Change  Src  Seq

Number ....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+
...10 Num Line  Date    Id   Number
          38=OOUTREC
6       1

*-------------------------------------------------------------------------------------------------*
6
             * RPG 记录格式：QUTREC                                    *
6
             * 外部格式           QUTREC：MYLIB/QUTFILE                      *
6

*-------------------------------------------------------------------------------------------------*
6
          39=0                                FLDY                    100   CHAR          00
6       2
          40=0                                FLDZ                    132   CHAR          32
6       3
             * * * * *   E N D   O F   S O U R C E   * * * * *
```

图 171 清单的源码部分

1.源码标题
行　号：从 1 开始，每行加 1，当用语句号调试时使用它。
标尺行：当规定标尺时，校准这行。

加　号：指出结构操作的级别。
页　行：给出源记录的头 5 行。
源码 ID：指出记录的源码(/COPY 或 DDS)。对于/COPY 成员也能用它从/COPY 成员表中得到外部成员名。
顺序号：给出源成员 SEU 的记录顺序号，也给出物理文件成员记录或从 DDS 生成记录的行增量数。

2.文件/记录信息
给出外部描述文件及它所包括的记录。
3.DDS 信息：指出从哪个外部描述文件得到的字段信息，如果有前缀值，也要给出。如果在 DDS 中规定了记录的注释，也给出来。
4.生成规范：给出从 DDS 生成的规范，由行号边上的‘=’指出。如果在 DDS 中规定了字段的注释，至多可显示 50 个字符。
5./COPY 成员信息：指出用哪一个/COPY 成员。如果有注释，也显示，给出对成员最后一次修改的日期和时间。
6./COPY 成员记录：给出从/COPY 成员中得到的记录，由行号边上的‘+’指出。
7.标识：当需要标识结构操作时，给出它们出现的方式。
8.指示器的使用：当使用指示器时，给出没用指示器的位置。

附录 1.4.1.3 附加的诊断信息

附加的诊断信息部分列出了编译信息，它指出一些错误，如果可能，信息指出错误所在源码的行号，图 172 给出了一个例子。

```
                    A d d i t i o n a l   D i a g n o s t i c   M e s s a g e s

 Msg id  Sv Number Message text
*RNF7086 00      2 RPG handles blocking for file INFILE. INFDS is updated only
                   when blocks of data are transferred.
*RNF7086 00      4 RPG handles blocking for file OUTFILE. INFDS is updated
                   only when blocks of data are transferred.
*RNF7066 00      0 Record-Format REC1 not used for input or output.
*RNF7066 00      0 Record-Format REC2 not used for input or output.

 * * * * *  E N D  O F  A D D I T I O N A L  D I A G N O S T I C  M E S S A G E S  * *
* * *
```

图 172　附加诊断信息的例子

附录 1.4.1.4 输出缓冲区字段的位置

当源码中含有程序描述的输出规范表时，字段在输出缓冲表中的位置包括在清单中，对每个输出的变量或文字，表包含输出字段规范表的行号以及它在输出缓冲区中起始和结束的位置，如果文字太长，在表中被截断，后加‘…’，(例如，‘Extremely-long-litera…’)。

附录 1.4.1.5 /COPY 成员表

/COPY 成员表标识了在源码中指定的/COPY 成员并列出它们的外部名称，可用源码的 ID 号找到成员的名字和位置，这张表作为被模块/程序使用成员的记录也是有用的，图 173 给出了一个例子。

```
                        / C o p y   M e m b e r s

  Line   Src  RPG name    <-------- External name -------> CCSID  <- Last change ->
  Number Id               Library    File       Member            Date     Time
      26    5 MYCPY       MYLIB      QRPGLESRC  MYCPY          37  09/30/94 11:55:20

           * * * * *   E N D   O F   / C O P Y   M E M B E R S   * * * * *
```

图 173 一个简单的/COPY 成员表

附录 1.4.1.6 编译时数据

编译时数据部分包括 ALTSEQ 或 NLSS 表的有关信息，以及表和数组的有关信息，在这个例子中，可以看到一个交替分配顺序和两个数组，如图 174 所示。

