目　　录

第一章　介绍

这章介绍 OS/400 的几个主要概念，这些概念在以后的章节中会陆续介绍。

系统操作是由以下几个方面控制的：

CL 命令：它用在交互和批作业及 CL 程序或过程中。

菜单选项：由选择一个菜单的选项来控制系统操作，交互用户也能用菜单来完成许多系统任务。

系统信息：用在程序和过程之间的通讯以及程序、过程和用户之间的通讯，它也能报告状态信息及错误条件。

1.1　控制语言

控制语言（CL）是与操作系统的主要接口，不同工作站用户可同时使用它们。单个的控制语言语句叫做一个命令,能用下列方式进入命令：

从工作站分别进入

批作业的一部分

做为源语句生成 CL 程序或过程

命令也能从命令行或命令入口显示中分别进入。为了简化使用 CL，所有的命令都用一个一致的语法规则。另外，对所有的命令，操作系统都支持一个提示，对大多数命令参数给出缺省值，也有有效性检查，以便保证在实现命令功能前保证命令的正确性。这样，CL 对不同的系统用户使用不同的系统功能提供一个简单的灵活的接口。

过程：是一个独立的高级语言语句。它完成某一特别功能或任务，然后返回到调用者。在 CL 中，过程通常由 PGM 语句开始，用 ENDPGM 语句结束。

模块：它是编译源语句产生的结果。模块要连编到程序中才能运行。一个 CL 模块由用户写的过程和由 CL 编译生成的程序入口例程组成。在其它高级语言中（例如

C），一个模块可以包括多个过程。

程序：一个 ILE 程序是由一个或多个模块组合在一起的目标。模块连编到程序中才能运行。一个程序必须要有一个程序入口例程，在生成每个模块时，CL 编译器生成一个程序入口例程。一个 OPM　CL 程序是用 CRTCLPGM 命令编译源码产生的目标。

服务程序：它是由一个或多个模块组成的目标，不能直接调用服务程序，可以用程序中的过程或服务程序调用另外服务程序中的过程。

命令语法：一个命令由命令名和参数组成。命令名通常由一个动词（动作）后跟一个名词或词组，它给出动作的接收者。做为命令名的字一般缩写或三个字母，例如，CL 的发送信息命令，命令名为 SNDMSG，用户用它往信息队列发送信息。CL 命令中的参数如是键字参数，键字通常也用缩写形式。在进入命令时，某些参数可按现定顺序来省略键字不写，这叫做位置参数格式。

1.2　CL 过程

CL 程序和过程是由 CL 命令组成的。命令编译 OPM 程序或模块，模块可与用 CL 或其它语言写的模块连编到一起构成程序。使用 CL 程序或过程的优点是：

它比单个输入命令和运行命令要快。

它对相同的命令及逻辑过程提供一致的处理。

某些功能不允许进入单个的 CL 命令，必须是程序或过程的一部分。能象其它 HLL 程序和过程一样测试和调试，能把参数传递给它，这样能根据使用要求分别完成操作。

能把 CL 模块与其它 ILE 高级语言过程或模块连编成程序。

CL 程序和过程能用于多种应用开发中，下面是些例子：

对交互应用的用户提供一个接口，通过它用户请求应用程序功能而不必了解所用的命令,这就使工作站用户作业变得很容易操作且减少由于进入命令产生错误的机会。

在应用程序中使用变量（例如：日期、时间或外部指针）来控制操作。规定应用程序使用的库列表，这就保证了无论何时运行应用程序都能完成预定操作。

为系统操作员提供预定的例程，例如启动子系统，提供文件备份。它能减少操作员频繁使用命令的数量，保证系统操作的一致性。

由系统提供的大多数命令都能用在 CL 程序或过程中，有些命令只能用在程序或过程中不能单个使用，这些命令包括：

逻辑控制命令：根据已存在的条件控制操作。

数据操作：提供与工作站用户通讯的方式。

程序或过程给工作站用户发送信息的命令。

接收信息的命令，它在程序和过程之间提供正常的通讯，指出错误及例外条件。

使用变量或参数在程序或过程的命令之间或在程序和过程之间传递信息。

调用其它过程。（仅能从命令行或批作业流中调用程序）

使用 CL 程序或过程，应用程序对某个功能可设计各自的程序或过程或 CL 程序或控制在应用程序中运行哪个程序或过程，应用程序由 CL 程序或其它 HLL 程序或过程组成，在这类应用程序，CL 程序或过程用于：

确定运行应用程序中的哪个程序或过程。

提供其它 HLL 语言不能完成的功能。

提供与应用程序用户的接口。

CL 程序或过程提供给应用程序用户很灵活的选择各种操作及运行必须过程的方法。

1.3 命令定义

命令定义允许系统用户建立适合特别的应用需求的命令。这些命令与系统命令很类似。

系统中的每个命令都有一个命令定义和命令处理程序（CPP），命令定义定义这个命令，其中包括：

命令名

CPP

命令中有效的参数及值

有效性检查信息，系统可用它来检查命令的有效性

如果要命令提示，给出提示的说明内容

联机帮助信息

CPP 是命令进入时调用的程序，这时系统对命令本身进行有效性检查，不检查传给它的参数是否有效。

命令定义功能用于：

在要保持一个一致接口时，为系统用户建立一个唯一的命令。

为 CL 命令定义一个交互对话以满足系统用户的要求，这个功能可包括不同的参数缺省值。这样不用对某些参数输入值即可执行命令，对这个参数也可定义常量值。

IBM 提供的命令是不能修改的。

有关命令定义详细的内容，请看第九章。

1.4 菜单

系统提供了大量的菜单，允许用户只要选择菜单选项就能完成很多功能。用菜单来完成系统任务有很多好处：

用户不用了解 CL 命令和命令语法

大大减少输入量及产生错误的机会

在“应用显示程序设计”一书中有建立菜单的详细信息。

1.5 目标和库

一个目标是一个有名的存储空间。它由一组说明其属性的内容和一些数据组成。目标要占有一定的存储空间，能完成某个操作。目标的属性包括名字、类型、大小、生成目期、以及生成它的用户给出的一些说明。目标的值是存储在目标中信息的集合。例如，程序的值就是组成程序的编码，文件的值是组成文件记录的集合。目标的概念就是给出一个简单的术语，能用它来引用存在系统中的不同项目，而不用管这些项目到底是什么。

1.5.1 目标

大多数 CL 命令所完成的功能都是应用于目标的。一些命令能用于所有类型的目标而有一些仅能用于特别类型的目标。系统支持规定的目标类型，某些类型与大多数数据处理系统相类似。例如：

文件、程序、命令、库、队列、模块、服务程序

而有些则有很大不同，例如：

用户配置文件、作业描述、系统描述、设备描述

不同类型的目标有不同的操作属性，这些不同使每种类型都是唯一的。例如，由于文件是有数据的目标，它的操作属性就不同于包括指令的程序。

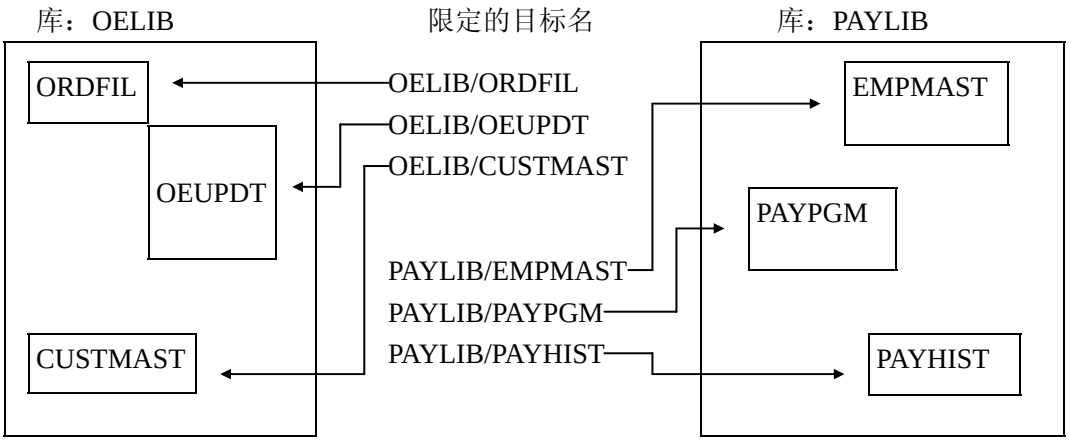
每个目标都有一个名字，目标名和目标类型用来标识这个目标。目标名是由建立目标的用户分配的，目标类型是由建立这个目标所用的命令确定的。例如，已生成了一个程序名为OEUPDT，那么总用这个名来引用这个程序，系统用目标名（OEUPDT）和类型（PGM）来查找目标并完成操作。几个目标可以有相同的名字，但它们必须有不同的类型或存在不同的库中，系统依据目标类型来维护完整性以避免造成功能上的混乱。例如，CALL 命令要调用一个程序目标来运行，如果规定 CALL 和一个文件名，那么命令要出错。

1.5.2 库

一个库用来把相关目标组成一组，且由名字来查找使用的目标。这样一个库是一组目标的目录。也可用库来把目标组成不同意义上的集合，例如，可以根据安全需要、备份、或处理来把目标分组。一个库中有多少目标，系统中能有多少个库仅由存储总量来限定，但不能多于 8000 以便保证完成保存操作。

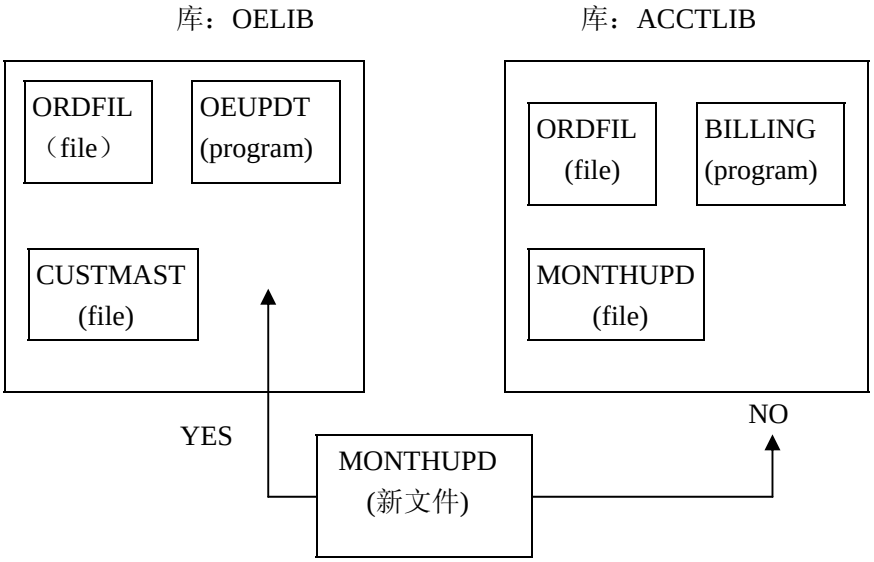
目标由库分组是逻辑上的分组。当建立库时，可以规定库所在的用户辅存池（ASP）。在一个库中建立的所有目标都与库在同一个 ASP 中。库中的目标必须物理的一个挨一个。库中目标的大少不受存储中临近空间总数的限制，系统会为目标找到所需的空间。多数类型的目标可以在生成时存在一个库中。目标放在库中时要分配给它一个公共授权，这个授权是在用 CRTLIB 生成库时由 CRTAUT 参数来规定的。多数类型的目标可从一个库移到另一个库，但一个目标不能同时在多个库中，当把目标移到不同库时，目标在存储中实际上不移动，只是用新的库来分配它，大多数类型的目标也能改名和复制。

库名用来给出目标名的另一级标识。如前所述，目标是由名字和类型来标识的，库名进一步限定目标名，目标名和库名放在一起组成目标的限定名，限定名告诉系统目标的名字和它所在的库名，下图给出两个库及库中目标的限定名：



同名及同类型的两 个目标可以放在不同的库中，两个同名目标要放在同一库中，它们的类型一定要不同，这种规定让程序用目标名引用目标来处理不同的目标。（目标名相同但所在库不同）。这样不用修改程序本身来达到运行程序的目的。而生成新目标的工作站用户

也不需要考虑在其它库中是否有同名目标。例如，在下图中，名为 MONTHUPD 的文件要加到库 OELIB 中，但不能放到 ACCTLIB 中，这是因为在此库中已有名为 MONTHUPD、类型为*FILE 的目标了。



库中的目标是由目标名和类型标识的，多数 CL 命令仅适合于一种类型，因此不一定要明确说明目标类型，对于适用于多种目标类型的命令，必须明确说明类型。

1.5.3 用库查找目标

目标名可以是限定名也可以是仅目标名本身。如果用限定名，系统要在规定的库中找目标，如果仅用目标名，系统要检索库列表，直到找到此目标的第一次出现或检索完库列表中的所有库却没有找到给定的目标。要检索的库以及它们被检索的顺序，是由库列表决定的，系统在作业启动时为它建立一个初始库列表。

注：库列表的初值及它的用户部分，可在系统值 QSYSLIBL 和 QUSRLIBL 中找到。这些系统值可由作业描述规定的值来取代。

一个库列表有四部分。第一部分是系统部分，它是最早检索的一些库。这部分规定了所有在系统中运行的作业使用的库。在安装系统时，这部分由 QSYS、QUSRSYS、QHLPSYS 和 QSYS2 组成。

第二部分是产品库。它是由系统在用户运行命令或菜单时修改的。根据完成功能的不同，产品库在作业运行时也各不相同。

第三部分是当前库。它是建立目标时所用的缺省库。用户可用 CL 命令规定作业的当前库。

第四部分是用户部分。它包括应用程序完成功能所使用的库。当安装系统时，它包括 QGPL、QTEMP，每个作业都有自己的 QTEMP，其它作业看不到，当作业结束时，清空 QTEMP 库。

当系统有多个用户定义的库时，不同作业的用户部分都不相同，下图给出库列表的一个例子：

检索顺序

QSYS	系统部分
QUSRSYS	
QHCPSYS	产品库 1
QPDA	产品库 2
QRPG	当前库
OELIB	
QELIB	用户部分
QGPL	
QTEMP	

仅用目标名和库列表来检索目标能使 AS/400 系统更易使用和更灵活。可给每个作业设计一个库列表以保证不用限定名即可找到正确的目标。这样做的好处是：

- 容易测试应用程序。当测试程序时可生成有样板数据的库。库中的目标名与正常产品库中的名字相同。有测试目标的库在库列表中放在正常产品库的前面，当程序测试好后，从库列表中移出这个库，程序用放在产品库中的目标操作，而不用修改程序中的目标名。
- 灵活使用系统中的库。当处理需要修改时，已有的库可能需要分成多个库有助于容易识别系统中的目标。这种修改不用改程序中的目标名，仅用修改作业使用的库列表。
- 能使不同用户使用同一个应用程序来处理不同的目标，每个用户或某组用户可以建立不同的库，每个用户作业的库列表保证使用到正确的目标。
- 同一应用程序处理不同的数据。例如，可能有多个公司或一组公司要分开处理，那么每个公司或一组公司有一个库，这样就可以把数据分开来放置。

由于有这些优点， 所以在使用已有目标时，不常用限定名，但此时要考虑可能出现的安全问题。例如，不用限定名调一个程序时，可能会由于某人修改了库列表而调用到了不同版本的程序，这时会出错。

1.6 信息

信息是用户、程序及过程与另外的用户、程序及过程通讯发出的消息。多数数据处理系统在系统和操作员之间提供通讯来处理错误及操作期间发生的条件。OS/400 也有信息处理功能支持程序和系统用户之间、程序之间、程序中的过程之间以及系统用户之间的双路通讯，它支持两类信息：

立即信息：它是由程序或系统用户生成的。生成后即发送，不留在系统中。

预先定义的信息：在使用之前已建立，建立时放在信息文件中，当使用时从这个文件中取出来。

由于信息做通讯用，在开发应用程序时要考虑它。下面的信息处理概念对应用程序开发是很重要的。

信息在信息文件中定义，它放在使用它的程序外部，在送信息时能在信息说明中给出不同的内容。这样，在信息修改时不用修改程序。这就允许同一程序使用和与不同语言传送信息的信息文件。

信息是从信息队列发送和接收的，它是系统中独立的目标。送往队列的信息能保存在队列里直到它被程序或工作站用户明确的接收到。

程序也能给需要程序信息的用户发送信息而不管用户是否在工作站注册，信息不必送往一个特别的设备。不同工作站可用一个程序而不用修改。

1.6.1 信息描述

信息描述为 OS/400 定义信息，它包括信息正文和有关替换变量的内容，以及由发送信息者提供的变量数据。信息描述存在信息文件中，每个描述在信息文件中都有一个唯一的标识，当发送信息时，信息文件和信息标识告诉系统使用哪个信息描述。

1.6.2 信息队列

当信息发送给过程、程序或一个系统用户时，它放在与之有关的信息队列中。程序及过程及用户用队列接收信息且能看到它。OS/400 为下列目标提供信息队列：

系统中的每个工作站
在系统中注册的每个用户
系统操作员
系统历史日志

可为应用程序生成另外的信息队列，送往信息队列的信息可以保留，所以接收到信息后不一定要立即处理。

1.7 测试功能

系统包括一些在程序运行时让程序员得到操作权的功能。这些功能用来设置一些不打算完成的操作。测试功能可用在工作站的批作业和交互作业。这时，要测试的程序必须放在测试环境中，这叫做调试方式。

测试功能主要针对找错。这比在源语句中找错要困难。通常，错误经常在产生的输出不是要求的结果时才表现出来。要找到这些错误，需要在某些点（叫做断点）把程序停下来，检查程序变量内容是否正确，在程序继续进行之前也要修改这些变量。使用测试功能，用户不需知道机器语言指令，也不用了解程序中有哪些特别的指令。

OS/400 测试功能让你做：

在断点停止运行程序

显示断点时的变量内容，在程序继续运行前修改变量值。说细内容请看第十章。

第二章　CL 程序设计

这章的重点是 ILE 而不是 OPM。因此，在这章中用‘过程’来代替‘程序’。但在讨论一般的 CL 命令时，仍用程序这个术语。

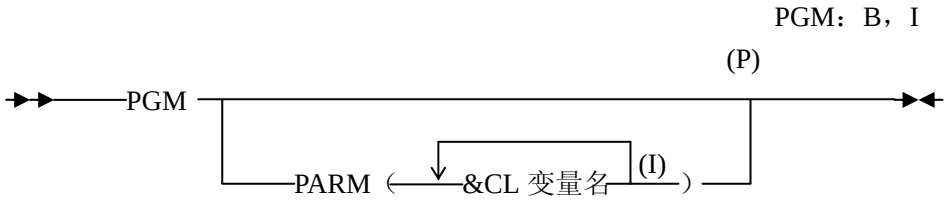
一个 CL 过程是一组 CL 命令，它告诉系统从哪得到输入，怎样处理，结果放在哪儿。过程要有一个名字，其它过程用此名来调用它，也用这个名字联编到程序中。与其它类型过程一样，CL 过程也要先输入源语句、编译、联编后才能运行。

在进入单个 CL 命令时，每个命令是分开处理的。当进入做为过程源语句的 CL 命令时，源语句可保留做以后修改用。命令编译形成模块，这个模块保留做永久的系统目标能连编到其它程序及运行。这样，CL 就相当于一个系统功能的高级程序设计语言。CL 过程保证一组命令处理的一致性，能用 CL 过程完成单个命令实现不了的功能，且能提供比单个命令运

行好得多的运行性能。

CL 过程在交互和批处理中都可使用，某些命令或过程只能在交互或批作业中处理。

CL 源语句由 CL 命令组成，不是所有 CL 命令都能做为源语句，有些只能用在 CL 过程或 OPM 程序中，可在 CL 参考手册中每个命令语法图右上角的标识来确定使用限制。如下图：



注：(I) 最多反复 40 次。

(P) 这点前以所有参数都可以按位置参数格式规定。

右上角的‘PGM：B，I’表示这个命令可用在交互和批作业中，但仅能用在 CL 程序或过程。如果此处没有 PGM 字样，说明此命令不能用做 CL 程序或过程的源语句。

源语句可从工作站交互地输入到源成员中，也可从设备上做为批作业的输入流。要用 CL 源语句生成一个程序，必须把源语句输入到数据库的源成员中，然后编译这些源语句生成 ILE 模块，再连编这个模块形成程序。

CL 过程可以做：

控制操作顺序及调用其它程序或过程
根据菜单选项的选择显示菜单及运行命令
读数据库文件
处理命令、程序或过程发布的错误条件，管理特定信息
为了控制应用程序的操作建立一些变量，例如日期、时间及外部指针。
为程序员提供预先定义的功能，例如启动子系统、备份文件，这样能减少经常使用的命令数，保证系统操作的性能一致。

CL 过程不可以做：

增加或修改数据库文件中的记录
使用打印机和 ICF 文件
在显示文件中使用子文件
使用程序描述的显示文件

2.1 生成 CL 程序

所有程序都按下列步骤来生成：

1. 输入源码，在大多情况下，源语句按应用程序设计要求的逻辑顺序输入到数据库文件中。
2. 生成模块，用 CRTCLMOD 命令和源语句生成系统目标，生成的 CL 模块可连编到程序中，一个 CL 模块由一个 CL 过程组成，其它的 HLL 语言的每个模块可包括多个过程。
3. 生成程序，用 CRTPGM 命令，这个模块（以及其它模块或服务程序）用于生成一个程序。

注：如果生成的程序仅有一个模块，那么可用 CRTBNDCL 命令，把第 2 步和第 3 步合成一步。

2.1.1 交互式输入

AS/400 提供许多菜单和显示来帮助程序员，象程序员菜单、命令入口显示、命令提示和 PDM 菜单，也能由用户配置文件来控制使用这些显示和菜单的权限。用户配置文件是由系统安全管理人员建立和维护的。

经常使用的源语句录入工具是 SEU。

2.1.2 批方式输入

可从软盘上用一个批输入流来建立 CL 源语句。下面给出它的基本部分，它用 SBMDKTJOB 命令提交给作业队列，输入流使用下面的格式：

```
// BCHJOB
CRTBNDCL PGM(QGPL/EDUPGM) SRCFILE(PERLIST)
// DATA FILE(PERLIST) FILETYPE(*SRC)
   .
   .              (CL Procedure Source)
   .
//
/*
// ENDINP
```

用这个输入流从联机源语句生成程序。如果要保留这些源语句，用 CPYF 命令把软盘上的内容复制到数据库文件中，然后用此文件生成程序，也可用外部介质上的 CL 源语句用 IBM 支持的设备文件直接生成 CL 模块。IBM 的软盘源文件为 QDKTSRC，带文件为 QTAPSRC。假定软盘上的源文件名为 PGMA，第一步用下面命令标识软盘上源文件的位置：

```
OVRDKTF FILE(QDKTSRC) LABEL(PGMA)
```

第二步用 CRTCLMOD 命令生成模块：

```
CRTCLMOD MODULE(QGPL/PGMA) SRCFILE(QDKTSRC)
```

在命令处理时，它把 QDKTSRC 当做数据库文件源文件。由于第一条命令，它到软盘上去找源文件，PGMA 生成在 QGPL 中，源语句仍留在软盘上。

2.1.3 CL 过程的组成部分

做为 CL 过程，每部分的源语句实际上都是一个 CL 命令，在典型的 CL 过程中它可以分成下列几个基本部分：

PGM 命令　　PGM　PARM(&A)

这是一个可选值，表示过程的开始及可接收的参数

说明命令　　（DCL，DCLF）

在使用变量时，说明命令要放在除 PGM 命令之外的所有命令前面

CL 处理命令　　CHGVAR，SNDPGMIUSR，OVRDBF，DLTF……

用来管理常量及变量

逻辑控制命令　　IF，THEN，ELSE，DO，ENDDO，GOTO

用来控制过程内部的处理

内部函数　　%SUB，%SWICH，%BIN

用于计算、关系或逻辑表达式

程序控制命令　　CALL，RETURN

用来把控制传递给其它程序

过程控制命令　　CALLPRC，RETURN

用来把控制传递给其它过程

ENDPGM 命令　　ENDPGM

这是可选的，表示程序的结束

以上命令使用的顺序、组合及扩展都依据应用程序的逻辑和设计。在生成过程、处理命令时，引用的目标必须已经存在。

在大多数情况下，要成功地运行一个过程，必须有：

一个显示文件：用来给出设备显示信息。如果过程使用显示，在生成过程前必须先生成显示文件，且在过程中用 DCLF 说明它，详细内容看本书 5.12 及数据管理一书。

一个数据库文件：CL 过程可以读数据库中的记录。如果用数据库文件，必须在生成过程前生成物理文件或逻辑文件，可用 DDS、SQL 或 IDDU 来定义文件中的记录格式。在过程中，要用 DCLF 命令来说明它。详细内容，请看本书 5.2 及 DB2 数据库程序设计一书。

其它程序：如果用 CALL 命令，那么在 CALL 运行前，调用的程序必须存在，但在编译使用 CALL 命令的程序时，它可以不存在。

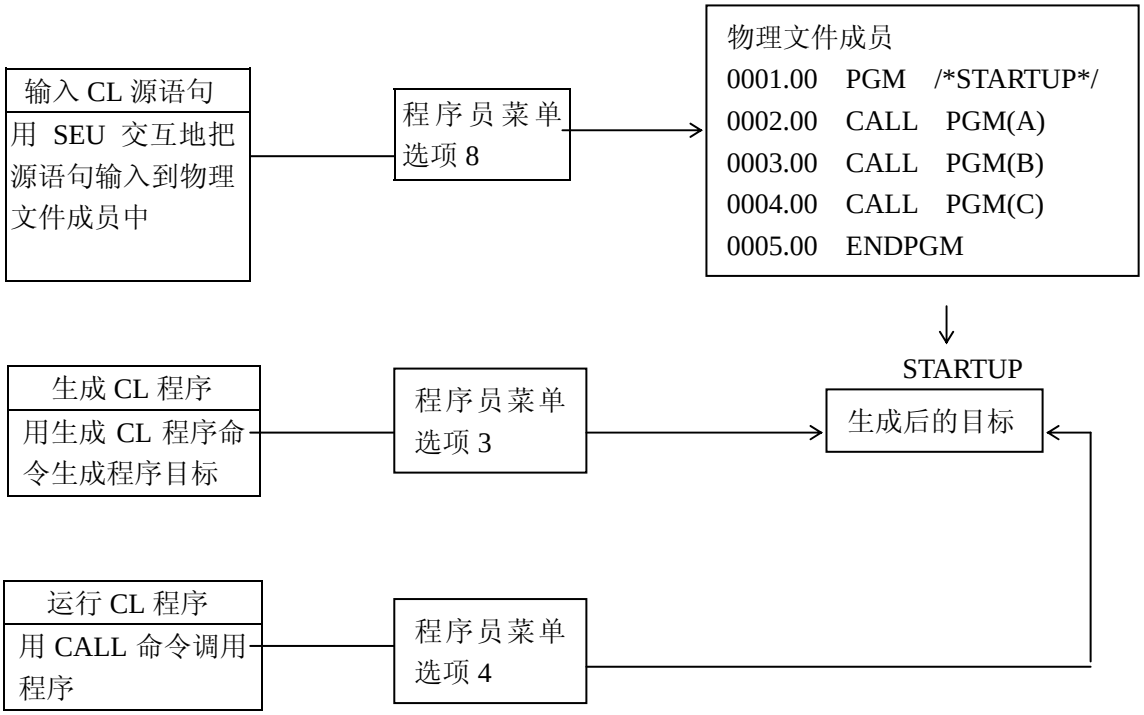
其它过程：如果用 CALLPRC 命令，在过程运行时被调用的过程必须存在，但编译时可以不存在。

2.1.4 简单的 CL 程序例子

一个 CL 程序可简单、可繁杂，这要看你设计的想法。下面的例子给出在一天开始时操作员正常要做的几件事情（调用程序 A、B、C），可用下列编码生成一个名为 STARTUP 的 CL 过程：

```
PGM  /*  STARTUP  */
CALL  PGM(A)
CALL  PGM(B)
CALL  PGM(C)
ENDPGM
```

在此例中，用程序员菜单生成程序，也可用 PDM。下图给出输入、生成及使用程序的步骤：



要输入 CL 源语句：

在程序员菜单选 8，在 PARM 字段写 STARTUP，在 TYPE 字段规定 CLLE，按执行键，在 SEU 显示中，用 I 行命令进入 CL 语句。

```
Columns.....:   1  71            Edit                        QGPL/QCLSRC
Find......: ____________________________________________          STARTUP
FMT A* .....A*. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
```

```
********** Beginning of data *********************************
.......
.......
.......
.......
.......
.......
```

在输入完后，用 F3 键结束 SEU，用结束屏的缺省值，按执行键回到程序员菜单，选 3（生成目标），不用修改此显示中的任何信息。

注：此时引用的程序（A、B、C）不必须已存在。

在程序生成后，用程序员菜单中的选项 4 调用它，如果要运行它，则程序 A、B、C 必须都存在。

2.2 CL 过程中使用的命令

一个 CL 过程仅包括 CL 命令。它包括 IBM 支持的及自己定义的命令。某些 CL 命令，象 TFRJOB 和 SBMJOB 都有 RQSDTA 或 CMD 参数，它用另外的 CL 命令做参数值。仅能用在 CL 过程中的命令不能用做 RQSDTA 或 CMD 参数的值。

下表给出在 CL 过程中经常使用的命令，从中你可选择这方面命令来满足需要。这些命令以功能分组，帮助你理解这章的其余部分内容。图中的（1）指出这些命令仅能用在 CL 过程中。

系统功能	命　　令	命令功能
修改过程控制	CALL (Call)	调用一个程序
	CALLPRC (Call Procedure) (1)	调用一个过程
	RETURN (Return)	返回到引起程序或过程运行的下一条命令
CL过程界限	PGM (Program) (1)	指出CL过程源码的开始
	ENDPGM (End Program) (1)	指出CL过程源码的结束
CL过程逻辑	IF (If) (1)	根据逻辑表达式的值执行命令
	ELSE (Else) (1)	对IF命令为假条件定义采取的动作
	DO (Do) (1)	指出DO组的开始
	ENDDO (End Do) (1)	指出DO组的结束
	GOTO (Go To) (1)	转移到另外的命令
CL过程变量	CHGVAR (Change Variable)(1)	修改CL变量的值
	DCL (Declare) (1)	说明一个变量
替换	CHGVAR (Change Variable)(1)	修改CL变量的值
	CVTDAT (Convert Date) (1)	修改日期格式
数据区	CHGDTAARA (Change Data Area)	修改数据区
	CRTDTAARA (Create Data Area)	生成一个数据区
	DLTDTAARA (Delete Data Area)	删除一个数据区
	DSPDTAARA (Display Data Area)	显示一个数据区

	RTVDTAARA (Retrieve Data Area)	把数据区的内容复制到一个CL变量中
文件	ENDRCV (End Receive) (1)	取消由前面的RCVF，SNDF或SNDRCVF命令对一个显示文件发出的输入请求
	DCLF (Declare File) (1)	说明一个显示文件或数据库文件
	RCVF (Receive File) (1)	从显示文件和数据库文件中读记录
	RTVMBRD (Retrieve Member Description) (1)	取得数据库文件成员的描述
	SNDF (Send File) (1)	往显示文件中写记录
	SNDRCVF (Send/Receive File) (1)	往显示文件中写记录,在用户回答后读记录
	WAIT (Wait) (1)	等待从显示文件发出的SNDF，RCVF或SNDRCVF命令接收数据
信息	MONMSG (Monitor Message) (1)	监控送往程序信息队列的逃逸、状态和通知信息
	RCVMSG (Receive Message) (1)	把信息从信息队列复制到一个CL变量中
	RMVMSG (Remove Message) (1)	从信息队列取消信息
	RTVMSG (Retrieve Message) (1)	把预先定义的信息从信息文件复制到CL变量中
	SNDPGMMSG (Send Program Message) (1)	往信息队列发送程序信息
	SNDRPY (Send Reply) (1)	给查询信息的发送者发送回答信息
	SNDUSRMSG (Send User Message)	给显示工作站或系统操作员发送消息或查询信息
混杂命令	CHKOBJ (Check Object)	检查目标是否存在及使用目标必须有的权限
	PRTCMDUSG (Print Command Usage)	产生一个用在某组CL过程中的一组命令中的交叉引用表
	RTVCFGSRC (Retrieve Configuration Source)	对生成的已存在的配置目标建立一个CL命令源码且把它放在源文件成员中
	RTVCFGSTS (Retrieve Configuration Status) (1)	从三个配置目标（线路、控制器和设备）中取得配置状态
	RTVJOBA (Retrieve Job Attributes) (1)	取得一个或多个作业属性的值且把它们放到CL变量中
	RTVSYSVAL (Retrieve System Value) (1)	取得系统值并且把它放到一个CL变量中
	RTVUSRPRF (Retrieve User Profile) (1)	取得用户配置文件属性并把它放到CL变量中
程序生成命令	CRTCLMOD (Create CL Module)	生成一个CL模块
	DLTMOD (Delete Module)	删除一个模块
	DLTPGM (Delete Program)	删除一个程序
	CRTBNDCL (Create Bound Control Language Program)	生成一个联编的CL程序
	CRTPGM (Create Program)	生成一个程序
	CRTSRVPGM (Create Service Program)	生成一个服务程序

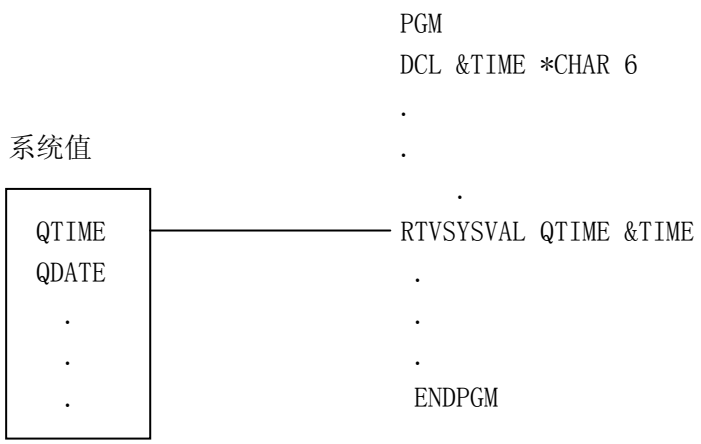
2.3 使用 CL 过程

CL 程序设计是一个很灵活的工具，能完成不同的功能。一般的，用 CL 过程能：

使用变量、逻辑控制、表达式和内部函数来维护和处理 CL 过程中的数据：

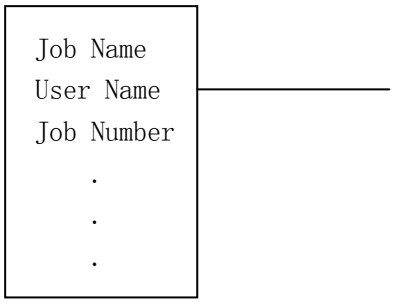
```
PGM
DCL &C *LGL
DCL &A *DEC VALUE(22)
DCL &B *CHAR VALUE(ABCDE)
.
.
.
CHGVAR &A (&A + 30)
.
.
.
IF (&A < 50) THEN(CHGVAR &C '1')
.
DSPLIB ('Q' || &B)
.
IF (%SST(&B 5 1)=E) THEN(CHGVAR &A 12)
.
.
.
ENDPGM
```

在过程中使用系统值做变量：



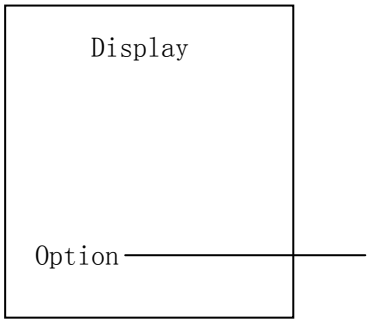
在过程中用作业属性做变量：

作业属性




```
PGM
DCL &USER *CHAR 10
.
.
.
RTVJOBA USER(&USER)
.
.
.
ENDPGM
```

用显示文件传输数据：



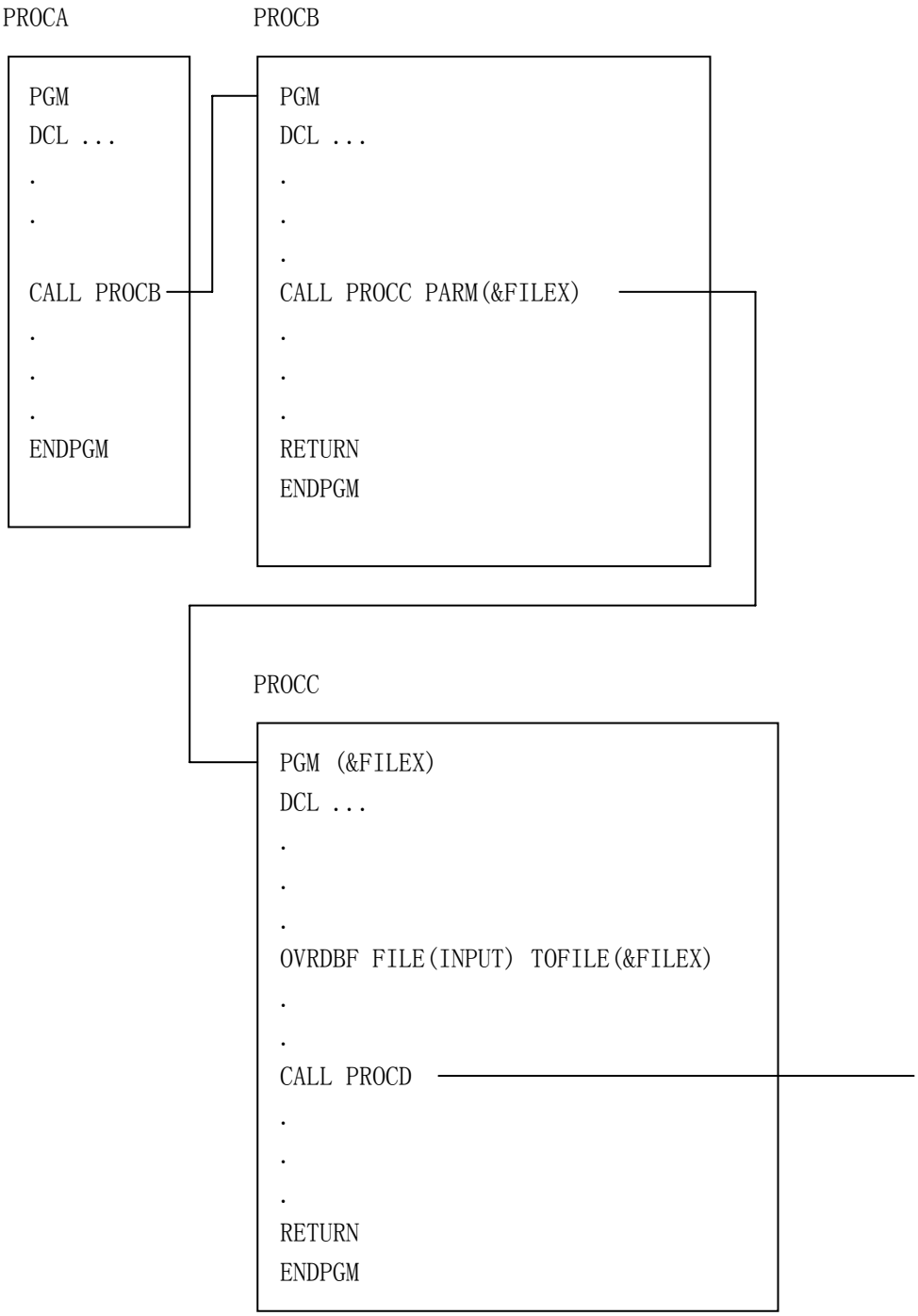

```
PGM
DCL FILE(DISPLAY)
DCL &OPTION *CHAR 2
.
.
.
RCVF ...
IF (&OPTION *EQ 1) THEN(CALL PGMA)
.
.
.
ENDPGM
```

用 CL 过程监控作业的错误信息，如需要，采取相应的动作：

```
PGM

MONMSG MSGID(CPF0001) EXEC(GOTO ERROR)
CALL PROGA
CALL PROGB
RETURN
```

```
ERROR:  SNDPGMMSG MSG('A CALL command failed') MSGTYPE(*ESCAPE)
        ENDPGM
```

控制过程和程序间的操作且在之间传递参数和替换文件：



作为一个控制过程，CL 过程也能调用用其它语言写的过程，下面解释如何在程序里的 CL 过程和 RPG　IV 及 ILE　COBOL 过程中传递控制。要用这个程序，工作站用户要请求 PGMA，它控制整个应用程序。

PGMA 包括：

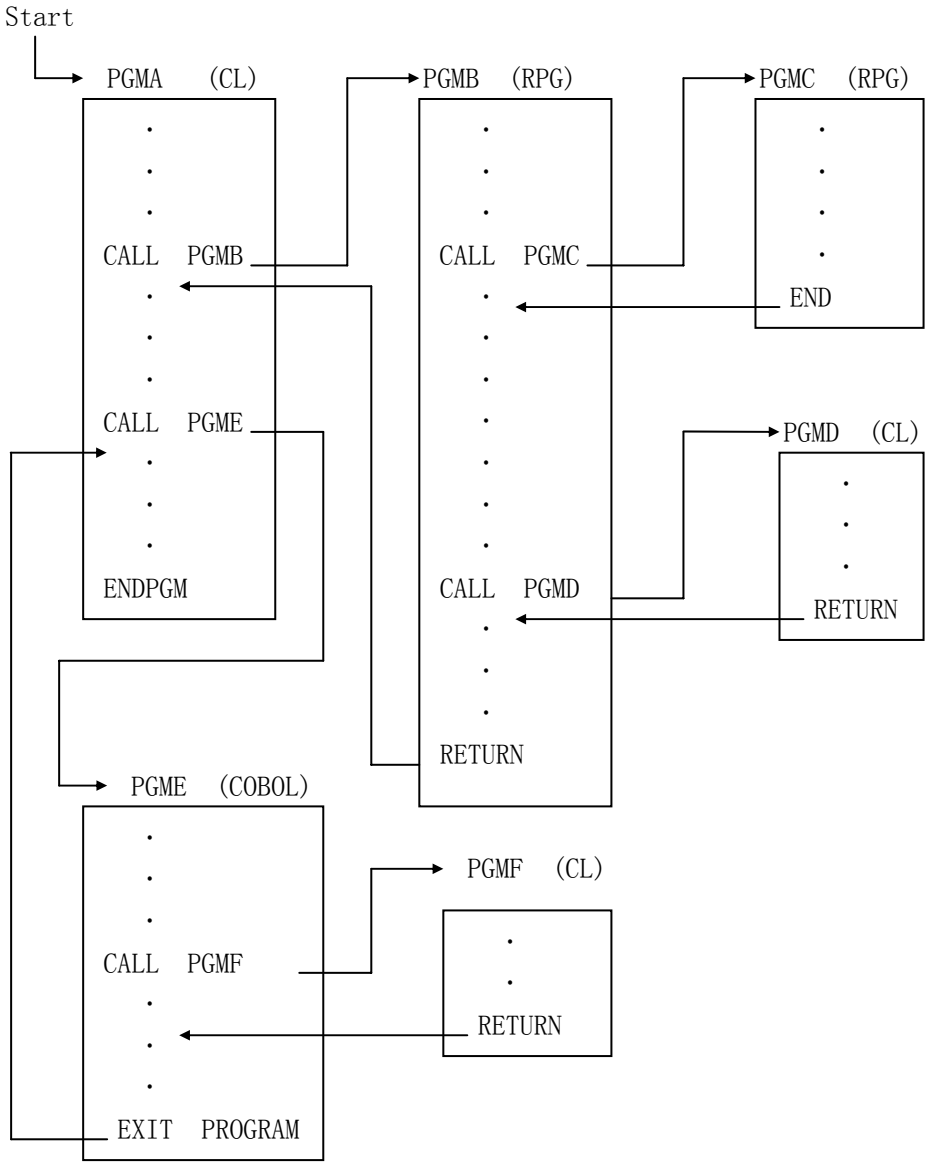
一个 CL 过程（PGMA），调用 RPG 过程（PGMB）

RPG 过程（PGMB）调用另外一个 RPG 过程（PGMC）

PGMB 调用 CL 过程（PGMD）

PGMA 调用 COBOC 过程（PGME）

PGME 调用 CL 过程 PGMF



这些过程能用下面给出的命令来生成，可把各自的源语句输入到不同的源成员中：

```
CRTCLMOD PGMA
CRTRPGMOD PGMB
CRTRPGMOD PGMC
CRTCLMOD PGMD
```

```
CRTCBLMOD PGME
CRTCLMOD PGMF
CRTPGM PGM(PGMA) +
       MODULE(PGMA PGMB PGMC PGMD PGME PGMF) +
       ENTMOD(*FIRST)
```