```
                          C o m p i l e   T i m e   D a t a

        27                                                                          **
940609         001800
                 *-----------------------------------------------------------------*
                 * Alternate Collating Sequence Table Data:                        *
                 *-----------------------------------------------------------------*
        31              ALTSEQ                                   1122ACAB4B7C36F83A657D73
940609         001900

                 *-----------------------------------------------------------------*
                 * Alternate Collating Sequence Table:                             *
                 * Number of characters with an altered sequence . . . . . . :  6  1 *
                 *     0_ 1_ 2_ 3_ 4_ 5_ 6_ 7_ 8_ 9_ A_ B_ C_ D_ E_ F_        2      *
                 * _0  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .   _0          *
```

```
            * _1  .  22  3 .  .  .  .  .  .  .  .  .  .  .  .  .   _1          *
            * _2  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .   _2          *
            * _3  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .   _3          *
            * _4  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .   _4          *
            * _5  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .   _5          *
            * _6  .  .  .  F8 .  .  .  .  .  .  .  .  .  .  .  .   _6          *
            * _7  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .   _7          *
            * _8  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .   _8          *
            * _9  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .   _9          *
            * _A  .  .  .  65 .  .  .  .  .  .  .  .  .  .  .  .   _A          *
            * _B  .  .  .  .  7C .  .  .  .  .  .  .  .  .  .  .   _B          *
            * _C  .  .  .  .  .  .  .  .  .  .  AB .  .  .  .  .   _C          *
            * _D  .  .  .  .  .  .  .  73 .  .  .  .  .  .  .  .   _D          *
            * _E  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .   _E          *
            * _F  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .   _F          *
            *     0_ 1_ 2_ 3_ 4_ 5_ 6_ 7_ 8_ 9_ A_ B_ C_ D_ E_ F_              *
            *------------------------------------------------------------------*
         32                                                                        **
000000        002000
            *------------------------------------------------------------------*
            * Array . . . : BLUE        Alternating Array  . . . . : GREEN     *
            *------------------------------------------------------------------*
         33                                                                   1234ZZ
000000        002100
         34                                                                   ABCDYY
000000        002200
         35                                                                   5432XX
000000        002300
         36                                                                   EDCBWW
000000        002400
      *RNF8042 00     37 Not enough source records provided to load array or table.
         37                                                                        **
940608        002500
            *------------------------------------------------------------------*
            * Array . . . : RED                                                *
            *------------------------------------------------------------------*
         38                                                                     3861
940608        002600
         39                                                                     TKJL
940608        002700
         * * * * *  E N D   O F   C O M P I L E   T I M E   D A T A   * * * * *
```

图 174　编译时数据的例子

1.改变字符的总数
表明分类顺序被改变了的字符数。
2.被改变的字符
表中的行和列一起指出要被改变的字符。
3.交替顺序
选择的字符新的十六进制分类值。
4.数组/表的信息
指出编译程序要求数据的表或数组的名字，如果定义了交替数组，则该数组的名字也将显示出来。

附录 1.4.1.7 键字字段信息

这部分主要包括对每个索引文件中有关键字字段的信息，还包括多记录公共键字的信息，图 175 显示了一个例子。

```
                    K e y   F i e l d   I n f o r m a t i o n

      File            Internal    External
        Record        field name  field name  Attributes
    2   KEYL6
        Common Keys:
                                                DATE *ISO- 10
                                                CHAR        3
        REC1
                      FLD12                     DATE *ISO- 10
                      FLD13                     CHAR        3
                      FLD15                     CHAR        5
        REC2
                      FLD22                     DATE *ISO- 10
                      FLD23                     CHAR        3
                      FLD24                     CHAR        4

  * * * * *  E N D  O F  K E Y  F I E L D  I N F O R M A T I O N  * * * * *
```

图 175　键字字段的信息

附录 1.4.1.8 交叉引用表

交叉引用表包含了三个列表：
- 文件和记录
- 字段
- 指示器

用它检查在模块/程序中何处使用了文件，字段和指示器。

注意信息 RNF7031，当一个标识符没有被引用时就给出此信息，它只出现在清单的交叉引用部分和信息总结中，图 176 给出了一个例子。