2.4 处理变量

CL 过程由命令组成，命令本身又有命令语句、参数及参数值。参数值可以是变量、常量和表达式。变量是一个命名的可变的值，用它的名字来修改和访问。变量可用在 CL 命令多数的参数中，当变量做参数值且运行命令时，变量的值做为参数值，命令每次运行时，变量可有不同的值。仅在过程和程序中能用变量及其表达式做参数值。变量不是放在库中，它们不是目标，它们的值在使用它们的程序不被调用时会破坏掉。变量名以&号开始后跟 10 个以内的字符。&后的第一个字符必须是字母。使用变量让 CL 程序设计变得更灵活，这是由于目标的高级管理可由特定的应用修改其内容。例如，可写 CL 程序来指挥其它程序的处理或几个工作站的操作，而不用规定要控制哪个程序或工作站，这些都可做为 CL 过程中的变量，在过程运行时可以定义变量的值。

所有的变量在使用之前都要说明：

说明变量：用 DCL 命令定义变量及其属性。属性包括类型、长度和初值。

```
DCL  VAR(&AREA)  TYPE(*CHAR)  LEN(4)  VALUE(BOOK)
```

说明文件：如果 CL 程序使用文件，必须在 DCLF 命令中的 FILE 参数中规定文件名，文件由记录格式及字段组成。在编译时，DCLF 命令隐含地说明字段的变量和指示器。

如果文件的 DDS 中有一个记录，记录中有两个字段（F1 和 F2），那么在程序中自动地说明两个变量&F1 和&F2。

```
DCLF   FILE(MCGANN/GUIDE)
```

如果文件不是用 DDS 生成的物理文件，说明的变量为整个记录所用，变量名与文件同名、长度与文件的记录长相同。

DCL 命令必须放在 PGM 后、其它命令前，多个说明命令可用任何顺序排列。

变量也可做：

在作业和过程间传递信息，请看第三章。

在过程和设备显示间传递信息，请看 5.2.7。

条件处理命令，请看 2.5。

生成目标，一个变量可用来放目标名或库名，下面的例子中，第一行是用规定的库生成物理文件，第二行是用变量代替库名。

```
CRTPF  FILE(DSTPRODLB/&FILE)
CRTPF  FILE(&LIB/&FILE)
```

不能用变量修改命令名，键字段用作 CALLPRC 命令的过程名，但在处理 CL 过程时可用提示功能来修改命令的参数，请看 6.5。也可用 QCAPCMD 功能把命令的键字和参数汇编

在一起来进行处理，有关 QCAPCMD 的详细内容请看 6.1。

2.4.1 说明变量

用最简单的格式 ，DCL 命令有下列参数。

DCL VAR（变量名） TYPE { *CHAR / *DEC / *LGL } LEN（长度） VALUE（初值）

使用 DCL 命令，要遵守下列规定：

CL 变量名要以&号开始后跟 10 个以内的字符。&后的第一个字符必须是字母，其余可是字母数字。

CL 变量的值须是以下之一：

—5000 以内个字符串

—最多 15 个数字，9 位小数的压缩十进制值

—逻辑值‘1’或‘0’

如果没有规定初值，则假定下列值为初值：

—十进制变量为 0

—字符变量为空格

—逻辑变量为‘0’

对十进制或字符类型，如果规定了初值但没给出长度，缺省长度为与初值等长，对字符类型，长度最长为 5000。

用 DCL 语句说明参数为变量。

2.4.2 用变量规定列表或限定名

参数的值可以是一个列表，例如，CHGLIBL 命令中的 LIBL 参数就需要一个库列表，列表中的各项可以是变量。

```
CHGLIBL    LIBL(&LIB1    &LIB2    &LIB3)
```

这时，每个变量必须分开用 DCL 说明：

```
DCL VAR(&LIB1) TYPE(*CHAR) LEN(10) VALUE(QTEMP)
DCL VAR(&LIB2) TYPE(*CHAR) LEN(10) VALUE(QGPL)
DCL VAR(&LIB3) TYPE(*CHAR) LEN(10) VALUE(DISTLIB)
CHGLIBL LIBL(&LIB1 &LIB2 &LIB3)
```

变量的各项不能象下面那样规定为字符串的列表：

```
DCL   VAR(&LIBS) TYPE(*CHAR) LEN(20) +
      VALUE('QTEMP QGPL DISTLIB')
CHGLIBL LIBL(&LIBS)
```

这时，系统不把它做为分开各项的列表处理，那么要出错。
也能用变量规定一个限定名，每个限定名要用 DCL 来分别说明：

```
DCL VAR(&PGM) TYPE(*CHAR) LEN(10)
DCL VAR(&LIB) TYPE(*CHAR) LEN(10)
CHGVAR VAR(&PGM) VALUE(MYPGM)
CHGVAR VAR(&LIB) VALUE(MYLIB)
.
.
.
DLTPGM PGM(&LIB/&PGM)
ENDPGM
```

在此例中，程序和库是分别说明的，它俩的名字不能用一个变量说明，象下面那样是错误的：

```
DCL VAR(&PGM) TYPE(*CHAR) LEN(10)
CHGVAR VAR(&PGM) VALUE('MYLIB/MYPGM')
DLTPGM PGM(&PGM)
```

这时系统把它做为一个字符串，而不是当作两个目标，如果限定名非要做为一个字符串，要用内部函数%SUB 和*TCAT 功能分配一个目标或库名来给各自的变量，详细内容请看 2.5.8 及第九章。

2.4.3 变量中的小写字符

一些保留值，比如*LIBL，做变量时要用大写字符，它们做字符串时，要用引号括起。例如，在命令中要把一个库名做为变量，正确的写法写：

```
DCL VAR(&LIB) TYPE(*CHAR) LEN(10) VALUE('*LIBL')
DLTPGM (&LIB)/MYPROG;
```

不正确的写法如：

```
DCL VAR(&LIB) TYPE(*CHAR) LEN(10) VALUE('*libl')
```

如果 VALUE 参数没放在引号内，也是错误的，这是因为没有引号要自动转换或大写，这个错误经常发生在这个参数从显示设备做为输入的字符串传送给过程或程序时，显示设备是用小写来输入的。

2.4.4 代替保留值或数值参数的值的变量

某些 CL 命令的参数允许数字或预先定义的值。这时，也用字符变量来代表这些参数的值。

每个参数只能接收一定类型的值。参数值可以是整数、字符串、保留值、某类变量以及这些的组合。如果一个参数允许数字值也允许保留值，也能用变量做参数值。如果用保留值

做变量，那么这个变量必须说明为类型为*CHAR。例如 CHGOUTQ 命令的参数 JOBSEP 的值，可以是数字 0—9，也可是予先定义的保留值*SAME。那么可以写下面的 CL 过程来把 JOBSEP 定义做字符变量：

```
        DCL &SEP *CHAR LEN(4)
        DCL &FILNAM *CHAR LEN(10)
        DCL &FILLIB *CHAR LEN(10)
        DCLF.....
        .
        .
        .
LOOP:   SNDRCVF.....
        IF (&SEP *EQ IGNR) GOTO END
        ELSE IF (&SEP *EQ NONE) CHGVAR &NRESP '0'
        ELSE IF (&SEP *EQ NORM) CHGVAR &NRESP '1'
        ELSE IF (&SEP *EQ SAME) CHGVAR &NRESP '*SAME'
        CHGOUTQ OUTQ(&FILLIB/&FILNAM) JOBSEP(&NRESP)
        GOTO LOOP
END:    RETURN
        ENDPGM
```

在此例中，工作站用户在显示上输入说明某些输出队列的作业分隔符的数字。变量 &NRESP 是一个字符变量，放数字和预先定义的值，那么在 CHGOUTQ 命令中的 JOBSEP 参数就会认识这些值,程序中显示文件的 DDS 用 VALUES 键字来限制只能用 IGNR,NONE，NORM 或 SAME 来响应。

如果参数允许数字类型的值（*INT2，*INT4，或*DEC）且用户也不想用保留值（例如 *SAME），那么在这个参数中可以使用数字型变量。

实现这个功能的另外一个方法是在 CL 过程中使用提示。

2.4.5 修改变量的值

可用 CHGVAR 命令来修改 CL 变量的值，可以修改为的值是：

修改为一个常量:

```
CHGVAR    VAR(&INVCMPLT)    VALUE(0)
```

或

```
CHGVAR    &INVCMPLT  0
```

&INVCMPLT变为零。

修改为另一个变量的值：

```
CHGVAR    VAR(&A)    VALUE(&B)
```

或

```
CHGVAR    &A   &B
```

&A 设为变量&B的值。

修改为表达式赋值后的值：

```
CHGVAR    VAR(&A)    VALUE(&A + 1)
```

或

```
CHGVAR    &A   (&A + 1)
```

修改为内部函数%SST 的结果值：

```
CHGVAR    VAR(&A)    VALUE(%SST(&B 1 5))
```

&A 为&B 的前五个字符。

修改为内部函数%SWITCH 的结果值：

```
CHGVAR   VAR(&A)   VALUE(%SWITCH(0XX111X0))
```

如果作业开关1和8是0，开关4、5、6是1，则&A为1，否则&A为0。

修改为内部函数%BIN 的结果值：

```
CHGVAR   VAR(&A)   VALUE(%BIN((%B 1 4))
```

&B 的前 4 个字符转换成等值的十进数，放在变量&A 中。

CHGVAR 命令也用于取出或修改本地数据区的内容。例如下列命令把数据区的前 10 个字节变为空格，且取出数据区的部分内容：

```
CHGVAR %SST(*LDA 1 10) ' '

CHGVAR &A %SST(*LDA 1 10)
```

对一个逻辑变量必须修改成逻辑值，对十进制变量，可以用十进制或字符，对字符变量，可以接收字符或十进制值。

在对字符变量规定十进制值时，注意下面几点：

字符变量要右对齐，必要的话用前置零填充。

字符变量要足够长以放下小数点或符号。

减号放在值的最左边。

例如，&A 是字符变量，要修改为十进制变量&B，&A 为 6 位长，&B 为 5 位长 2 位小数，&B 的值为 123，则&A 的结果值为 123.00。

在对十进制变量规定字符值时，注意以下几点：

小数点是由在字符值中的小数点位置定的。如果字符值不包括小数点，则小数点放在值的最右位置。

字符值可在值的最左边有十、一号，符号和字符间不能有空格。如果字符值无符号，则认定为正值。

如果字符值在小数点后的字符多于十进变量能放下的值，字符要被截断，但如果超出的字符值是在小数点左边，则不发生截断但出错。例如：&C 是十进变量要修改为字符变量 D 的值，&C 是 5 位长 2 位小数，&D 是 10 位长，现在值为 123.1bbbb（这里 b 是空格），则&C 的结果值是 123.10。

2.4.6 命令参数末尾的空白

有些命令定义为 VARY(*YES)值，它把值的长度传输成引号内的字符数，当 CL 变量规定用这种方法定义的参数值时，系统在确定要传送的变量长度前要去掉尾部的空白。如果此空白对参数来讲是有意义的，那么必须用特别的方法来保证传送的长度中包括它们，大多数命令的参数在定义和使用时都不会发生此类情况。类似的情况在 OVRDBF 中的 POSTION 参数中的键值元素也会发生，当发生这种情况时，要构造一个命令串来把参数值放在引号中，然后把串传给 QCMDEXC 或 QCAPCMD 来处理，就能得到预期的结果。

下面是一个用来运行 OVRDBF 命令的程序，尾部的空白能做为键值的一部分，同样的技术也可用于用 VARY(*YES)定义参数的命令，末尾空白必须传送给参数。

```
            PGM         PARM(&KEYVAL &LEN)
/*    此程序介绍如何规定一个有结尾空格的键值来做为CL程序中的OVRDBF     */
/*    命令中位置参数的一部分。                                          */
/*    这个键值的元素是用VARY(*YES)参数定义的，在ELEM命令定义语句中     */
/*    此参数的说明规定了，用这种方法定义的参数是作为CL变量规定的，     */
/*    它的长度作为去掉结尾空格的变量传送。                              */
/*    用单引号限制此键值长度的调用，QCMDEXC可作为这个动作的循环。       */
/*    参数--                                                           */
      DCL         VAR(&KEYVAL) TYPE(*CHAR) LEN(32)  /*  请求键的
                                                  的值，定义为32个字符 */
      DCL         VAR(&LEN) TYPE(*DEC) LEN(15 5)  /*  用此键值的
                                                长度1—32的任何值都可用 */
/*    替换命令的结果串传给QCMDEXC（写两个引号才能得到一个引号的结果） */
      DCL         VAR(&STRING) TYPE(*CHAR) LEN(100) +
                    VALUE('OVRDBF FILE(X3) POSITION(*KEY 1 FMT1  '' '')
/*    位置标记  123456789 123456789 123456789 123456789                */
      DCL         VAR(&END) TYPE(*DEC) LEN(15 5)  /* 计算 &STRING
                                                      键的结尾的变量    */
      CHGVAR      VAR(%SST(&STRING 40 &LEN)) VALUE(&KEYVAL)  /*把
                          键值送在命令串中，让紧接引号后的QCMDEXC使用*/
      CHGVAR      VAR(&END) VALUE(&LEN + 40)  /*  定位在键值的最
```

```
                                                    后一个字符之后  */
            CHGVAR      VAR(%SST(&STRING &END 2)) VALUE(''))'')  /*  把结
                                                束引号和括号放在结尾参数上  */
            CALL        PGM(QCMDEXC) PARM(&STRING 100)  /*  调用处理命
                                                          令过程   */
            ENDPGM
```

注：如果用 VARY(*YES)和 RTNVAL(*YES)且传送 CL 变量，传送的是变量的长度而不是 CL 变量中数据的长度。

2.4.7 在 CL 过程中写注释

如果是在 CL 过程中写注释或对命令加注释，使用字符对 /* 和 */。注释放在这对符号中间。

注释的起始界限符/*，要三个字符，除非/*出现在命令串的头二个位置。此时，在命令前/*后不用跟空格。

可用下列方法之一写三个字符的注释起始界限（b 为空格）：

```
/*b
b/*
/**
```

这样，注释起始界限符可由四种方法输入：

从命令串的第一个位置开始
前加一空格
后跟一空格
后跟一个星号（/**）

在下面的过程中，写一个注释来提示用户可以响应的菜单选项：

```
          PGM /* ORD040C Order dept general menu */
          DCLF FILE(ORD040CD)
 START:   SNDRCVF RCDFMT(MENU)
          IF (&RESP=1) THEN(CALL CUS210)
          /*Customer inquiry */
          ELSE +
                IF (&RESP=2) THEN(CALL ITM210)
                /**Item inquiry */
                ELSE +
                      IF (&RESP=3) THEN(CALL CUS210)
                       /* Customer name search */
                      ELSE +
                            IF (&RESP=4) THEN(CALL ORD215)
                            /** Orders by cust */
                            ELSE +
                                  IF (&RESP=5) THEN(CALL ORD220)
```

```
                                    /* Existing order */
                                    ELSE +
                                          IF (&RESP=6) THEN(CALL ORD410C)
                                          /** Order entry */
                                          ELSE +
                                                IF (&RESP=7) THEN(RETURN)

          GOTO START
          ENDPGM
```

2.5 CL 过程中的控制处理

CL 过程中的命令是按顺序处理的，处理完一个再处理后面的一个。可用修改逻辑流程的命令来改变这个处理的顺序，这些命令可以是条件的（IF）或无条件的（GOTO）

无条件转移意思是不管转移指令发生时存在什么样的条件，过程执行都转移到过程中的某个命令或一组命令中，这时要用 GOTO 命令。

条件转移是在一定的条件下，处理要转移到过程内非连续的段或命令中，可转移到过程中的任何语句。因为转移发生在某个条件为真时，所以叫做条件处理。这时经常用 IF 命令，如果条件不为真，可用 ELSE 命令规定替代的处理。DO 命令生成一组一起处理的命令，即在某种条件下，执行一组命令。

2.5.1 使用 GOTO 命令及标号

GOTO 命令处理无条件转移，无论什么时候遇到 GOTO 命令，都要直接执行过程中的另一部分（用一个标号指出），这种转移不依赖表达式的求值。转移到标号指定的语句后，从这个语句开始，继续顺序处理。

它不返回到 GOTO 命令处，除非由另外指令特别指定返回。可以往前或向后转移，不能用 GOTO 语句走到过程外的标号处。GOTO 命令没有参数，它只包含要转移语句的标号：

```
          GOTO  CMDLBL(标号)
```

标号标识过程中的语句，指出 GOTO 命令执行的方向，要使用 GOTO 语句，要转移的去处必须有一个标号：

```
          PGM
          .
          .
          .
START:    SNDRCVF RCDFMT(MENU)
          IF (&RESP=1) THEN(CALL CUS210)
          .
          .
          .
          GOTO START
          .
          .
```

```
.
ENDPGM
```

此例中，标号是 START，标号最多 10 个字符，结尾必须有冒号，命令名和标号之间有空格。

2.5.2 使用 IF 命令

IF 命令用来设置一个条件。如果为真，要规定在过程中运行的语句或一组语句，与 IF 一起可以规定 ELSE 命令，来规定条件不为真时执行的语句。

命令包括一个表达式（它用来测试真、假）和一个 THEN 参数，它现定表达式为真时要采取的动作，IF 命令的格式为：

```
IF  COND(逻辑表达式)  THEN(CL 命令)
```

逻辑表达式可以是一个逻辑变量或常量，或多个操作之间的关系，然后计算表达式是真还是假，详细内容请看 2.5.6。

如果逻辑表达式的条件为真，处理 THEN 参数中的命令，它可以是一个命令，也可以是一组命令，如果条件不为真，运行下一个顺序的命令。COND 和 THEN 都是命令的键字，它们可以省略。下面是正确的写法：

```
IF COND(&RESP=1) THEN(CALL CUS210)
IF (&A *EQ &B) THEN(GOTO LABEL)
IF (&A=&B) GOTO LABEL
```

在 IF 和键字或值（&A‥）之间要有空格，在键字之间不允许有空格，如果有，用左边的括号括住值。

下面是 IF 的例子，假定在编码前，&A 的值为 2，&4 的值为 4。

```
        IF (&A=2) THEN(GOTO FINAL)
        IF (&A=3) THEN(CHGVAR &C 5)
        .
        .
        .
FINAL:  IF (&C=5) CALL PROGA
        ENDPGM
```

在这种情况下，处理第一个 IF 之后转 到 FINAL 处，跳过中间的语句，不返回第二个 IF 命令处，在 FINAL 中，由于要测试&C 是否为 5，此时为假，则不调用 PROGA，而执行 ENDPGM。它指出过程结束，控制返回给调用它的过程。

如果变量的初值不同，则用同样的编码，处理逻辑会不同，比如开始&A 为 3，&C 为 4，则第一个 IF 为假，则不执行 GOTO，而顺序执行第二个 IF。测试为真，则把&C 改为 5。继续执行后续语句，当做到最后一个 IF，此时&C=5,为真，则调用 PROGA。

几个相邻 IF 语句是独立运行的，例如：

```
        PGM /*  IFFY  */
      DCL &A..
      DCL &B..
      DCL &C..
      DCL &D..
      DCL &AREA *CHAR LEN(5) VALUE(YESNO)
      DCL &RESP..
      IF (&A=&B) THEN(GOTO END)   /*  IF #1  */
      IF (&C=&D) THEN(CALL PGMA)   /*  IF #2  */
      IF (&RESP=1) THEN(CHGVAR &C 2)  /*  IF #3  */
      IF (%SUBSTRING(&AREA 1 3) *EQ YES) THEN(CALL PGMB)  /*  IF #4  */
      CHGVAR &B &C
      .
      .
      .
END:  ENDPGM
```

在上例中，如果&A≠&B，运行下面语句，假如&C=&D，则调用 PGMA，当 PGMA 返回时，考虑第三个 IF，等等。这些顺序的 IF 语句之间逻辑和处理都是不同的，有 ELSE 语句和嵌套的 IF，详细内容请看 2.5.4。

一个嵌套的命令是完全包括在另一个命令的参数中，在下例中，CHGVAR 和 DO 命令是嵌套的：

```
IF (&A *EQ &B) THEN(CHGVAR &A (&A+1))


IF (&B *EQ &C) THEN(DO)
                    .
                    .
                    .
                    ENDDO
```

2.5.3　使用 DO 命令和 DO 组

DO 命令处理一组命令，这一组命令是用 DO 和 ENDDO 之间的命令组成的，这组命令的处理通常是以有关命令为条件的。它经常与 IF、ELSE 或 MONMSG 一起使用。例如，

```
IF  (&A=&B) THEN(DO)
                    .
                    .          DO 组
                    .
                    ENDDO
  .
  .
  .
```

```
ENDPGM
```

如果逻辑表达式（&A=&B）为真，则执行 DO 组，如果不为真，则跳过 DO 组，处理 ENDDO 后的命令。

在下例中，如果&A≠&B，则调用 PGMB，不调用 PGMA，否则执行 DO 组中的命令。

```
IF   (&A=&B)   THEN (DO)
                 CALL   PGMA
                 CHGVAR  &A  &B        } DO 组
                 SNDPGMMSG…
                 ENDDO
CALL   PGMB
CHGVAR  &ACCTS  &B
```

DO 组也能嵌入到另外的 DO 组中，最多可嵌套 10 层，下例中有三层，注意每个 DO 组都由 ENDDO 结束。

```
PGM
 .
 .
 .
 IF  (&A=&B) DO
              CALL   PGMA
              IF  (&A=5)  DO
                          CHGVAR  &A  26
                          CALL   PGMB
                          IF  (&AREA=YES)  DO
                                           CHGVAR &AREA NO
                                           CHGVAR &P  (&P+2)
                                           ENDDO
                          CALL   ACCTSPAY
                          ENDDO
              ENDDO
CALL   PGMC
ENDPGM
```

（第一层、第二层、第三层）

此例中，如果第一层的&A≠5，调用 PGMC，如果&A=5，则执行第三个 DO 组中的语句，如果第二个 DO 组中的&AREA≠YES，则调用 ACCTSPAY，这是国为处理转到 DO 组后的下一个命令。

CL 编译程序不指出 DO 组的起始和结束，如果编译程序发现不成对的条件，它不是很容易指出实际的错误，怎样指出 DO 组的层次错误请看 QUSRTOOL 的 DSPCLPDO 中的例

子。

2.5.4　使用 ELSE 命令

ELSE 命令在 IF 命令的条件为假时规定所做的处理。

IF 命令也能不用 ELSE：

```
IF (&A=&B) THEN(CALLPRC PROCA)
CALLPRC PROCB
```

在这种情况下，仅在&A=&B 时调用 PROCA，而总是调用 PROCB。

如果要在这个过程中用 ELSE 命令，处理逻辑要修改，在下例中，如果&A=&B，则调用 PROCA，但不调用 PROCB，如果&A≠&B，则调用 PROCB。

```
IF (&A=&B) THEN(CALLPRC PROCA)
ELSE CMD(CALLPRC PROCB)
CHGVAR &C 8
```

如果 IF 表达式有一个不为真条件导致明显的转移（即绝对的非此即彼转移），则必须用 ELSE 命令。

实际上 ELSE 非常有用体现在与 DO 组一起使用上。在下例中，依据 IF 表达式的条件，可能不运行 DO 组，但总要运行其余的命令。

```
IF （&A=&B） THEN（DO）
                ·
                ·   仅当条件为真时运行
                ·
          ENDDO
CHGVAR &C  8
SAVOBJ…
CALL  PGM（PAYROLL）   不管条件是否为真，无条件运行
ENDPGM
```

如果 IF 的表达式不为真，用 ELSE 命令可以规定仅执一个命令或一组命令，则整个逻辑都改变了：

```
IF （&A=&B） THEN（DO）
                ·
                ·   仅在条件为真时做
                ·
          ENDDO
ELSE  DO
        ·
        ·   仅在条件为假时做
        ·
```

```
   ENDDO
CHGVAR  &C  8
SAVOBJ...                  } 无条件做
CALL PGM（PAYROLL）
```

每个 ELSE 前必须有一个与之相关的 IF 命令，如果 IF 有嵌套，那么每个 IF 的 ELSE 都要与之匹配。

```
IF ... THEN ...
IF ...THEN(DO)
        IF ...THEN(DO)
                  .
                  .
                  .
        ENDDO
        ELSE DO

                  IF ...THEN(DO)
                      .
                      .
                      .
                      ENDDO
                  ELSE DO
                      .
                      .
                      .
                      ENDDO
        ENDDO
ELSE IF ... THEN ...
IF ... THEN ...
IF ... THEN ...
```

要查过程中的 ELSE 是否匹配，要从最里面的一组查起。

ELSE 也能用来检查一系列互相冲突的选项。在下例中，在成功地检查完第一个 IF 后，执行嵌套的命令和 RCLRSC 命令：

```
IF  COND(&OPTION=1) THEN(CALLPRC PRC(ADDREC))
 ELSE    CMD(IF COND(&OPTION=2) THEN(CALLPRC PRC(DSPFILE)))
  ELSE   CMD(IF COND(&OPTION=3) THEN(CALLPRC PRC(PRINTFILE)))
    ELSE CMD(IF COND(&OPTION=4) THEN(CALLPRC PRC(DUMP)))
RCLRSC
RETURN
```

2.5.5 使用嵌套的 IF 命令

一个 IF 命令可以嵌入到另一个 IF 中，当在条件为真时，执行的是自身的另外一个 IF 命令（要用 THEN 参数）

```
IF (&A=&B) THEN(IF (&C=&D) THEN(GOTO END))
GOTO START
```

在执行某个或某组命令前必须满足几个条件时,这种嵌套是很有用的。在前面的例子中，如果第一个表达式为真，系统读第一个 THEN 参数，假如&C=&D，表达式为真，系统执行第二个 THEN 的命令，即 GOTO　END。二个表达式都为真，才能执行 GOTO　END。如果有一个不为真，则执行 GOTO　START。注意表达式和命令的括号。

在 CL 程序设计中可允许 10 层嵌套。

嵌套层越多，逻辑就变得越复杂。可以用自由格式使嵌套看起来更清楚：

```
PGM
DCL &A *DEC 1
DCL &B *CHAR 2
DCL &RESP *DEC 1
IF (&RESP=1) +
    IF (&A=5) +
          IF (&B=NO) THEN(DO)
                        .
                        .
                        .
                        ENDDO
CHGVAR &A VALUE(8)
CALL PGM(DAILY)
ENDPGM
```

最前边的 IF 做一个嵌套命令处理，其中如有一个为假，则处理转移到嵌套外的语句（CHGVAR 及后续的命令），只有所有的为真，才执行 DO 组，这就比在一个命令中用*AND 组合几个表达式要容易。

在某些情况下，有些转移必须是在条件为假时才发生，这时可以对每个嵌套的 IF 加一个 ELSE 语句：

```
PGM
DCL &A ...
DCL &B ...
DCL &RESP ...
IF (&RESP=1) +
      IF (&A=5) +

            IF (&B=NO) THEN(DO)
                        .
                        .
```

```
                                .
                                SNDPGMMSG ...
                                .
                                .
                                .
                              ENDDO
                    ELSE CALLPRC PROCA
          ELSE CALLPRC PROCB
    CHGVAR &A 8
    CALLPRC PROC(DAILY)
    ENDPGM
```

如果所有条件为真，则执行 SNDPGMMSG，以及后边的 CHGVAR，如果第一、第二个条件为真，第三个条件为假，则调用 PROCA。当从 PROCA 返回时，执行 CHGVAR。如果第二个条件为假，调用 PROCB，接着执行 CHGVAR，最后，如果&RESP≠1，立即执行 CHGVAR 命令。ELSE 为每个检查提供不同的转移路径。下面三个例子与前例中嵌套的 IF 是等价的：

```
IF (&RESP=1) THEN(IF (&A=5) THEN(IF (&B=NO) THEN(DO)))

IF (&RESP=1) THEN +
           (IF (&A=5) THEN +
                        (IF (&B=NO) THEN(DO)))

IF (&RESP=1) +
        (IF (&A=5) +
              (IF (&B=NO) THEN(DO)))
```

2.5.6　使用*AND、*OR、和*NOT 操作

*AND 和*OR 是逻辑操作的保留值，用来表示逻辑表达式操作之间的关系。*AND 可以用&表示，*OR 可以用¦表示，后边必须有一空格。一个逻辑表达式的操作用关系表达式或逻辑变量或常量用逻辑操作组合在一起。*AND 表示两个操作都为真时才产生真的结果，*OR 表示操作有一个为真就产生真的结果。其它非逻辑操作符，也可用在表达式中指出操作数之间的逻辑关系或表达式操作要完成的动作，除逻辑操作符外，有三类操作符：

_　算术 (+, -, *, /)

_　字符 (*CAT, ||, *BCAT, |>, *TCAT, |<)

_　关系 (*EQ, =, *GT, >, *LT, <, *GE, >=, *LE, <=, *NE, ?=, *NG,
　　?>, *NL, ?<)

下面是逻辑表达式的例子：

```
((&C *LT 1) *AND (&TIME *GT 1430))
(&C *LT 1 *AND &TIME *GT 1430)
((&C < 1) & (&TIME>1430))
((&C< 1) & (&TIME>1430))
```

逻辑表达式由三部分组成：两个操作数和一个操作符，它是由操作符的类型（*AND 和*OR）来表示表达式是逻辑型的而不是由操作数类型确定。逻辑表达式中的操作数可以是逻辑变量或其它表达式。例如：

```
((&C *LT 1) *AND (&TIME *GT 1430))
```

整个逻辑关系放在括号中，操作数都是关系表达式，也分别由括号括起。从逻辑表达式中的第二个例子看出，操作数不非得放在括号中，但放在括号中就比较清晰，不用括号是因为*AND 和*OR 有不同的优先级。即先做*AND，后做*OR。对同一优先级的操作，要用括号控制操作顺序。

在命令中可以用关系表达式做条件：

```
IF (&A=&B) THEN(DO)
              .
              .
              .
              ENDDO
```

关系表达式中操作数可以是常量。

如果要规定多个条件，可以用逻辑表达式与关系表达式一起做操作数：

```
IF ((&A=&B) *AND (&C=&D)) THEN(DO)
                          .
                          .
                          .
                          ENDDO
```

2.5.5 中的例子也可如下编码：

```
PGM
DCL &RESP *DEC 1
DCL &A *DEC 1
DCL &B *CHAR 2
IF ((&RESP=1) *AND (&A=5) *AND (&B=NO)) THEN(DO)
                                        .
                                        .
                                        .
                                        ENDDO
CHGVAR &A VALUE(8)
```

```
CALLPRC PROC(DAILY)
ENDPGM
```

这时在关系表达式中多次使用逻辑操作。

由于逻辑表达式中也可用另外的逻辑表达式做操作数。那么可以形成复杂的逻辑关系：

```
IF (((&A=&B) *OR (&A=&C)) *AND ((&C=1) *OR (&D='0'))) THEN(DO)
```

此时，&D 是逻辑变量。

任何逻辑或关系表达式的结果都为‘0’或‘1’。如果整个表达式结果为 1，则执行相关的命令，下面的命令解释了这种情况：

```
IF   ((&A = &B) *AND (&C = &D)) THEN(DO)

     ((true'1') *AND (not true'0'))
          (not true '0')
```

表达式最后结果为 0，这样，不执行 DO。

同样也可用逻辑变量来计算表达式：

```
PGM
DCL &A *LGL
DCL &B *LGL
IF (&A *OR &B) THEN(CALL PGM(PGMA))
.
.
.
ENDPGM
```

这时要计算条件表达式的值，看&A 或&B 是否有一个为 1。如果是这样，整个表达式为真，则调用 PGMA。

在逻辑变量或常量间用*OR 时，使用下列的真值表：

```
If &A is: '0' '0' '1' '1'

and &B is: '0' '1' '0' '1'



the OR expression is: '0' '1' '1' '1'
```

简单地说，对多个 OR 操作，如果所有值都为假，则结果为假，有一个为真则结果为真。

```
PGM
```

```
DCL &A *LGL VALUE('0')
DCL &B *LGL VALUE('1')
DCL &C *LGL VALUE('1')
IF (&A *OR &B *OR &C) THEN(CALL PGMA)
.
.
.
ENDPGM
```

此时值不全为假，则结果为真，调用 PGMA。

在用*AND 和逻辑变量计算逻辑表达式时，用下面的真值表：

如果 &A 是: '0' '0' '1' '1'

且 &B 是: '0' '1' '0' '1'

AND的结果是: '0' '0' '0' '1'

对多个*AND 操作，有一个值为假则结果为假，全为真时结果为真。

```
PGM
DCL &A *LGL VALUE('0')
DCL &B *LGL VALUE('1')
DCL &C *LGL VALUE('1')
IF (&A *AND &B *AND &C) THEN(CALL PGMA)
.
.
.
ENDPGM
```

此时值不全为真，则结果为假，不调用 PGMA。

在前面的这些例子中，在操作数代表一个逻辑值时，才能在表达式中使用逻辑操作数，非逻辑型变量用 OR 或 AND 是不对的，例如：

```
PGM
DCL &A *CHAR 3
DCL &B *CHAR 3
DCL &C *CHAR 3
```

不正确的是：

```
IF (&A *OR &B *OR &C = YES) THEN...
```

正确的是：

```
IF ((&A=YES) *OR (&B=YES) *OR (&C=YES)) THEN...
```

这时，在关系操作符之间是 OR。

逻辑操作符*NOT 用来取逻辑变量或常量的非。*NOT 要在*AND 或*OR 前计算，*NOT 后的操作要在任何操作数间逻辑关系前进行。

```
PGM
DCL &A *LGL '1'
DCL &B *LGL '0'
IF (&A *AND *NOT &B) THEN(CALL PGMA)
```

在此例中，所有值都为真，表达式才为真，即调用 PGMA。

```
PGM
DCL &A *LGL
DCL &B *CHAR 3 VALUE('ABC')
DCL &C *CHAR 3 VALUE('XYZ')
CHGVAR &A VALUE(&B *EQ &C)
IF (&A) THEN(CALLPRC PROCA)
```

在这个例子中，值不为真，不调用 PROCA。

2.5.7 使用%BINARY 内部函数

二进制内部函数（%BIN）把规定的 CL 字符变量的内容解释为一个二进整数。从规定的位置开始继续 2 个或 4 个字符，这个函数的语法为：

%BIN（字符变量名　　起始位　　长度）
或%BINARY（字符变量名　　起始位　　长度）

起始位和长度是可选的，如果没规定，起始位为 1，长度为用的字符变量长，字符变量的长必须是 2 或 4。

如果规定了起始位，也要规定长度为 2 或 4，起始位必须是大于或等于 1 的正数。如果起始位和长度的和大于变量的长度，则出错。（起始位也可用 CL 十进制变量）。

可与二进制函数一起使用 IF 和 CHGVAR 命令。可做算数或逻辑表达式；也能在参数为数字型的命令中使用它。

在 IF 命令中的 COND 参数用%BIN，或 CHGVAR 的 VALUE 用%BIN 时，字符变量的内容解释为二——十进制转换。

在 CHGVAR 中的 VAR 参数中使用%BIN 时，VALUE 中的十进值转换或二字节或四字节的二进整数，结果放在起始位置规定的字符变量中，小数位被截断。

用在 CALLPRC 命令的 RTNVAL 参数中的%BIN 指出调用一个过程并希望返回一个二进整数。2 字节字符变量能装下从-32768—32767 的二进整数。4 字节字符变量能装下从-2147483648—2147483647 的二进整数。

下面是%BIN 的例子：

```
DCL     VAR(&B2)  TYPE(*CHAR)  LEN(2)     VALUE(X'001C')
```

```
DCL     VAR(&N)    TYPE(*DEC)    LEN(3 0)
CHGVAR    &N    %BINARY(&B2)
```

变量&B2 的内容看作二字节的二进整数，转换成十进等值为 28。然后分配给十进变量&N。

```
DCL     VAR(&N)    TYPE(*DEC)    LEN(5 0)   VALUE(107)
DCL     VAR(&B4)   TYPE(*CHAR)   LEN(4)
CHGVAR    %BIN(&B4)     &N
```

十进变量&N 转换成 4 字节的二进数，并放在字符变量&B4 中，&B4 的值将 X‘0000006B’。

```
DCL     VAR(&P)   TYPE(*CHAR)   LEN(100)
DCL     VAR(&L)   TYPE(*DEC)    LEN(5 0)
CHGVAR    &L   VALUE(%BIN(&P 1 2) * 5)
```

变量&P 的头两个字符看作二进整数。转换成它的十进等值，再乘以 5，结果放在数字变量&L 中。

```
DCL     VAR(&X)    TYPE(*CHAR)   LEN(50)
CHGVAR    %BINARY(&X 15 2)   VALUE(122.56)
```

数 122.56 截取为整数 122，然后转换为 2 字节二进整数，放在字符变量&X 的 15 位和 16 位上，则&X 的 15 位和 16 位中包括 X‘007A’。

```
DCL     VAR(&B4)   TYPE(*CHAR)   LEN(4)
CHGVAR    %BIN(&B4)   VALUE(-57)
```

值-57 转换为 4 字节二进整数，然后分配给字符变量&B4。&B4 中将是 X‘FFFFFFC7’。

```
DCL     VAR(&B2)   TYPE(*CHAR)   LEN(2)    VALUE(X'FF1B')
DCL     VAR(&C5)   TYPE(*CHAR)   LEN(5)
CHGVAR    &C5   %BINARY(&B2)
```

&B2 的内容看作 2 字节二进整数，并转换成它的十进等量-229。这个数被转换成字符放在&C5 中，&C5 中将是‘-0029&csq’。

```
DCL     VAR(&C5)   TYPE(*CHAR)   LEN(5)    VALUE(' 1253')
DCL     VAR(&B2)   TYPE(*CHAR)   LEN(2)
CHGVAR    %BINARY(&B2)   VALUE(&C5)
```

&C5 中的字符 1253 转换成十进制数，然后转换为二字节二进整数且存在&B2 中，&B2 中将是 X‘04E5’。

```
DCL     VAR(&S)  TYPE(*CHAR)  LEN(100)
IF      (%BIN(&S 1 2) &gt 10)
        THEN( SNDPGMMSG MSG('Too many in list.') )
```

&S 的头两个字节看作是一个二进整数与 10 比较。如果它的值大于 10，则执行 SNDPGMMSG。

```
DCL     VAR(&RTNV) TYPE(*CHAR) LEN(4)
CALLPRC PRC(PROCA) RTNVAL(%BIN(&RTNV 1 4))
```

过程 PROCA 返回一个 4 字节整数存在&RTNV 中。

2.5.8　用%SUBSTRING 内部函数

字串内部函数(%SST)产生一个已有字符串的子串，它仅能用在 CL 过程中。在 CHGVAR 中，%SST 也能规定修改后变量放在哪儿（VAR 参数），或变量修改后的值(VALUE 参数)在 IF 命令中，%SST 也能在表达式中规定。

%SST 的格式为：

%SUBSTRING（字符变量名　　起始位　　长度）
　　或%SST（字符变量名　　起始位　　长度）

可用*LDA 做字符变量名指出%SST 在本地数据区中取子串。子串功能从规定的 CL 字符变量或本地数据区中取子串，子串从‘超始位’开始取（它可以是一个变量名），继续到规定的长度（也可为一个变量名）。

起始位、长度不能是 0 和负数。如果起始位和子串长的和大于整个变量的长度，则出错，本地数据区的长度为 1024。

下面是子串的例子：

如果字符变量&NAME 的头两位是 IN，则调用 INV210。&NAME 的整个值传送给 INV210。而&ERRCODE 不变。否则，它的值为 99。

```
DCL &NAME *CHAR VALUE(INVOICE)
DCL &ERRCODE *DEC (2 0)
IF (%SST(&NAME 1 2) *EQ 'IN') +
THEN(CALL INV210 &NAME)
ELSE CHGVAR &ERRCODE 99
```

如果&A 的头两位与&B 的头两位匹配，则调用 CUS210：

```
DCL &A *CHAR VALUE(ABC)
DCL &B *CHAR VALUE(DEF)
IF (%SST(&A 1 2) *EQ %SUBSTRING(&B 1 2)) +
CALL CUS210
```

位置和长度也可以是变量。下例修改变量&X 中从位置&Y 开始长度为&2 的内容为 123。

```
CHGVAR %SST(&X &Y &Z) '123'
```

在命令运行前&A 为 ABCDEFG。在运行下列命令后，&A 为 A123EFG：

```
CHGVAR %SST(&A 2 3) '123'
```

在下例中，子串的长度 5 给出与之比较的操作数 YES 的长度。操作数要填零以便能比较 YESNO 与 YESbb，条件为假。

```
DCL VAR(&NAME) TYPE(*CHAR) LEN(5) VALUE(YESNO)
.
.
.
IF (%SST (&NAME 1 5) *EQ YES) +
  THEN(CALL PROGA)
```

如果子串比操作数短，子串要填空格做比较，例如：

```
DCL VAR(&NAME) TYPE(*CHAR) LEN(5) VALUE(YESNO)
.
.
.
IF (%SST(&NAME 1 3 ) *EQ YESNO) THEN(CALL PROG)
```

由于 YESbb 不等于 YESNO，故条件为假。
下例中，变量&A 的值放到本地数据区的 1—10 位上。

```
CHGVAR %SST(*LDA 1 10) &A
```

如果本地数据区的 1—3 位与常量‘XY2’连接后等于变量&A，则调用 PROCA。假如，数据区 1—3 位为‘ABC’，&A 中为 ABCXY2，则测试结果为真，调用 PROCA。

```
IF (((%SST*LDA 1 3) *CAT 'XYZ') *EQ &A) THEN(CALLPRC PROCA)
```

这个过程扫描变量&NUMBER 且修改前导零或空格。这用在显示一个信息前加一个编辑字段：

```
      DCL &NUMBER *CHAR LEN(5)
      DCL &X *DEC LEN(3 0) VALUE(1)
      .
      .
LOOP:IF (%SST(&NUMBER &X 1) *EQ '0') DO
```

```
            CHGVAR (%SST(&NUMBER &X 1)) ' '  /* Blank out */
            CHGVAR &X (&X + 1)  /* Increment */
            IF (&X *NE 5) GOTO LOOP
            ENDDO
```

下列过程使用%SST 在 50 个字符的字段&INPUT 中查找第一次出现的至少两个字符且没有嵌入句号的句子。然后把剩余内容放在&REMAINDER 中。

```
          PGM (&INPUT &REMAINDER)   /*  SEARCH  */
          DCL &INPUT *CHAR LEN(50)
          DCL &REMAINDER *CHAR LEN(50)
          DCL &X *DEC LEN(2 0) VALUE(03)
          DCL &L *DEC LEN(2 0)   /*  REMAINING LENGTH  */
SCAN:     IF (%SST(&INPUT &X 1) *EQ '.') THEN(DO)
              CHGVAR VAR(&L) VALUE(50-&X)
              CHGVAR VAR(&X) VALUE(&X+1)
              CHGVAR VAR(&REMAINDER) VALUE(%SST(&INPUT &X &L))
              RETURN
              ENDDO
          IF (&X *EQ 49) THEN(RETURN)
          CHGVAR &X (&X+1)
          GOTO SCAN
          ENDPGM
```

过程开始检查第三位是不是句号。从&INPUT 的第三位开始长度为 1 的%SST，指的就是第三位（长度不能为零）。如果是句号，则计算剩余长度，&X 中放的是余下部分的起始位置。余下部分的内容从&INPUT 移到&REMAINDER 中。

第三位不是句号，过程检查是否到了第 49 位。如果是，则假定第 50 位是句号且返回。如果不是，过程到&X 从第 4 位开始重复这个处理。

2.5.9 使用%SWITCH 内部函数

开关内部函数%SWITCH 把 8 位中的 1 位或多位开关与已建立的 8 位作业开关相比较，返回一个逻辑值‘0’或‘1’。作业开关的初值由 CRTJOBD 命令确定，缺省值为 8 个零。如有需要，可用 SBMJOB，CHGJOB 或 JOB 命令中的 SWS 参数修改，这些缺省值是由作业描述设置的。其它高能语言也能设置作业开关。

如果%SWITCH 的值与作业开头的值比较后，每位都相同，则返回‘1’。如果有一位不匹配，则返回‘0’。

%SWITCH 的语法为：

%SWITCH（8 个屏蔽字符）。

8 个屏蔽字符指出要测试哪个作业开关以及每个开关要测试的值。屏蔽位与作业开关的 8 个位相对应，每个屏蔽位能规定为 0、1、X 三个字符之一：

0：测试相对应的作业开关是否为 0

1：测试相对应的作业开关是否为 1

X：不测试相对应的作业开关，它不影响%SWITCH 的结果。

例如，如果规定了%SWITCH（0X111XX0），那么测试作业开关 1 和 8 是否为 0。作业开关 3、4、5 是否为 1，不测 2、6、7 位。如果作业开关的每位都与屏蔽位相同，则结果为 1。