```
                          C r o s s   R e f e r e n c e

        File and Record References:

           File              Device            References (D=Defined)
             Record
           INFILE            DISK                 2D
             INREC                             12       31
           KEYL6             DISK                 3D
             REC1                              15
             REC2                              20
           OUTFILE           DISK                 4D
             OUTREC                            36       38


        Field References:

           Field             Attributes        References (D=Defined M=Modified)
           *IN20             A(1)                 30       32
           BI_FLD1           A(5)                 25M
           BLUE       (5)    A(4)                  5D       6       27
*RNF7031   DSEXT1            S(100)                8D
           FLDA              A(25)                13D      33
*RNF7031   FLDB              A(65)                14D
           FLDY              A(100)               39
           FLDZ              A(32)                40
*RNF7031   FLD12             D(10*ISO-)           16D
*RNF7031   FLD13             A(3)                 17D
*RNF7031   FLD14             A(4)                 18D
*RNF7031   FLD15             A(5)                 19D
*RNF7031   FLD22             D(10*ISO-)           21D
*RNF7031   FLD23             A(3)                 22D
*RNF7031   FLD24             A(4)                 23D
*RNF7031   FLD25             A(5)                 24D
           GREEN      (5)    A(2)                  6D      28
           RED        (2)    A(4)                  7D      29


        Indicator References:
```

```
Indicator                              References (D=Defined M=Modified)
20                                         31D
LR                                         37D


*****  E N D  O F  C R O S S  R E F E R E N C E  *****
```

图 176　交叉引用表样本

附录 1.4.1.9 外部引用清单

外部引用部分列出了在连接时需要的外部过程和字段，这部分还给出了什么时候源码包含静态的连接过程，输入或输出字段。

静态连接过程包括过程名，CALLB 操作引用的名字及%PADDR 内部函数名。

输入和输出字段部分包括字段名，如果是一个数组的话，给出尺寸，字段属性和它的定义引用。

附录 1.4.1.10 信息总结

信息总结包含了发生错误的严重级别的总结，如果指定了 OPTION(*SECLBL)，它也提供二级信息文本，图 177 给出了一个例子：

```
                    M e s s a g e  S u m m a r y

 Msg id  Sv Number Message text
*RNF7031 00     12 The name or indicator is not referenced.
*RNF7066 00      2 Record-Format name of Externally-Described file is not
                   used.
*RNF8042 00      1 Not enough source records provided to load array or table.

      *****  E N D  O F  M E S S A G E  S U M M A R Y  *****
```

图 177　信息摘要样本

附录 1.4.1.11 最后总结

这部分提供了最终信息统计和源码统计，它还指出的编译状态，图 178 给出一个例子。

```
                    F i n a l  S u m m a r y

  Message Totals:
```

```
  Information  (00) . . . . . :       17
  Warning      (10) . . . . . :        0
  Error        (20) . . . . . :        0
  Severe Error (30+)  . . . . . :      0
  -------------------------------  -------
  Total . . . . . . . . . . . . . :    17


Source Totals:

  Records . . . . . . . . . . . . :    50
  Specifications  . . . . . . . . :    39
  Data records  . . . . . . . . . :     7
  Comments  . . . . . . . . . . . :     0


        * * * * *  E N D  O F  F I N A L  S U M M A R Y  * * * * *

Program MYSRC placed in library MYLIB. 00 highest severity. Created on 94/06/09 at 17:44:55.


          * * * * *  E N D  O F  C O M P I L A T I O N * * * * *
```

图 178　最后总结

附录 1.4.1.12 编码生成和连接错误

在最后总结之后，可以找到编码生成的错误和/或连接错误部分。

当编译程序生成模块目标码时，如果出现错误，才出现编码生成错误部分，通常不出现，这部分在 CRTBNDRPG 命令的连接期间，只要出现信息，就产生连接错误部分，在执行 CRTBNDRPG 命令期间，常见的是在源码中引用外部过程和字段失败。