作业开关可用在 CL 过程中控制流程，可以与 IF、CHGVAR 命令一起使用。可用 CHGJOB 命令在 CL 过程中修改开关。对 CL 过程这些修改立即有效。

2.5.9.1 用%SWITCH 的命令

在 IF 命令中，%SWITCH 可在 COND 参数中规定做为要测试的逻辑表达式，在下例中，0X111XX0 与作业开关相比较：

```
IF  COND(%SWITCH(0X111XX0))  THEN(GOTO C)
```

如果作业开关的 1、3、4、5、8 依次是 0、1、1、1 和 0，则结果为真，过程转移到标号为 C 处，如果作业开关有一位不符，则结果为假，不做转移操作。

在下面的例子中，开关控制二个过程的处理：

```
SBMJOB  JOB(APP502) JOBD(PAYROLL) CMD(CALL APP502)
    SWS(11000000)

PGM  /* CONTROL */
IF (%SWITCH(11XXXXXX)) CALLPRC PROCA
IF (%SWITCH(10XXXXXX)) CALLPRC PROCB
IF (%SWITCH(01XXXXXX)) CALLPRC PROCC
IF (%SWITCH(00XXXXXX)) CALLPRC PROCD
ENDPGM

PGM  /* PROCA */
CALLPRC TRANS
IF (%SWITCH(1XXXXXXX)) CALLPRC CUS520
ELSE CALLPRC CUS521
ENDPGM
```

2.5.9.2 用%SWITCH 的 CHGVAR 命令

在 CHGVAR 命令中，可用%SWITCH 来修改逻辑变量的值。逻辑变量的值是由%SWITCH 与作业开关相比较确定的。如果结果为真，则变量值为‘1’，否则为‘0’，下例中，作业开关为 10000001。&A 的结果值为‘1’。

```
PGM
DCL &A *LGL
CHGVAR VAR(&A) VALUE(%SWITCH(10000001))
.
.
```

```
.
ENDPGM
```

2.5.10 使用监控信息命令 MONMSG

逃逸信息是由 CL 过程中的命令或它们调用的程序或过程传送给 CL 过程的。信息告诉过程已检查到错误且没有完成功能。CL 过程也能监控逃逸信息的到达，能用命令来处理这些信息。例如，如果一个 CL 过程试图往已删除的数据区移动数据，那么会由 MOVOBJ 命令发出‘目标没找到’的逃逸信息。

用 MONMSG 命令，要控制程序在有错误发生时立即采取什么样的动作，MONMSG 可以监视逃逸、通知及状态信息，把它送到使用 MONMSG 命令的调用堆栈中。MONMSG 有下列参数：

```
MONMSG   MSGID(信息标识)  CMPDTA(比较数据)  EXEC(CL 命令)
```

对每个错误发送的信息都有唯一的标识。在 MSGID 参数中最多可有 50 个标识，CMPDTA 参数用来检查在信息的 MSGDTA 部分的特别字符串，用 EXEC 参数规定做错误恢复的 CL 命令。（例如，CALL，DO 或 GOTO）

在下例中，MONMSG 命令是跟在 RCVF 后，因此，它仅监控由 RCVF 发送的信息。

```
READLOOP:  RCVF                          /* Read a file record  */
           MONMSG  MSGID(CPF0864)  EXEC(GOTO CMDLBL(EOF))
           /*  Process the file record  */
           GOTO  CMDLBL(READLOOP)       /*  Get another record  */
EOF:       /*  End of file processing  */
```

在文件中没有可读记录时，CPF0864 送到过程相关的队列中，由于规定了 MSGID（CPF0864），MONMSG 监控这个条件，当收到该信息后，运行 GOTO 命令。

也可用 MONMSG 命令来监控由 CL 过程中命令发送的信息。在下例中，有两个 MONMSG 命令，第一个监控信息 CPF001 和 CPF1999。这些信息可能是过程中其后运行的任一命令发出的。不论收到哪个信息，控制都转移到由标号 EXIT2 标识的命令处。第二个 MONMSG 监控 CPF2105 和 MCH1211。由于没有 EXEC 参数，则忽略此信息。

```
PGM
DCL
MONMSG MSGID(CPF0001 CPF1999) EXEC(GOTO EXIT2)
MONMSG MSGID(CPF2105 MCH1211)
.
.
.
ENDPGM
```

CPF0001 标识信息本身指定的错误，大部分的调试命令发送 CPF1999 信息，它指出在命令中发现错误，但它不标识信息中的命令。

由 MONMSG 监控的错误条件都用 EXEC 参数规定的方式来处理，此例中是 EXIT2，

它也不可能返回出错后下一个顺序语句。要避免发生这种情况，可在每个命令之后监控条件，规定错误恢复例程。

所有没有规定 EXEC 的监控到的信息都被忽略，过程继续执行下个命令。

如果在计算 IF 命令的表达式时发生错误，条件做为假处理。在下例中，在 IF 中可能发生 MCH1211 错误，则条件为假，调用用 PROCA。

```
IF(&A / &B *EQ 5) THEN(DLTF ABC)
ELSE CALLPRC PROCA
```

如果在 CL 过程的开始就用 MONMSG 命令，那么它监控整个过程中的规定信息，而不考虑是哪个命令产生的信息。如果用 EXEC 参数，那么只能规定 GOTO 命令。

在过程级或命令级可规定相同的信息标识，命令级的 MONMSG 要复盖过程级的。在下例中，如果在 CMDB 中收到信息 CPF0001，则运行 CMDC。如果在过程中的其它命令上收到 CPF0001，过程转到 EXIT2。如果在任何命令上接收到 DCPF1999，转移到 EXIT2。

```
PGM
MONMSG MSGID(CPF0001 CPF1999) EXEC(GOTO EXIT2)
CMDA
CMDB
MONMSG MSGID(CPF0001) EXEC(CMDC)
CMDD
EXIT2:  ENDPGM
```

由于有很多逃逸信息会发送给过程，所以必须考虑你想监控处理哪一个。大多数信息在过程中有错时才发送，而另一些是由于条件超出过程外。一般来说，CL 过程监控这些信息基本功能，且做适当的处理。对所有其它信息，&CPF 假定发生了错误且采取了适当的缺省动作。处理信息的详细内容，请看第七章和第八章。

2.6 能做变量的值

2.6.1 取系统值

系统值包括控制操作系统某部分的信息。IBM 提供了几类系统值，例如 QDATE 和 QTIME 是系统日期和时间，它是在&CPF 起动时设置的。可用 RTVSYSVAL 命令在 CL 过程中把系统值做为变量来对待：

```
RTVSYSVAL     SYSVAL(系统值名)     RTNVAR(CL 变量名)
```

RTNVAR 参数规定接收系统值的过程中变量的名，变量的类型必须与系统值类型相匹配。对于字符和逻辑系统值，CL 变量长度必须等于值的长度，对十进制变量的长度必须大于或等于系统值的长度。

2.6.1.1 系统值 QTIME

在下例中，接收 QTIME 并把它放在一个变量中，然后与另外变量比较。

```
PGM
DCL  VAR(&PWRDNTME)  TYPE(*CHAR)  LEN(6)  VALUE('162500')
DCL  VAR(&TIME)  TYPE(*CHAR)  LEN(6)
RTVSYSVAL  SYSVAL(QTIME)  RTNVAR(&TIME)
IF  (&TIME *GT &PWRDNTME)  THEN(DO)
SNDBRKMSG('Powering down in 5 minutes.  Please sign off.')
PWRDWNSYS  OPTION(*CNTRLD)  DELAY(300)  RESTART(*NO)  +
          IPLSRC(*PANEL)

ENDDO
ENDPGM
```

2.6.1.2　系统值 QDATE

在很多应用中，都用当前的系统日期。这可用接收系统值 QDATE 把它放在一个变量中实现，也可能在应用过程中修改日期的格式，要转换格式，用 CVTDTA 命令。

系统日期的格式由系统值 QDATFMT 决定,初值为 MDY(月日年)。可以把它改为 YMD、DMY 或 JUL 格式。JUL 格式中，QDAY 的值为 001—366 的三位字符，它用来确定两个日期之间的天数，可用 CVTDAT 命令来删除日期分隔符的字符，CVDDAT 的格式为：

```
CVTDAT     DATE(要转换的日期)     TOVAR(CL 变量)  +
           FROMFMT(旧格式)     TOFMT(新格式)     TOSEP(新分隔符)
```

DATE 参数可以是常量或变量，日期一经转换好后，就放在 TOVAR 规定的变量中。在下例中，日期在&DATE 中，它的格式为 MDY，要转换为 DMY 格式放在变量&CVTDAT 中：

```
CVTDAT     DATE(&DATE) TOVAR(&CVTDAT) FROMFMT(*MDY) TOFMT(*DMY)
           TOSEP(*SYSVAL)
```

日期分隔符使用系统值 QDATSEP 中规定的符号。

在生成或增加名字中有日期的目标或成员时，CVTDAT 是很有用的。例如，假定成员要用当前的系统日期为名加到文件中，当前日期格式为 MDY，要转换成 JUL 格式：

```
PGM
DCL &DATE6 *CHAR  LEN(6)
DCL &DATE5  *CHAR  LEN(5)
RTVSYSVAL  QDATE  RTNVAR(&DATE6)
CVTDAT  DATE(&DATE6)  TOVAR(&DATE5)  TOFMT(*JUL)  TOSEP(*NONE)
ADDPFM  LIB1/FILEX  MBR('MBR'  *CAT  &DATE5)
.
.
.
ENDPGM
```

假如当前日期为 1988 年 1 月 5 日，则增加的成员名为 MBR88005。

在转换日期时，要注意以下几点：

在 DATE 参数中值的长度和 TOVAR 中变量的长度要与日期格式相匹配。TOVAR 中变量长度至少是：

1. 对非 JUL 日期处理 2 位年：
   a、在不用分隔符时，用 6 位字符。1978 的 7 月 28 日写为 072878。
   b、在用分隔符时，用 8 位字符。1978 年 7 月 28 日写为 07-28-78。
2. 对 4 位年的非 JUL 日期：
   a、不用分隔符时，用 8 位字符。1978 年 7 月 28 日写为 07281978。
   b、在用分隔符时，用 10 位字符。1978 年 7 月 28 日写为 07-28-1978。
3. 对 2 位年的 JUL 日期：
   a、不用分隔符时，用 5 位字符。1996 年 12 月 31 日写为 96365。
   b、在用分隔符时，用 6 位字符。1996 年 12 月 31 日写为 96-365。
4. 对 4 位年的 JUL 日期：
   a、不用分隔符时，要 7 位字符。1997 年 2 月 4 日写为 1997035。
   b、用分隔符时，要 8 位字符。1997 年 2 月 4 日写为 1997-035。

如果转换的字符在变量中放不下，要有错误信息。如果比变量短，要在右边添空格。

除 JUL 格式外，年、月、日都是 2 字节字段，转换后的值都是右对齐，必要时添前导零。用 JUL 格式，日为 3 字节、年为 2 字节 ，转换后的值为右对齐，如需要，可加前导零。

下例是使用 ILE 连编 API 为转换成 JUL 格式的取当前本地时间。要生成程序，须用 CRTBNDCL 命令，或用 CRTCLMOD，再用 CRTPGM 命令。

```
PGM
DCL  &LILDATE  *CHAR  LEN(4)
DCL  &PICTSTR  *CHAR  LEN(5)  VALUE(YYDDD)
DCL  &JULDATE  *CHAR  LEN(5)
DCL  &SECONDS  *CHAR  8       /*  从CEELOCT取得秒数          */
DCL  &GREG     *CHAR  23      /*  从CEELOCT取得阳历          */
                              /*                             */
CALLPRC  PRC(CEELOCT)         /*  取得当前日期和时间         */ +
         PARMS (&LILDATE)     /*  Lilian 格式的日期          */ +
                &SECONDS      /*  不用的秒字段               */
                &GREG         /*  不用的阳历字段             */
                *OMIT         /*  省略反馈参数               */

CALLPRC  PRC(CEEDATE) +
         PARMS (&LILDATE)     /*  今天的日期                 */ +
                &PICTSTR      /*  如何格式化                 */ +
                &JULDATE      /*  儒略日期                   */ +
                *OMIT

ADDPGM   LIB1/FILEX  MBR('MBR'  *CAT   &JULDATE')

ENDPGM
```

2.6.2 取配置源成员

用 RTVCFGSRC 命令，可把已有的配置目标用来建立 CL 命令源码，把源码放在一个源文件成员中。生成的 CL 源码可用于下列方面：

把配置在系统间交换
管理现场配置
备份配置（不用 SAVSYS）

2.6.3 取配置状态

用 RTVCFGSTS 命令，可从三个配置目标中为应用程序取配置状态。三个配置目标为线路，控制器和设备。此命令用在 CL 过程中检查配置描述的状态。

2.6.4 取网络属性

用 RTVNET 命令可取得系统的网络属性。这些属性可用 CHGNETA 修改，用 DSPNETA 来显示。

下面是 RTVNETA 的例子：

```
PGM
DCL  VAR(&OUTQNAME) TYPE(*CHAR) LEN(10)
DCL  VAR(&OUTQLIB) TYPE(*CHAR) LEN(10)
RTVNETA  OUTQ(&OUTQNAME) OUTQLIB(&OUTQLIB)
CHGNETA  OUTQ(QGPL/QPRINT)
.
.
.
CHGNETA  OUTQ(&OUTQLIB/&OUTQNAME)
ENDPGM
```

2.6.5 取作业属性

可取作业属性把它放在一个变量中来控制应用程序。所用的命令为 RTVJOBA，可取得所有作业属性或它们的组合。

在下例中，RTVJOBA 取得调用此过程的用户名字：

```
        PGM
         /* ORD410C Order entry program  */
        DCL &CLKNAM TYPE(*CHAR) LEN(10)
        DCL &NXTPGM TYPE(*CHAR) LEN(3)
        .
        .
        .
        RTVJOBA USER(&CLKNAM)
BEGIN:  CALL ORD410S2 PARM(&NXTPGM &CLKNAM)
         /* Customer prompt */
        IF (&NXTPGM *EQ 'END') THEN(RETURN)
```

.
.

变量&CLKNAM 是传来的用户名，当调用 ORD410S2 时，要有两个变量&NXTPGM 和&CLKNAM 传给它，&NXTPGM 传的是空格但可由 ORD410S2 来修改。

下例中，包括这个过程的程序由交互作业提交给批处理。用 RTVJOBA 命令取作业所用的信息队列名，用它来与提交作业的用户通信。

```
PGM
DCL &MSGQ *CHAR 10
DCL &MSGQLIB *CHAR 10
DCL &MSGKEY *CHAR 4
DCL &REPLY *CHAR 1
DCL &ACCTNO *CHAR 6
.
.
.
RTVJOBA SBMMSGQ(&MSGQ) SBMMSGQLIB(&MSGQLIB)
IF (&MSGQ *EQ '*NONE') THEN(DO)
   CHGVAR &MSGQ 'QSYSOPR'
   CHGVAR &MSGQLIB 'QSYS'
ENDDO
.
.
.
IF (. . . ) THEN(DO)
 SNDMSG:SNDPGMMSG MSG('Account number ' *CAT &ACCTNO *CAT 'is +
                  not valid. Do you want to cancel the update +
                  (Y or N)?') TOMSGQ(&MSGQLIB/&MSGQ) MSGTYPE(*INQ) +
                  KEYVAR(&MSGKEY)
 RCVMSG MSGQ(*PGMQ) MSGTYPE(*RPY) MSGKEY(&MSGKEY) +
          MSG(&REPLY) WAIT(*MAX)
 IF (&REPLY *EQ 'Y') THEN(RETURN)
 ELSE IF (&REPLY *NE 'N') THEN(GOTO SNDMSG)
ENDDO
.
.
.
```

这里说明了两个变量，&MSGQ 和&MSGQLIB，用来接收信息队列名及其库名。由于作业可能没规定信息队列名，那么把信息队列名与*NONE 比较。如果相等，则表示没有信息队列，这时修改变量用 QSYSOPR 做信息队列名，用 QSYS 做库名。在检查到有错时，往信息队列送请求信息，并接收应答和处理。

在下列方面也可用 RTVJOBA 命令：

取得一个或多个作业属性，这样可做临时修改而后再改为原来的值。

取得一个或多个作业属性，用在 SBMJOB 命令中，这样使已提交和要提交的作业属性相同。

2.6.6 取目标描述

可用 RTVOBJD 命令取某个目标的描述返回给 CL 过程，用变量来返回描述，也可用这些描述帮助你确定没有用的目标，也可用 QUSRJOBD API 来返回目标描述，也是用变量来返回的。

2.6.7 取用户配置文件属性

用 RTVUSRPRF 命令来取得用户配置文件的属性，把它们放在 CL 变量中来控制应用程序，可在命令中规定 10 个字符的用户配置文件名或用*CURRENT。在运行 RTVUSRPRF 后可监控逃逸信息。

在下面的 CL 过程中，用 RTVUSRPRF 取得调用 CL 过程的用户名和此用户的信息队列名：

```
DCL  &USR  *CHAR  10
DCL  &USRMSGQ  *CHAR  10
DCL  &USRMSGQLIB  *CHAR  10
.
.
.
RTVUSRPRF  USRPRF(*CURRENT)  RTNUSRPRF(&USR)  +
           MGSQ(&USRMSGQ)  MSGQLIB(&USRMSGQLIB)
```

下列信息会返回给过程：

&USR 中放的是调用这个程序的用户名

&USRMSGQ 中放的是用户配置文件中规定的信息队列名

&USRMSGQLIB 中放的是用户配置文件中规定的信息队列所在的库名

2.6.8 取成员描述信息

用 RTVMBRD 命令取数据库文件成员的信息，用在应用程序中。在下面的例子中，用 RTVMBRD 命令取规定成员的描述。假定数据库文件名为 MFILE，放在 MYLIB 中，其中有三个成员（AMEMBER，BMBMBER 和 CMEMBER）。

```
DCL  &LIB      TYPE(*CHAR) LEN(10)
DCL  &MBR      TYPE(*CHAR) LEN(10)
DCL  &SYS      TYPE(*CHAR) LEN(4)
DCL  &MTYPE    TYPE(*CHAR) LEN(5)
DCL  &CRTDATE  TYPE(*CHAR) LEN(13)
DCL  &CHGDATE  TYPE(*CHAR) LEN(13)
DCL  &TEXT     TYPE(*CHAR) LEN(50)
DCL  &NBRRCD   TYPE(*DEC)  LEN(10 0)
DCL  &SIZE     TYPE(*DEC)  LEN(10 0)
```

```
DCL  &USEDATE  TYPE(*CHAR) LEN(13)
DCL  &USECNT   TYPE(*DEC)  LEN(5 0)
DCL  &RESET    TYPE(*CHAR) LEN(13)
.
.
.
RTVMBRD   FILE(*CURLIB/MYFILE) MBR(AMEMBER *NEXT) +RTNLIB(&LIB) RTNSYSTEM(&SYS) RTNMBR(&MBR) +
            FILEATR(&MTYPE) CRTDATE(&CRTDATE) TEXT(&TEXT) +
            NBRCURRCD(&NBRRCD) DTASPCSIZ(&SIZE) USEDATE(&USEDATE) +
            USECOUNT(&USECNT) RESETDATE(&RESET)
```

能返回下列信息：

当前库名放到&LIB 中

找到 MYFILE 的系统放到&SYS 中（*LCL 表示在本地系统找到文件，*RMT 表示在远程系统中找到文件）

BMEMBER 放到&MBR 中，因为它在成员名列表中排在 AMEMBER 后面（*NEXT）

文件属性放到&MTYPE 中（*DATA 表数据成员，*SRC 表源成员）

BMEMBER 的生成日期放到&CRTDATE 中

BMEMBER 的说明放到&TEXT 中

BMEMBER 的当前记录数放到&MBRRCD 中

BMEMBER 的数据空间大小放到&SIZE 中

BMEMBER 最后中使用的日期放到&USEDATE 中

BMEMBER 已经使用的天数放到&USECNT 中，它的起始日期放到&RESET 中

有关使用 RTVxxx 命令的更多例子，可在 QUSRTOOL 中找到。

2.7 处理 CL 过程

一个 CL 源过程编译成模块，然后连编成程序才能运行，一步生成 CL 程序，可用 CRTBNDCL 命令与模块生成连编程序，也可用 CRTCLMOD 先生成模块，然后用 CRTPGM 或 CRTSRVPGM 命令连编成程序或服务程序。

下面例子用 CRTCLMOD 命令生成模块 ORD040C 并把它放到 DSTPROLIB 中。

```
CRTCLMOD      MODULE(DSTPRODLB/ORD040C) SRCFILE(QCLSRC)
              TEXT('Order dept general menu program')
```

ORD040C 的源命令放在 QCLSRC 文件中，源成员名叫做 ORD040C。在 CRTBNDCL 命令中，可以规定清单选项和是否在程序主人的用户配置文件下操作。一个程序可能在主人的用户配置文件下运行，也可在其它用户下运行。

可用 PDM 菜单中的选项或程序员菜单来生成 CL 过程和程序，这样可不用直接写 CRTxxx 命令。

2.7.1

在用 CRTCLMOD 或 CRTBNDCL 命令编译过程时，可在 LOG 参数中规定下列值之一来把过程中运行的大多数 CL 命令写到作业日志中。

*JOB，这是缺省值，它指出当规定作业日志选项为 ON 时要写日志，这个选项初值为不做日志。但可用 CHGJOB 命令中的 LOGCLPGM 修改，这样，当

用*JOB 值生成程序或模块时，能修改每个作业的日志选项。

*YES，规定只要运行 CL 过程就写日志，它不能用 CHGJOB 命令来修改。

*NO，规定不做日志，它不能用 CHGJOB 命令修改。

由于这些值是 CRTCLMOD 和 CRTBNDCL 命令的一部分，那么如修改它们必须重新编译。

在规定做日志时，要小心使用 RMVMSG 命令，不要从作业日志中移出任何日志命令。如果在 RMVMSG 命令中规定 CLEAR(*ALL)，那么所有在 RMVMSG 命令前运行命令的日志都从作业日志中清除掉，这个仅影响有 RMVMSG 的过程而不影响前后发生的已日志过的命令。

不是所有的命令都记到日志中，下面是不日志的命令：

CHGVAR	DO	GOTO
DCL	ELSE	IF
DCLF	ENDDO	MONMSG
PGM	ENDPGM	CALLPRC

如果日志选项为 ON，则日志的信息送到过程的信息队列中。如果是交互地运行过程，而且作业的 LOG 参数中信息级别为 4，则可用 F10 键来看所有命令的日志。如果在注销时规定*PRINT，则可打印日志。

日志包括时间、程序或过程名、信息正文和命令名。命令名是由源语句中限定的，也日志命令的参数。如果参数是变量，则打印变量的内容。

命令的日志会影响性能。

2.7.2 CL 模块编译清单

在 CRTCLMOD 时，可用 OPTION 和 OUTPUT 参数来生成几类清单。

OPTION 可用的值及意义为：

*GEN 和*NOGEN，是否生成模块（*GEN 为缺省值）

*XREF 或*NOXREF，是否产生源输入的变量的交叉引用表或数据引用清单。

OUTPUT 可用的值及意义为：

*PRINT——打印清单

*NONE——没有编译清单

规定了 OUTPUT 参数生成清单叫做编译清单，下面是清单的例子：

```
      1
                                  2
          3
5763SS1 V3R1M0  940909              Control Language                MYLIB/DUMPERR        05/06/94 11:12:55  Page 1
Program . . . . . . . . . . . . . . . . . . . :   DUMPERR
  Library . . . . . . . . . . . . . . . . . . :     MYLIB
Source file . . . . . . . . . . . . . . . . . :   QCLSRC
  Library . . . . . . . . . . . . . . . . . . :     MYLIB
Source member name  . . . . . . . . . . . . . :   DUMPERR   05/06/94 10:42:26  4
Source printing options . . . . . . . . . . . :   *XREF  *NOSECLVL  *NOEVENTF
User profile  . . . . . . . . . . . . . . . . :   *USER
```

```
 Program logging . . . . . :   *JOB
 Default activation group  . . . . . :   *YES
 Replace program . . . . . :   *YES
 Target release  . . . . . :   V3R1M0
 Authority . . . . . :   *LIBCRTAUT
 Sort sequence . . . . . :   *HEX
 Language identifier . . . . . :   *JOBRUN
 Text  . . . . . :   Test program
 Optimization  . . . . . :   *NONE
 Debugging view  . . . . . :   *STMT
 Compiler  . . . . . :   IBM AS/400 Control     Language Compiler  5
     6                         Control Language Source
 SEQNBR  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+.  DATE   8
    100- PGM                                                                                                 05/06/94
    200- DCL &ABC *CHAR 10 VALUE('THIS')                                                                     05/06/94
    300- DCL &XYZ *CHAR 10 VALUE('THAT')  7                                                                  05/06/94
    400- DCL &MNO *CHAR 10 VALUE('OTHER')                                                                    05/06/94
    500- CRTLIB LB(LARRY)                                                                                    05/06/94
* CPD0043 30  Keyword LB not valid for this command.  9
    600- DLTLIB LIB(MOE                                                                                      05/06/94
* CPD0013 30  A matching parenthesis not found.
    700- MONMSG CPF0000 EXEC(GOTO ERR)                                                                       05/06/94
    800- ERROR:                                                                                              05/06/94
    900- CHGVAR &ABC 'ONE'                                                                                   05/06/94
   1000- CHGVAR &XYZ 'TWO'                                                                                   05/06/94
   1100- CHGVAR &MNO 'THREE'                                                                                 05/06/94
   1200- DMPCLPGM                                                                                            05/06/94
   1300- ENDPGM                                                                                              05/06/94
                              * * * * *  E N D  O F  S O U R C E  * * * * *
 5763SS1 V3R1M0  940909                Control Language                MYLIB/DUMPERR        05/06/94 11:12:55  Page 2
                                        Cross Reference
 Declared Variables
 Name             Defined     Type            Length      References
 &ABC                200      *CHAR             10           900
 &MNO                400      *CHAR             10          1100
 10
 &XYZ                300      *CHAR             10          1000
 Defined Labels
 Label            Defined     References  11
 ERR              ******         700
* CPD0715 30  Label 'ERR       ' does not exist.
 ERROR               800
                     * * * * *  E N D  O F  C R O S S  R E F E R E N C E  * * * * *
 5763SS1 V3R1M0  940909                Control Language                MYLIB/DUMPERR        05/06/94 11:12:55  Page 3
```

```
                                Message Summary

             Severity

Total        0-9  10-19  20-29  30-39  40-49  50-59  60-69  70-79  80-89  90-99   12

    3          0      0      0      3      0      0      0      0      0      0

Program DUMPERR not created in library MYLIB. Maximum error severity 30.  13

                        * * * * *  E N D  O F  M E S S A G E  S U M M A R Y  * * * * *
```

下面是对图中标出号码处的解释：

标题：

1. 程序号码，版本，修改级别及&CPF 的日期。
2. 编译的日期和时间。
3. 清单页号。

序：

4. 在 CRTCLMOD 中规定的参数值。如果源码不在数据库文件中，省去成员名，日期和时间。
5. 编译程序名称。

源码：

6. 源码中的行顺序号，顺序号后的‘一’指出源语句开始的顺序号，没有‘一’的语句是前面语句的继续。

   源语句之间的注释与其它源语句一样处理且有顺序号。
7. 源语句。
8. 源语句修改或增加的最后日期，如果源语句不是在数据库文件中，则日期用 RGZPFM 重设且省略它。
9. 如果在编译期间发现有错，且能跟踪到某个语句，那么在这个语句之后立即打印错误信息。*指出这行有错，这行中给出信息标识、级别、和信息说明。详细内容请看 2.7.3。

交叉引用：

10. 符号变量表是在程序中说明过的变量交叉引用列表。此表列出变量、说明变量的源语句顺序号，变量属性及引用此变量的语句顺序号。
11. 标号表是程序中定义的有效标号列表。此表列出标号，定义此标号的语句顺序号及引用此标号的语句顺序号。

信息：

此例中没有包括这段。这是因为它没有发出一般信息。如果有一般信息发出，这段对每个信息有一个给出信息标识、级别的说明。

信息总结：

12. 在编译期间，给出信息总结，以级别分类给出。
13. 在信息总结后给出完成信息。

标题、序号、源码和信息总结段的内容在有*SOURCE 选项时都打印出来。如果规定*XREF 才打印交叉引用信息，如果有错误时才打印信息段内容。

用 CL 源成员中的 CL 命令编译程序编译它产生结果，需要一个中间系统码来完成程序运行时所需要的功能。

2.7.3 在编译期间遇到的错误

在模块的编译清单中，一个与命令有关的错误条件直接跟在此命令之后给出，不直接和某些命令有关的更一般的错误在信息段给出，它不放在源语句中间。

编译期间检查到的错误包括语法错误，引用一个没定义的变量或标号，语句错。下列错误会停止生成模块：

值错误

语法错误

命令中参数之间的关系错误

有效性检查错误

遇到以上各种错误，编译程序继续检查源语句。程序员可以看编译清单改正多处错误，再次编译生成模块。

2.7.4 获得过程转储

在处理过程期间可得到转储结果。转储由过程信息队列的所有信息以及所有变量说明组成，可用它来解决影响过程执行出现的问题。要得到转储，做下列之一：

运行 DMPCLPGM 命令，这个命令仅用在 CL 过程中，不能结束过程。

对 CPA0702 信息回答 D，这个信息是在有没监控的逃逸信息时发出的。如果程序在交互下运行，信息送往作业的外部信息队列。如果程序是做为批作业运行，信息送往操作员信息队列。

对作业规定 INQMSGRPG(*SYSRPYL)，它要求回答信息 CPA0702 或 CPA0701。在送出回答后会打印转储。

把对信息 CPA0702 信息回答的缺省值 C 改为 D，当发生功能检查时会打印过程转储，要改缺省值，用下列命令：

```
CHGMSGD MSGID(CPA0702) MSGF(QCPFMSG) DFT(D)
```

注：此命令是有权限限制的。

在下列条件之一下修改信息缺省值能打印转储：

系统操作员信息队列是在缺省方式且信息是从批作业发出的，工作站用户没给出回答按执行键，则用信息缺省值。

对作业规定 INQMSGRPY（*DFT）。

```
                                                                                1
5763SS1 V3R1M0 940909                          CL Program Dump                 5/24/94  11:05:03 2  Page 1


Job name . . . . . :  DSP04 3   User name . . . . . :   SMITH  3   Job number  . . . . . : 01329  3
Program name . . . . . :  DUMP 4    Library . . . . . :   MYLIB 4    Statement . . . . . : 1200  5
Module name  . . . . . :  DUMP      Procedure name  . . . :   DUMP


                                                  Messages

           Message  6                Message                    From                    To
Time       ID              Sev       Type    Text               Program         Inst    Program         Inst
110503     CPC2102         00        COMP    Library LARRY created.  QLICRLIB      *N      DUMP            *N
110503     CPF2110         40        ESC     Library MOE not found.  QLICLLIB      *N      DUMP            *N
```

```
                                    Variables  7

Variable        Type       Length          Value                      Value in Hexadecimal

                                           *...+....1....+....2....+   *...+....1....+....2....+
&ABC            *CHAR              10      'ONE       '               D6D5C5404040404040
&XYZ            *CHAR              10      'TWO       '               E3E6D6404040404040

                          *****  E N D  O F  D U M P  *****
```

1. 程序号，版本修改级别和&CPF 日期。
2. 打印转储的日期和时间。
3. 过程运行的作业全名。
4. 程序名及库名。
5. 产生转储时的语句号。如果命令是嵌套的，语句号是外部命令的语句号。
6. 调用信息队列的每个信息。包括信息发送的时间、信息标识、级别、类型、说明、发送的程序和指令号码，以及接收的程序和指令号码。
7. 程序中说明的所有变量。包括变量名、类型、长度、值和十六进制值。如果十进变量中有无效十进数，则做为*CHAR 变量打印字符或十六进制值。

如果没分配变量的值，则打印*NOT ADDRESSABLE。如果程序中有 TYPE(*NULL)或 PASSVAL(*NULL)参数的命令时，或规定了 RTNVAL(*YES)，但没给出返回的变量，则会发生这种情况。

如果变量是 TYPE(*LGL)，在转储中它做为 1 位长的*CHAR 给出。

2.7.5 显示模块属性

用 DSPMOD 可显示模块属性，这个信息可用来确定生成模块所用命令的选项。

2.7.6 显示程序属性

可用 DSPPGM 命令来显示一个程序的属性，它可用来确定生成程序所用命令的选项。

2.7.7 返回码总结

在 RTVJOBA 命令中的RTNCDE 参数是五位无小数的十进制值，它指出被调用程序的状态。

下表给出&SYS 支持的语言所用返回码：

RPG IV 程序：

由 RPG IV 编译程序发出的返回码是：

0 程序生成了

1 程序没生成

由运行 RPG IV 程序发出的返回码是：

0 程序起动或由 CALL 调用程序

1 程序以 LR 为 ON 结束

2 程序因错误结束

3 程序因指示器失败结束

在 CALL 后检查出的 RPG IV 返回码：

0 或 1　无错

3　给出 RPG　IV 状态码 231

其它值　给出 RPG　IV 状态码 202（CALL 因错误结束）

返回码不能相接用用户测试。

CL 程序：

返回码的当前值可用 RTVJOBA 中的参数 RTNCDE 得到。

2.8　编译以前版本的源程序

可用 CRTCLPGM 中的 TGTRLS 参数来编译以前版本写的源程序。TGTRLS 规定试图运行在哪个版本下生成的 CL 目标程序，可以规定*CURRENT，*PRV 或某一个版本号。

用 TGTRLS(*CURRENT)编译的程序仅能在当前版本或以后版本运行，用其它的值编译的程序能在规定版本及以后版本下运行。

2.8.1　以前版本（*PRV）库

CL 编译程序从 CL 以前版本库中取得以前版本的命令和文件信息。有二类库包含这种支持：系统库和用户库。各字分别为 QSYSVxRxMx 和 QUSRVxRxMx。例如，QUSRV2R2M0 就表示是支持版本 2.20 系统。

在 CL 编译程序编译支持以前版本的程序时，首先要检查在以前版本库中的命令和文件，如果查找失败，则查找库列表和以下的限定库：

QSYSVxRxMx 库：当 CL 编译程序支持以前版本时就安装了这些库，库中包括在 QSYS 库中找到的为某个版本的目标定义的命令和输出文件。

QUSRVxRxMx 库：可以自己生成这个库来放支持以前版本的命令和文件的复本。如果在当前版本下修改了命令或文件，这是特别重要的。

当编译程序查找以前版本命令和文件时，先查 QUSRVxRxMx，再查 QSYSVxRxMx。

注：用 QUSRVxRxMx 来放以前版本的命令和文件，可不用 QSYSVxRxMx。在安装以后版本的 CL 编译程序时，也安装支持后来版本的 CL 编译程序，用时也安装支持以前版本的，可以修改或取消 QSYSVxRxMx 库。

不要把以前版本的库加到库列表中，它们包括不能在当前系统下运行的命令或文件。仅仅 CL 编译程序可以使用和引用以前版本库中的命令和文件，支持以前版本的系统命令是在系统的主语言中，没有第二语言可用。

对如何保存不同版本下的目标，请看相应的 CL 参考手册。

2.8.1　安装支持以前版本的 CL 编译程序

要安装*PRV　CL 编译程序和 QSYSVxRxMx 库：

1. 在命令行上输入 GO　LICPGM，出现特许程序菜单。
2. 选 11。
3. 选择 5769SS1—*PRV　CL　COMPILER　SUPPORT。它安装 QSYSVxRxMx 库。

如果不用这个库，可用下列命令去掉：

```
DLTLICPGM　LICPGM(5769SS1)　OPTION(9)
```

移出后，QSYSVxRxMx 不在系统中了，但 QUSRVxRxMx 还在，如果也不用它，要用 DLTLIB 命令删除它。

第三章　程序和过程间的流程控制及通讯

可用 CALL、CALLPRC 和 RETURN 命令来在程序和过程之间传递控制，每个命令都有不同的特点。在传递控制时会把要传递的信息做为参数传给被调用程序。

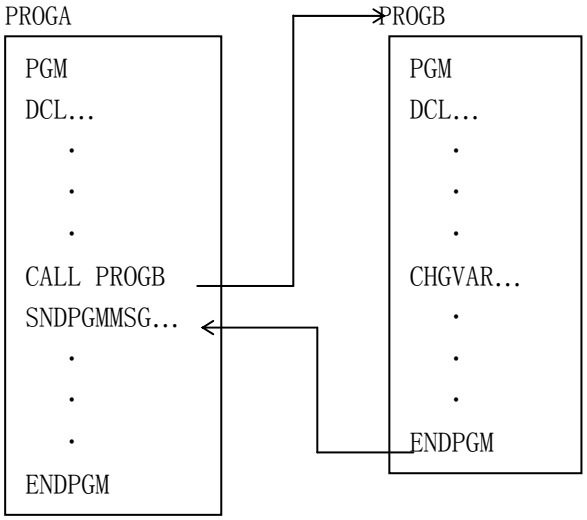
特别要注意的是用 CALL 或 CALLPRC 运行用 USRPRF(*OWNER)生成的程序，这与在主人用户配置文件下运行的程序处理有明显的不同安全特性。这章包括一般使用的程序设计接口（GUPI），可用在客户写的程序中。

3.1　CALL 命令

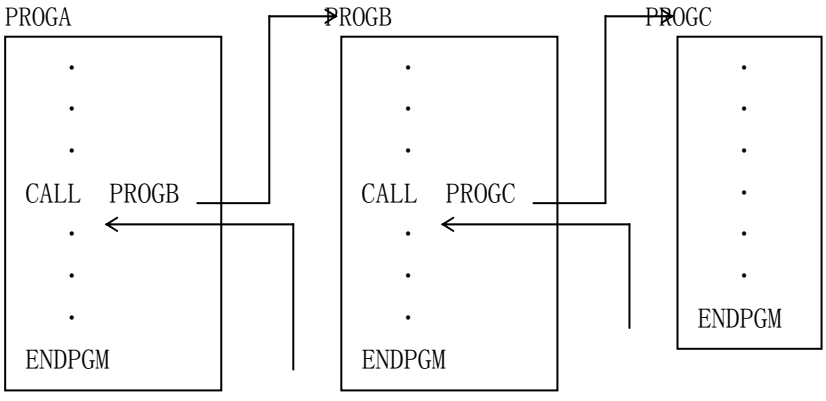
CALL 命令调用规定程序且把控制传给它，格式如下：

```
CALL    PGM(库存名/程序名)    PARM(参数值)
```

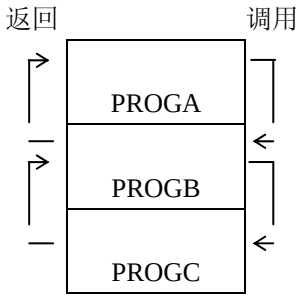
程序名和库名可以是变量。如果被调用的程序所在的库不在库列表中，必须在 PGM 参数中规定库名。PARM 参数在 3.4.4 中介绍。当被调用的程序完成运行，控制返回给调用程序的下一条命令。



程序中一些 CALL 命令的互相调用叫做调用堆栈。例如：



调用堆栈是：



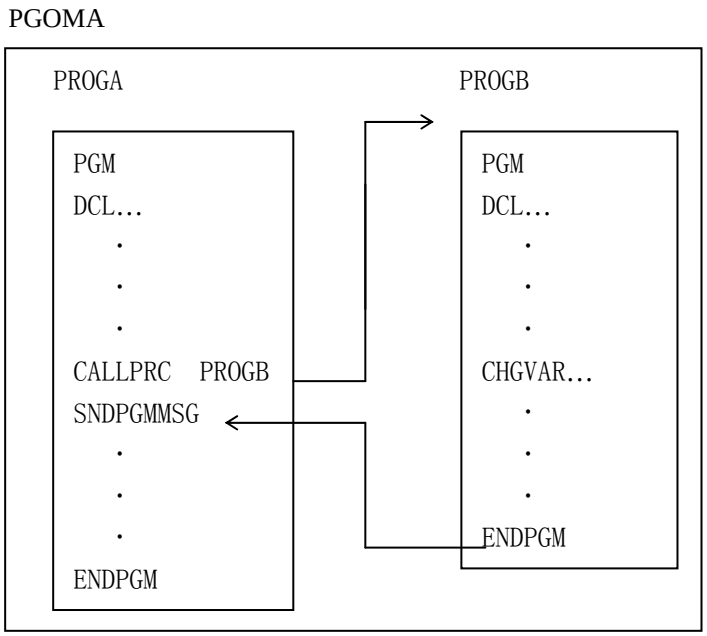
在 PROGC 完成操作时，控制返回到 PROGB 中调用 PROGC 的后一条命令，控制在堆栈中向上返回。无论 PROGC 用 RETURN 或 ENDPGM 结束都这样。CL 程序也能调用自身。

3.2　CALLPRC 命令

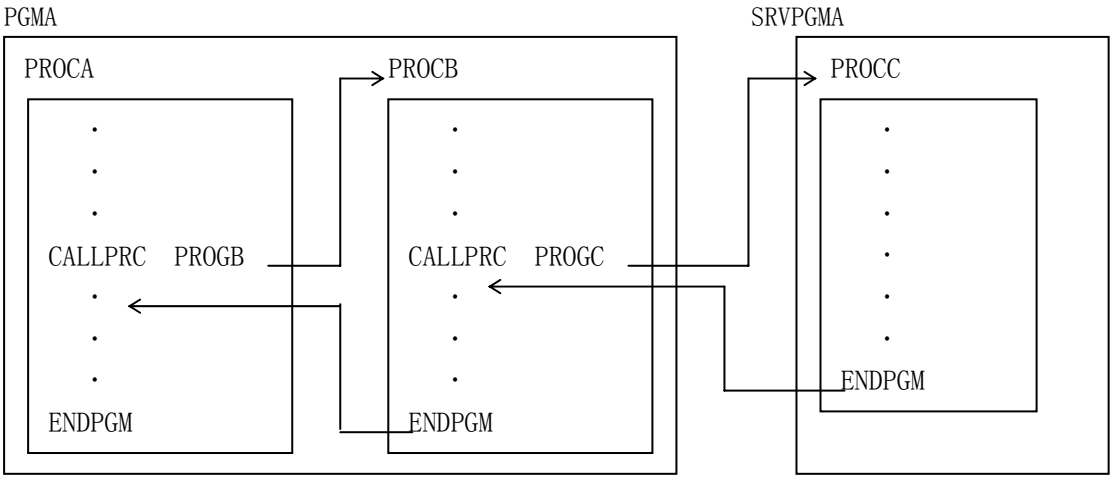
CALLPRC 命令调用规定的过程且把控制传给它。格式如下：

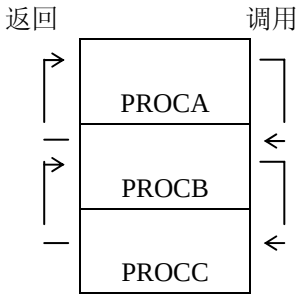
CALLPRC　PROCEDURE(过程名)　PARM(参数值)　RTNVAL(返回值变量)

过程名不能是变量，PARM 参数在 3.4 中介绍，当调用过程做完控制返回到调用过程的下一条命令。



过程中 CALLPRC 互相调用的顺序叫做调用堆栈，例如：





在 PROCC 完成操作时，控制返回到 PROCB 中调用 PROCC 的后一条命令，控制在堆栈中向上返回，无论 PROCC 用 RETURN 或 ENDPGM 结束都这样。CL 过程也能调用自身。

3.3 RETURN 命令

CL 过程中或 OPM 程序中的 RETURN 从调用堆栈中取消过程或 OPM 程序。如果用 CALLPRC 命令调用有 RETURN 的过程，控制返回到调用程序的 CALLPRC 命令后的下一条语句。

如果 MONMSG 命令规定了一个用 RETURN 结束的动作，控制返回到调用过程的下一条语句或有 MONMSG 命令的程序。

RETURN 设有参数。

注：如果在初始程序中用 RETURN 给出命令入口显示，出于安全考虑要避免发生这种情况。

3.4 在程序和过程之间传递参数

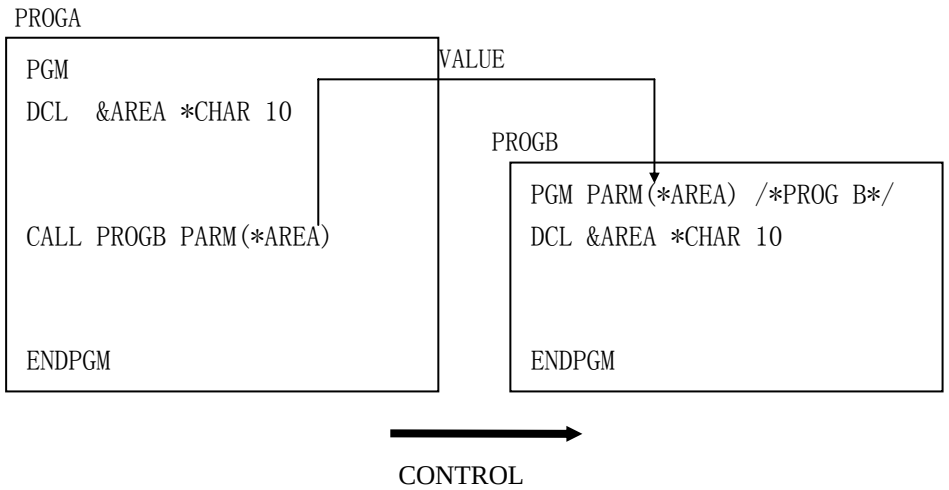
在把控制传给另外的程序或过程时，也能把修改或使用的信息传给接收的程序或过程，可在 CALL 或 CALLPRC 命令中的 PARM 规定要传送的信息。

如果 PROGA 有下面命令：

```
CALL    PROGB    PARM(&AREA)
```

它调用 PROGB 且把值&AREA 传送给它，PROGB 必须以 PGM 开始，且必须一定要有接收的参数：

```
PGM    PARM(&AREA)   /*PROGB*/
```



对这两个命令必须要有 PARM 参数，且在接收程序中的 PGM 中也要规定 PARM 参数。由于参数是按位置而不是按名字传送的，所以在 CALL 和 CALLPRC 中传送值的位置必须和接收的 PGM 中的位置相同。例如，PROGA 中有下列命令：

```
CALL    PROGB    PARM(&A   &B   &C   ABC)
```

它传送三个变量和一个字符串。如果 PROGB 用下列命令开始：

```
PGM    PARM(&C   &B   &A   &D)   /*PROGB*/
```

那么 PROGA 中的&A 就传给 PROGB 的&C。依此类推，&D 即为 ABC，而 PROGB 中 DCL 语句的顺序并不重要，仅是在 PGM 语句中规定的参数顺序决定变量传送的顺序。

对于位置参数，也要注意它们的长度和类型。在接收程序中列出的参数必须说明为与调用程序同样长度和类型，十进常量总是用（15　5）的长度传送。

在使用 CALLPRC 命令且传送字符串常量时，要规定确切的字节数且传送这个数。被调用的过程可用这些操作描述符中的信息确定传送的确切字节数，也可用 API CEEDOD 来访问操作描述符。

在使用 CALL 命令时，长度小于或等于 32 字节的字符串常量总以 32 字节传送。如果长度大于 32，必须给出确切的字节数且传送这个数。

CALLPRC 在执行过程中不检查传送的参数是否与期望值匹配，这就让程序员可以灵活的使用变量长度参数列表来编程。例如，在后来规定的参数列表中的参数不需要根据前面参数列表中值来确定，这也适应用其它 ILE 语言写的过程及调用 CL 过程。下面是一个接收&VAR1 的过程：

```
PGM PARM(&VAR1)   /*PGMA*/
DCL VAR1 *CHAR LEN(36)
.
.
```

```
.
ENDPGM
```

CALL 或 CALLPRC 命令必须规定 36 个字符：

```
CALLPRC PGMA(ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJ)
```

下列例子规定缺省长度：

```
PGM PARM(&P1 &P2)
DCL VAR(&P1) TYPE(*CHAR) LEN(32)
DCL VAR(&P2) TYPE(*DEC) LEN(15 5)
IF (&P1 *EQ DATA) THEN(CALL MYPROG &P2)
ENDPGM
```

要调用此程序，规定 CALL PROG(DATA 136)

则字符串 DATA 传给&P1，十进值 136 传给&P2。

引用本地定义的变量要比引用传送的变量花费要少。这样，被调用程序如果频繁地引用传送来的变量，就不如把传送的值复制到本地变量中来引用，这样性能要好一些。

在调用用一个 OPM CL 程序时，传送给它的参数数量必须确切的与接收程序预计的数量一致。这个预计值是由程序生成时确定的（操作系统要限制你调用与被调用程序的参数不一致）。在调用一个 ILE 程序或过程时，操作系统不检查调用与被调用的参数的数量。另外，操作系统存储参数的空间在程序调用时并不重新初始化。如果预计有 n 个参数的过程调用了 n-1 个参数，它将用参数空间的值来访问第 n 个参数。这样，结果是不可预测的，这也适合用其它 ILE 语言写的调用 CL 过程的程序，或由 CL 过程调用的过程。

这给用户写 ILE CL 过程的更大的灵活性。这是因为可以写有变量长度参数列表的过程。例如，根据一个参数的值，可不需要在此列表后面规定它，如果指出的控制参数是一个没规定的可选参数，那么被调用过程就不去引用 这个可选参数。

可用*OMIT 来规定从参数列表中去掉某些参数。这时，被调用的过程传送空指针。在 CL 语言中，可在第一次引用去掉的参数时监控 MCH3601 信息来检查空指针。在过程中要规定有 MCH3601 时要采取的动作。

下面的例子中有二个 CL 过程，第一个预计有一个参数，如果参数不是期望值，则结果不可预测，第一个过程调用第二个过程 PROC1。PROC1 预计有一个或二个参数，如果第一个参数为 1，则希望规定第二个参数，如果第二个参数为‘0’，假定第二个参数为非预计值而使用缺省值。PROC1 也用 API CEEDOD 来确定从第二个参数传来的实际长度。

```
MAIN: PGM   PARM(&TEXT)/* 必须规定&TEXT，如果不规定结果不可预测          */
      DCL   VAR(&TEXT) TYPE(*CHAR) LEN(10)
      CALLPRC   PRC(PROC1) PARM('0')
      CALLPRC   PRC(PROC1) PARM('1' &TEXT)
      CALLPRC   PRC(PROC1) PARM('1''Goodbye')
      ENDPGM
PROC1: PGM   PARM(&P1 &P2) /* PROC1 – 带有可选参数&P2的过程            */
```

```
DCL  VAR(&P1) TYPE(*LGL) /*标记指出是否规定了&P2，如果值为1，
                                                    即规定了&P2。 */
DCL   VAR(&P2) TYPE(*CHAR) LEN(10)
DCL   VAR(&MSG) TYPE(*CHAR) LEN(10)
DCL   VAR(&PARMPOS) TYPE(*CHAR) LEN(4) /* CEEDOD的参数位置        */
DCL   VAR(&PARMDESC) TYPE(*CHAR) LEN(4) /* CEEDOD的参数说明       */
DCL   VAR(&PARMTYPE) TYPE(*CHAR) LEN(4) /* CEEDOD的参数数据类型   */
DCL   VAR(&PARMINFO1) TYPE(*CHAR) LEN(4) /* CEEDOD的参数信息      */
DCL   VAR(&PARMINFO2) TYPE(*CHAR) LEN(4) /* CEEDOD的参数信息      */
DCL   VAR(&PARMLEN) TYPE(*CHAR) LEN(4) /* CEEDOD的参数长度        */
DCL   VAR(&PARMLEND) TYPE(*DEC) LEN(3 0) /* 参数长度的十进制格式 */
IF   COND(&P1) THEN(DO) /* 规定了Parm 2，用它作信息正文           */
CHGVAR  VAR(%BIN(&PARMPOS 1 4)) VALUE(2) /* 告诉CEEDOD对第二个参
                                                    数要操作主字码  */
CALLPRC  PRC(CEEDOD) PARM(&PARMPOS & PARMDESC +
     &PARMTYPE &PARMINFO1 &PARMINFO2 &PARMLEN) +
     /* 调用CEEDOD得到&P2的数据长度                              */
 CHGVAR  VAR(&PARMLEND) VALUE(%BIN(&PARMLEND 1 4)) /* 把由CEEDOD
                                        返回的长度转换成十进制格式 */
 CHGVAR  VAR(&MSG) VALUE(%SST(&P2 1 &PARMLEND)) /* 复制传送到本
                                                    地变量的数据  */
 ENDO
 ELSE   CMD(CHGVAR VAR(%MSG) VALUE('Hello')) /* 用"Hello" 作信息正文*/
 SNDPGMMSG MSG(&MSG)
 ENDPGM
```

3.4.1 使用 CALL 命令

在 CL 过程中发布 CALL 命令时，传给被调用程序的参数值可以是字符串常量、数值常量、逻辑常量或 CL 变量，最多可传送 40 个参数。参数的值是按在 CALL 命令中的顺序传送的，传送的变量名不必须与接收参数的名字相同，接收值的变量名必须在被调用程序中说明，但说明的顺序不重要。

被调用程序和它接收的变量在存储位置之间没有联系。当传送变量时，变量存储在最初说明它的程序中，变量是由地址传送的。当传送常量时，常量的复本出现在调用程序中，复本的地址传送给被调用程序。

变量传给被调用程序时，它可以被修改，且修改影响到调用程序，新值不必须返回给调用程序以后用，即不需要特别编码把变量的值返回给调用程序。当传送常量时，它的值可由被调用用程序修改，也不用通知调用程序。这样，如果调用程序再重新调用同一程序，它要再初始化成常量值而不是变量的值。

使用 CALL 调用一个还没有编译的 CL 程序时（交互的 CALL 或用 SBMJOB），*DEC 类型的参数用 LEN(15　5)，*CHAR 用 LEN(32)传给接收程序。

不在 CL 过程或程序中的 CALL 命令不能传送变量。当 CALL 运行一个定义为*CMDSTR 的命令参数时，在 PARM 中定义的任何变量的内容都转换成常量。在 SBMJOB、ADDJOBSCDE 和 CHGJOBSCDE 命令中的 CMD 参数便是例子。

在传递和接收参数时要注意下列情况：

长度等于或小于 32 字节的字符常量可用 32 字节长传送。（在右边加空格）。如果它比 32 字节长，则传送整个长度，如果定义的参数长度大于 32 字节，CALL 命令必须传送确切长度的常量。比 32 长的常量不添加空格来匹配接收程序规定的长度。接收程序可以接收传过来参数的一部分。例如，如果程序规定要接收 4 个字符，而传过来的是 ABCDEF（其余 26 位空格），则仅接收 ABCD 给程序用。

如果接收程序接收比传过来多的字节数，则结果不可预测。做为字符值传送的数字值必须用引号括起。

系统用*CMDSTR 计算参数中的字符。因括号内的第一个字符做为计数的第一个字符，命令名、参数名、空格、变量长度、引号都做为字符计数。如果在命令串中使用字符变量，要加一个替代值且计算引号。

十进常量用压缩格式传送且长度为(15　5)，这样，如果传关 12345，接收程序必须用 LEN(15　5)说明一个十进字段，参数以 12345.00000 接收。

如果传送数值常量而接收程序不想以（15　5）的长度接收，则常量要转换成十六进制格式。下列命令是要把 25.5 传给程序变量，说明为 LEN(5　2)

```
CALL PGMA PARM(X'02550F')
```

逻辑常量用 32 个字节的长度传送，逻辑值 0 或 1 放在第一个字节，其余字节为空格，如果非 0，1 值传给用逻辑值的程序，结果是不可预测的。

浮点整数或浮点特殊值（*NAN、*INF 或*NEGINF）做双精度值传送，它含 8 个字节，虽然 CL 程序不能处理浮点数，但它能接收浮点值放在字符变量中，且把变量传给能处理浮点值的 HLL 程序。

从 CL 程序或过程中的 CALL 也能传送变量，此时接收程序必须说明这个字段使之与调用程序或过程中定义的变量相匹配。例如，如果 CL 过程或程序定义了一个 LEN(5　0)的变量&CHKUM，接收程序必须说明字段为压缩的无小数位的 5 位数字。在 CL 过程或程序中用 SBMJOB 使 CALL 在批方式下运行时，变量做为常量来对待。

如果十进常量或程序变量能传给被调用程序，参数必须定义为 LEN(15　5)。任何调用程序须符合定义，如果类型、数量、顺序和长度在调用和接收程序之间不匹配结果不可预测。

由于空值不能传给另外的程序。反以不能用*N 规定空值。

在下例中，程序 A 传送六个参数：一个逻辑值、三个变量、一个字符常量和一个数值常量。

```
PGM /* PROGRAM A */
DCL VAR(&B) TYPE(*CHAR)
DCL VAR(&C) TYPE(*DEC) LEN(15 5) VALUE(13.529)
DCL VAR(&D) TYPE(*CHAR) VALUE('1234.56')
CHGVAR VAR(&B) VALUE(ABCDEF)
CALL PGM(B) PARM('1' &B &C &D XYZ 2)  /* 注意参数间的空格 */
.
.
.
ENDPGM
```

```
PGM PARM(&A &B &C &W &V &U) /* PROGRAM B */
DCL VAR(&A) TYPE(*LGL)
DCL VAR(&B) TYPE(*CHAR) LEN(4)
DCL VAR(&C) TYPE(*DEC)
            /* 缺省长度 (15 5) 适合程序A的 DCL LEN  */
DCL VAR(&W) TYPE(*CHAR)
DCL VAR(&V) TYPE(*CHAR)
DCL VAR(&U) TYPE(*DEC)
.
.
. ENDPGM
```

注：如果传给 PGMB 的第五个参数是 456 而不是 XYZ 且扩充为字母数字数据，则参数的值为 456。

逻辑常量‘1’不非得在调用程序中说明，它在程序 B 中的名为&A 是逻辑类型。由于在&B 的DCL 语句中没有规定长度，所以传送缺省长度 32 个字符，仅规定了&B 的六个字符（ABCDEF），由于在程序 B 中&B 说明为四个字符，那么仅接收四个字符。如果在程序 B 中修改了它们，那么在程序 A 中的&B 的四位也要修改。

在程序 A 中必须规定&C 的 LEN 参数，如果没规定，用缺省长度，它将与程序 B 中想要的缺省长度不兼容，&C 的值为 13.52900。

程序 B 中的&W（程序 A 的&D）由于说明是字符型，故做为字符接收。如果类型为*CHAR，则不用引号。在程序 A 中，长度为缺省值 7（小数点也考虑做一位），程序 B 希望长度为 32，传送头 7 个字符，而 7 位后的内容不能断定。

变量&V 是字符串 XYZ，在左边添空格。变量&U 是数值数据 2.00000。

3.4.2 在调用过程或程序中的一般错误

下面介绍用 CALL 或 CALLPRC 命令传送值时比较常见的错误。其中某些错误很难调试，某些对程序功能产生严重后果。

3.4.2.1 使用 CALL 的数据类型错

命令串的总长度包括命令名、空格、参数名、括号、变量的内容及引号，对大多数命令，命令初始为处理程序期望的那样，但某些命令或变量不象预期的那样传送。

当在 SBMJOB 中与 CMD 参数一起用 CALL 时，可能发生非期望情况。按语法规则，当使用 CMD 参数和直接编译 CALL 命令都可出现 CALL 语句，在用 CMD 参数时，CALL 命令转换成命令串，批处理子系统初始化它以便后来运行。当 CALL 用于自身时，CL 编译程序生成编码来完成调用。

一般经常发生问题的是十进常量和字符串变量，在下列情况下不需要构成命令串：

十进数转换成十进常量。

在运行命令串时，十进常量用 LEN(15　5)压缩格式传送，它不以在 CL 变量中规定的格式传送。

字符变量说明为长度大于 32 字符。

字符变量的内容如前所述传送。通常做为去掉尾部空格的字符常量放在引号内。这样，被调用程序可能得不到足够的数据。

下面的方法可以改正构成命令串时发生的错误：

生成一个用于提交的 CALL 命令串，它把命令的变量部分连在一起形成一个 CL 变量，用 SBMJOB 中的 RQSDTA 参数提交这个命令。

对于长度大于 32 个字符且结尾空白有意义的 CL 字符变量，生成一个的需长度的变量，以非空的字符结尾，这就避免了截断有意义的空白。被调用程序由于依据了预期的长度而忽略额外的字符。

生成一个初始化被调用程序的命令。提交这个新命令代替 CALL 命令，命令定义保证了参数用处理程序期望的那样传送给命令。

3.4.2.2　数据类型错

当传送一个值时，调用过程或程序与被调用过程或程序的数据类型（TYPE 参数）必须相同（*CHAR，*DEC 或*LGL）。当试图传送数值常量时经常发生类型错误。如果数字常量放在引号中，它是做为字符串传送的，如果不放在引号中，它是按 LEN(15　5)的压缩数值字段传送的。

在下例中，引号中的数字值传给希望是十进值的程序，在被调用程序（PGMA）引用&A 时发生十进错误（MCH1202）

```
CALL    PGMA PARM('123')  /* CALLING PROGRAM */
PGM     PARM(&A)  /* PGMA */
DCL     &A *DEC LEN(15 5)  /* DEFAULT LENGTH */
.
.
.
IF (&A *GT 0) THEN(...) /* MCH1202 OCCURS HERE */
```

在下例中，一个十进值传送给定义了字符变量的程序。一般说，这个错误不导致运行失败，但通常是结果不正确：

```
CALL PGMB PARM(12345678)  /* CALLING PROG */

PGM PARM(&A)                /* PGMB */
DCL &A *CHAR 8
.
.
.
ENDPGM
```

PGMB 中的&A 有十六进制值 001234567800000F。

一般讲，数据可以从逻辑变量传送给字符变量，反之也可，不出错，值为‘0’或‘1’。

3.4.2.3　十进长度和精度错

如果一个十进制用不正确的十进长度和精度（太长或太短）来传送，在引用变量时会发生十进数据错（MCH1202）。在下例中，数值常量以 LEN(15　5)传送，而在被调用程序中说明为 LEN(5　2)。数值常量总是用压缩十进制（15　5）传送：

```
CALL PGMA PARM(123)      /* CALLING PROG */
```

```
PGM PARM(&A)                    /* PGMA */
DCL &A *DEC (5 2)
.
.
.
IF (&A *GT 0) THEN(...) /* MCH1202 OCCURS HERE */
```

如果一个十进变量在调用过程或程序中说明为 LEN(5　2)，传送时做为变量而不是常量，则不发生错误。

如果要传送一个数值常量给过程或程序，而它希望长度不是（15　5），常量可用十六进制编码。下列命令给出如何把值 25.5 传给说明为（5　2）的程序变量：

```
CALL    PGMA    PARM(X‘02550F’)
```

如果一个十进值用正确长度但错误的精度（小数位）传送，接收的过程或程序就不能正确解释。在下例中，数值常量值（15　5）传给程序时处理为 25124.00。

```
CALL PGMA PARM(25.124)  /* CALLING PGM */

PGM PARM(&A)            /* PGMA */
DCL &A *DEC (15 2)      /* LEN SHOULD BE 15 5*/
.
.
.
ENDPGM
```

这个错误发生在第一次引用变量时，而不是传送和说明变量时。在下例中，被调用程序不引用变量，而是简单的放一个值在变量中返给调用程序，在变量返给调用程序后第一次引用时才发生错误，这类错误很难发现。

```
PGM             /* PGMA */
DCL &A *DEC (7 2)
CALL PGMB PARM(&A)  /* (7 2) 传送给程序B */
IF (&A *NE 0)  THEN(...)  /* 这里发生 *MCH1202 */
.
.
ENDPGM


PGM PARM(&A) /* PGMB */
DCL &A *DEC (5 2)         /* 错误长度 */
.
.
```

```
.
 CHGVAR &A (&B-&C)       /* 把值放在 &A 中*/
RETURN
```

当控制返给 PGMA，第一次引用&A 时出错。

3.4.2.4 字符长度错

如果传送的字符值比接收变量的字符长，接收的过程或程序不能访问超出的长度。在下例中，PGMB 把传给它的变量改为空格，由于变量说明为 LEN(5)，在 PGMB 中，仅有 5 个字符改成了空格，而 PGMA 引用时，其余部分字符不变。

```
PGM         /* PGMA */
DCL &A *CHAR 10
CHGVAR &A 'ABCDEFGHIJ'
CALL PGMB PARM(&A)            /* PASS to PGMB */
.
.
.
IF (&A *EQ '  ') THEN(...)  /* 检查失败 */
ENDPGM

PGM PARM(&A)      /* PGMB */
DCL &A *CHAR 5    /* 长度错误 */
CHGVAR &A '  '    /* 仅用5位；其他没有影响 */
RETURN
```

这类错误不引起逃逸信息，用这种方法处理的变量功能与期望值不同。

如果传送的值比要接收的值短，那么可能是更严重的后果。这时，被调用程序中变量的值由传送的原值组成，其余部分就不知道是什么内容。如果传送的值是变量，它可能由其它变量接上或由过程或程序的内部控制指令接上，如果传送的值是常量，它可能由 CALL 或 CALLPRC 传送的其它常量接上或由内部控制指令接上。

假如接收的过程或程序修改这个值，它在原值和所在存储上操作，这马上会影响修改其它变量或常量，或修改内部指令而引起过程或程序失败，修改所在存储会立即起作用。

在下例中，二个三字符常量传给被调用程序，用 CALL 命令传送最少 32 个字符。（通常，传送 3 个字符后用空格填充）如果接收程序说明接收变量比 32 长，那么用未知的所有存储填上额外的位，在此例中，假定二个常量在存储中是相邻的：

```
CALL PGMA ('ABC''DEF')  /* PASSING PROG */

PGM  PARM(&A &B) /* PGMA */
DCL  &A *CHAR 50  /* VALUE:ABC+29''+DEF+15'' */
DCL  &B *CHAR 10  /* VALUE:DEF+7'' */
CHGVAR VAR(&A) ('')  /* THIS ALSO BLANKS &B */
.
```

```
.
.
ENDPGM
```

做为变量传送的值情况与此非常一样。

在下例中，二个三字符常量传给被调用程序。用 CALLPRC 命令仅传送规定的字符，如果接收程序说明的变量比传送的长，额外的位置用未知的所在存储填充：

在下例中，假定两个常量在存储中是相邻的：

```
CALLPRC PRCA ('ABC''DEF')  /* PASSING PROG */

PGM  PARM(&A &B)        /*  *PRCA  */
DCL  &A *CHAR 5         /*   VALUE:'ABC' + 'DE' */
DCL  &B *CHAR 3         /*   VALUE:'DEF' */
CHGVAR &A ' '           /*    &B的头两个字节为空格 */
.
.
.
ENDPGM
```

3.5　使用数据队列在程序和过程间通讯

数据队列是程序员生成的系统目标。HLL 过程或程序可以往其中传送数据，另外的过程或程序可以从中接收数据，接收程序可以等待数据，也可在晚些时候接收数据。使用数据队列的优点是：

使用数据队列可以使作业更灵活的做一些工作。如果作业是交互作业，它可以提供更好的响应时间，减少交互程序的大小及它的处理访问组（PAG）。这样，就可帮助提高系统性能。例如，几个工作站用户都做修改和增加文件记录的操作，那么交互作业把请求都交给某个交易做为一个批作业执行，这样系统会完成的更好。

数据队列是两个作业之间异步通讯最快的方法，它比使用数据文件、信息队列发送和接收数据要更合算。

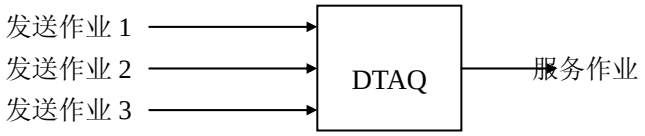
可在任何 HLL 过程或程序中调用 QSNDDTAQ、QRCVDTAQ、QCLRDTA、QMHRDQM 和 QMHQRDQD 程序来与数据队列传送数据和得到数据队列的描述，而不用结束 HLL 或调用其它 CL 过程或程序做这些事。

当从数据队列接收数据时，可以设置一个延迟时间，让作业在此时间内等着接收数据队列来的数据。这与 OVRDBF 命令中的 EOFDLY 参数是不同的，它在无论何时延迟时间结束，作业是活动的。

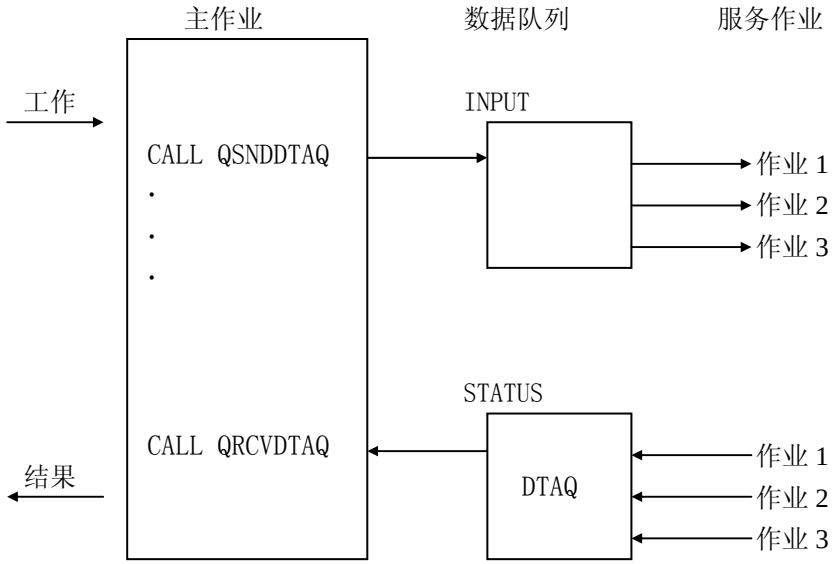
多个作业可从同一数据队列接收数据，这就使在一定的性能限制下应用程序可以处理多个作业。例如，在系统中有多个打印机可用，几个交互作业往一个数据队列发送请求，每个打印机作业可从数据队列接收数据，可用 FIFO、LIFO 或键字顺序来接收数据。

数据队列给每个放在其中的信息建一个发送 ID。这个 ID 是在生成队列时建立起来的属性，其中有限定的作业名和用户配置文件信息。

下例给出数据队列如何工作，几个作业在同一数据队列都有入口，这些入口是由一个服务作业管理的，它使多个作业往一个打印作业中发送处理请求，多个作业可往同一队列发送数据：



下面是另一个使用数据队列的例子。作业得到工作请求把它送到数据队列中（调用 QSNDDTAQ）。服务作业从数据队列接收请求（调用 QRCVDTAQ）且处理数据。服务作业也能用另外的数据队列把状态报告给主作业。数据队列允许主作业把工作按规定路线发送给服务作业，这就让主作业可以接收下一个工作请求，可有多个服务作业从同一数据队列接收信息。



当数据队列中无信息时，服务作业有以下选项：

等待队列中有信息

等待一个时间周期，如仍无信息到达，则继续处理

不等待，立即返回

数据队列也能用于程序需要等待从数据队列与过程或程序通讯的显示文件、ICF 文件和数据队列输入数据时，此时，要在下列命令中用 DATQ 参数：

CRTDSPF、CHGDSPF、OVRDSPF、CRTICCF、CHGICFF、OVRICFF

可以在下列情况下指出数据队列中已有信息：

从请求的显示设备上输入 ENABLE 命令或按执行键。

数据从请求的 ICF 对话中成为可用的。

用 CRTOUTQ 和 CHGOUTQ 命令可以使与数据队列有关的选项用于输出队列，在假脱机文件在输出队列中是 RDY（准备好）状态时，会在数据队列中有一个入口记录，用户程序能确定什么时候有可用的 SPLF，再从数据队列接收信息。

系统中运行的作业能用在 QSNDDTAQ 程序中规定 DATQ 参数来往同一数据队列中放信息，用程序调用 QRCVDTAQ 接收放在数据队列中的每项信息，然后根据是由显示文件、ICF 文件或 QSNDDTAQ 发送的不同情况来进行处理。详细内容请看 3.5.7.2 和 3.5.7.3。

3.5.1 远程数据队列

可用 DDM 文件访问远程数据队列，用 DDM 文件可使在一个 AS/400 的程序访问远程 AS/400 的数据队列来完成下列功能：

往数据队列发送数据
从数据队列接收数据
清除数据队列中的数据

当前使用标准数据队列的应用程序不用再修改或编译就能访问远程的 DDM 数据队列。要保证访问正确，可做以下之一：

删除标准的数据队列，生成同名的 DDM 数据队列
把标准的数据队列改名

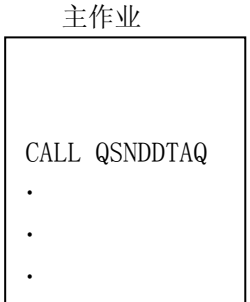
可用下列命令生成 DDM 数据队列：

```
CRTDTAQ DTAQ(LOCALLIB/DDMDTAQ) TYPE(*DDM)
   RMTDTAQ(REMOTELIB/REMOTEDTAQ) RMTLOCNAME(SYSTEMB)
   TEXT('DDM data queue to access data queue on SYSTEMB')
```

也可用前面的例子（主作业/服务作业）来生成 DDM 数据队列来使用远程数据队列。主作业在系统 A 中，数据队列和服务作业放在系统 B 中。在生成两 个 DDM 数据队列后（INPUT 和 STATUS），主作业继续异步地与系统 B 中的服务作业通信。下例给出如何生成做为远程数据队列的 DDM 数据队列：

```
CRTDTAQ DTAQ(LOCALLIB/INPUT) TYPE(*DDM)
   RMTDTAQ(REMOTELIB/INPUT) RMTLOCNAME(SystemB)
   TEXT('DDM data queue to access INPUT on SYSTEMB')

CRTDTAQ DTAQ(LOCALLIB/STATUS) TYPE(*DDM)
   RMTDTAQ(REMOTELIB/STATUS) RMTLOCNAME(SystemB)
   TEXT('DDM data queue to access STATUS on SYSTEMB')
```

主作业调用 QSNDDTAQ，然后传送数据队列名 LOCALLIB/INPUT 和数据到系统 B 的远程数据队列中（REMOTELIB/INPUT）。主作业调用 QRCVDTAQ 来传送数据队列名 LOCALLIB/INPUT。



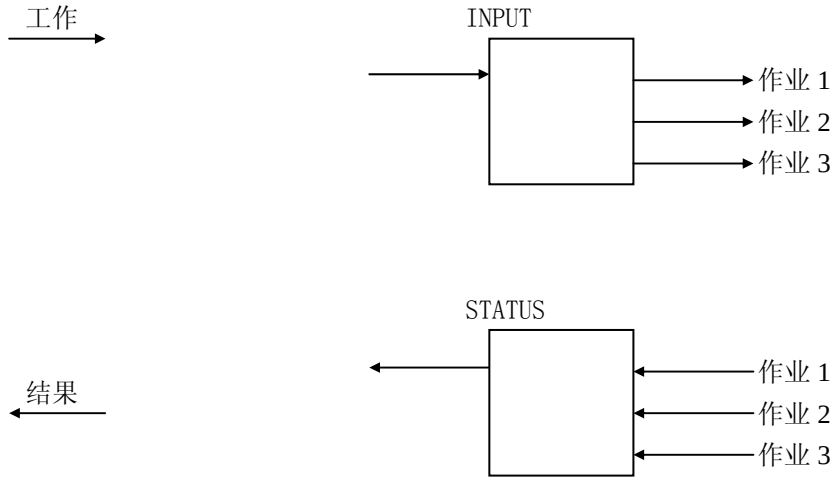


图 3—1 访问远程数据队列的例子

3.5.2　使用数据文件和数据队列的比较

下面说明使用数据队列与数据文件间的不同：

数据队列能在活动的过程或程序间提供通讯，不能存储大量的数据，如果要这样，用数据文件做队列。

数据队列不能用做数据的长项目存储，要做这用数据文件。

在使用数据队列时，在程序中要有非正常结束的例程，来在系统结束前恢复没处理完的项目。

最好是周期性的（比如一天一次）删除、再生成数据队列，如果队列中有太多内容不取消，则会影响系统性能。周期性的再生成数据队列能使数据队列恢复到满意的大小。

3.5.3　与信息队列的相似点

数据队列与信息队列很相似，这体现在传送和接收数据方面。但多个程序可同时从一个数据队列接收数据，而只有一个程序能从一个信息队列中接收数据。（仅一个程序从数据队列接收一项，其它多个程序要等待）。数据队列中的项是按 FIFO、LIFO 或键字顺序处理的，当接收一项时，它从数据队列中取消。

3.5.4　使用数据队列的准备工作

要使用数据队列，要先用 CRTDTAQ 命令生成，下面是命令的例子：

```
CRTDTAQ DTAQ(MYLIB/INPUT)  MAXLEN(128)
        TEXT('Sample data queue')
```

MAXLEN 参数规定送到数据队列中项的总长度（1—64512 字符）。

3.5.5　数据队列使用的存储管理

在从数据队列接收一项后，这项从数据队列取消但并没释放所占存储。在数据队列中有新的项送到时，会再使用同一存储空间，队列所占空间随着送到项的增加但没被接收而增大。如果数据队列的大小能保持在不多于 100 项时，性能会好，如果数据队列太大，要用 DLTDTAQ 命

令删除，再用 CRTDTAQ 生成。

3.5.6 分配数据队列

如果应用程序需要一个不能同时让多个作业访问的数据队列，那么在使用数据队列之前要写一个有 ALCOBJ 命令的语句，在应用程序使用完它之后用 DLCOBJ 命令重新分配数据队列。

ALCOBJ 命令本身不能限制另外的作业从数据队列中发送和接收数据及清除数据队列。但如果所有的应用程序在使用数据队列前都编码了包括 ALCOBJ 命令的语句，那么在数据队列已经分配给一个作业时如再分配给另一个就会失败，即在同一时刻要防止多个作业使用数据队列。此时，系统分发送 CPF1002 错误信息，可在应用程序中用 MONMSG 命令监控这个信息且响应它，可以给用户发送信息来重新分配数据队列。

3.5.7 使用数据队列的例子

下面的例子给出处理数据队列文件的三种方法。

3.5.7.1 例 1．等待二小时从数据队列接收数据

下例中，程序 B 等待二小时从数据队列接收数据项，程序 A 往 QGPL 库中的 DTAQ1 数据队列发送数据项。如果程序 A 在二小时之内发送了一项，程序 B 接收它，立即处理，如果过了二小时程序 A 没发送，由于字段长度为零，故程序 B 处理超时条件。程序 B 继续接收数据项直到出现超时条件。此程序是用 CL 写的，但可用任何高级语言写。

用下列命令生成数据队列：

```
CRTDTAQ DTAQ(QGPL/DTAQ1) MAXLEN(80)
```

在此例中，所有数据队列项都是 80 字节长。

在程序 A 中，下列语句与数据队列有关：

```
PGM
DCL  &FLDLEN  *DEC  LEN(5 0)  VALUE(80)
DCL  &FIELD  *CHAR  LEN(80)
.
.(determine data to be sent to the queue)
.
CALL  QSNDDTAQ  PARM(DTAQ1 QGPL &FLDLEN &FIELD)
.
.
.
```

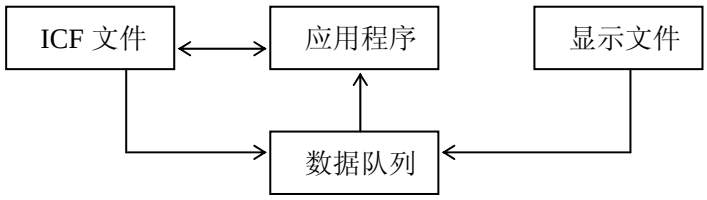
在程序 B 中，下列语句与数据队列有关：

```
PGM
DCL  &FLDLEN  *DEC  LEN(5 0)  VALUE(80)
DCL  &FIELD  *CHAR  LEN(80)
DCL  &WAIT  *DEC  LEN(5 0)  VALUE(7200)   /* 2 小时 */
.
```

```
            .
            .
LOOP:   CALL QRCVDTAQ  PARM(DTAQ1 QGPL &FLDLEN &FIELD &WAIT)
        IF     (&FLDLEN *NE 0) DO    /* 接收数据项 */
               .
               . (处理数据队列中的数据)
               .
               GOTO LOOP    /* 从数据队列得到下一项 */
        ENDDO
        .
        . (2小时没有接收到数据；处理超时条件)
        .
```

3.5.7.2　例 2．等待从显示文件和 ICF 文件的输入

下例中只有一个作业用数据队列。数据队列在作业中的各目标之间通讯而不仅是在两个作业之间通讯。



在此例中，程序等待来自显示文件和 ICF 文件中的输入。

不用交替的一个接一个等待，而用数据队列使程序等待一个目标（数据队列）。程序调用 QRCVDTAQ 且等待显示文件和 ICF 文件中规定的放到数据队列中的项。当在这两个文件中有可用数据时，相应的数据管理把两类项放到数据队列中。ICF 文件项用*ICFF 开始，显示文件项用*DSPF 开始，它们放在数据队列中是 80 个字符长，其中包括在下表中给出的属性说明，这样，用 CRTDSPF、CHGDSPF、OVRDSPF、CRTICFF、CHGICFF、OVRICFF 命令规定的数据队列必须至少有 80 个字符长。

位置	数据类型	说明
1－10	字符	放在数据队列中的文件类型，有两个：*DSPF 和*ICFF 如果作业从队列接收的数据仅打开了一个显示文件或 ICF 文件，仅用这一个字段来决定从数据队列接收的项类型。
11－12	二进制	这是唯一的文件标识，这个标识与文件的打开反馈区值是一样的。仅在多个文件以相同名字往数据队列中放数据项时，程序用此字段接收数据队列中的项。
13－22	字符	显示文件或 ICF 文件名，这是在所有替换命令处理完

		后实际打开的文件名，与在文件的打开反馈区的名字相同。它仅用在有多个显示文件或 ICF 文件在数据队列中有数据项时，程序用这个字段接收数据项。
23－32	字符	文件所在的库名。这是一个在所有替换操作执行完后的库名，它与文件在打开反馈区中所用库名相同。这个字段仅用在有多个显示文件和 ICF 文件在数据队列中有数据项时，程序用它接收数据项。
33－42	字符	程序设备名，它与在打开反馈区中程序设备定义清单中所用的名相同。对于*DSPF，这是有可用数据的程序设备名，仅在有多个设备或对话往数据队列放数据项时，程序才用它接收数据项。
43－80	字符	保留。

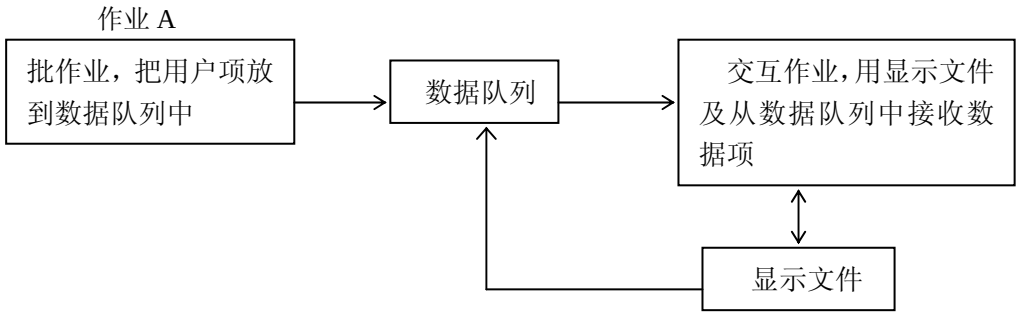
下面给出前面程序的编码逻辑：

```
  .
  .
  .
  .
  OPEN DSPFILE ... /* 打开显示文件。 在文件的CRTDSPF, CHGDSPF, 或
                                    OVRDSPF命令中规定 DTAQ 参数       */

  OPEN ICFFILE ... /* 打开ICF文件。在文件的CRTICFF, CHGICFF, 或
                                    OVRICFF命令中规定 DTAQ 参数       */
.
.
DO
  WRITE DSPFILE   /* 显示文件写请求                                    */
  WRITE ICFFILE   /* ICF文件写请求                                     */

  CALL QRCVDTAQ   /* 从文件的DTAQ参数规定的数据队列接收数据项。        */
                  /* 在请求设备上或在文件对话中有可用数据时，把数据    */
                  /* 项放在数据队列中。                                */
                  /* 接收数据项后，确定哪个文件有可用数据，读数据处    */
                  /* 理它，再请求文件，返回到处理数据队列的下一项。    */
  IF 'ENTRY TYPE' FIELD = '*DSPF     ' THEN   /* 来自显示文件的数据项，*/
    DO                                        /*  此项不包括接收的数据，*/
                                              /* 那么要先从文件中读数据 */
                                              /* 然后处理。             */
      READ DATA FROM DISPLAY FILE
      PROCESS INPUT DATA FROM DISPLAY FILE
      WRITE TO DISPLAY FILE                    /* 写请求                */
    END
  ELSE                                        /* 来自ICF文件的数据项，  */
                                              /*  此项不包括接收的数据，*/
```

```
                                        /* 那么要先从文件中读数据 */
                                        /* 然后处理。             */
      READ DATA FROM ICF  FILE
      PROCESS INPUT DATA FROM ICF FILE
      WRITE TO ICF FILE                  /* 写请求               */
  LOOP BACK TO RECEIVE ENTRY FROM DATA QUEUE
.
.
.
END
```

3.5.7.3　　例 3．等待来自显示文件和数据队列的输入

在下例中，作业 B 中的程序等待从作业 A 中往数据队列中放数据项。它是用显示文件送给作业 B 中的程序。替代的方法是程序仅从数据队列一个目标中得到输入。



程序调用 RCVDTAQ，等待数据项放到显示文件规定的数据队列中，作业 A 也把项放在同一数据队列中。有两类项放到这个队列中：显示文件项和用户定义的项。显示文件项是在显示文件中有可用数据时由显示数据管理放到数据队列中的项，用户定义的项是由作业 A 放到数据队列中的。

显示文件项的结构在前例中已说明，用户定义项的结构是由应用程序员定义的。

下例给出作业 B 中应用程序的编码逻辑：

```
 .
 .
 .
 OPEN DSPFILE ... /* 打开显示文件。在文件的CRTDSPF, CHGDSPF, or OVRDSPF*/
                  /*  命令中规定 DTAQ 参数                              */
.
.
DO

  WRITE DSPFILE   /* 写显示文件请求                                     */
```

```
CALL QRCVDTAQ    /* 从DTAQ规定的数据队列接收数据项，在显示文件请求的  */
                 /* 设备上有可用数据时，由作业A或显示数据管理把数据   */
                 /* 项放在数据队列中。                              */
                 /* 在接收数据项后，确定它的类型，处理它，返回到接收  */
                 /* 数据队列的下一项。                              */
IF 'ENTRY TYPE' FIELD = '*DSPF     ' THEN   /* 来自显示文件的数据项。*/
  DO                                        /*  此项不包括接收的数据，*/
                                            /* 那么要先从文件中读数据 */
                                            /* 然后处理。             */
    READ DATA FROM DISPLAY FILE
    PROCESS INPUT DATA FROM DISPLAY FILE
    WRITE TO DISPLAY FILE                    /* 写请求                */
  END
ELSE                                        /* 来自作业A的数据项。    */
                                            /* 它包括从作业A来的数据 */
                                            /* 所以在处理前不用读数据*/
    PROCESS DATA QUEUE ENTRY FROM JOB A
LOOP BACK TO RECEIVE ENTRY FROM DATA QUEUE
.
.
.
END
```

3.6 使用数据区在过程和程序间通讯

数据区是用来放数据以便在系统中运行的作业访问的一类目标。在需要存储有限大小的信息时用数据区。它与过程或文件是相独立的。数据区的典型用法是：

提供一个很容易很灵活修改的字段来控制作业中的引用。例如：

－提供分配的下一个顺序号。

－提供下一个检测数。

－提供下一个供保存 / 重存用的中间卷。

提供一个区来传送作业内的信息，这个区可在每个作业的 QTEMP 库中。

提供一个常量字段给几个作业用，例如税率和分布清单。

给需要数据区的较大处理程序提供一个有限的访问方式。

一个数据区可以由一个用户锁住。这样就避免其它用户同时操作。

要生成一个非本地或成组数据区，用 CRTDTAARA 命令，它可以在指定的库中生成有初值的数据区。要在 CL 过程或程序中使用这个值，用 RTVDTAARA 命令，它把数据区的当前值送给过程或程序中的变量。如果在 CL 过程或程序中修改此值且想把新值返回给数据区，用 CHGDTAARA 命令。要显示数据区中的当前值，用 DSPDTAARA 命令，可用 DLTDTAARA 命令删除数据区。

3.6.1 本地数据区

本地数据区是由系统中的每个作业建立的。包括自动启动的作业，由读出器启动的作业以及子系统监控的作业，系统来生成本地数据区且初值为全空格，长度为 1024，类型为*CHAR。

当用户用 SBMJOB 命令提交作业时，提交作业的数据区值复制到被提交作业的数据区中，可用 CHGDTAARA、RTVDTAARA、DSPDTAARA 中 DTAARA(*LDA)或对*LDA 用子串功能（%SST）来引用自己作业的本地数据区。

在使用本地数据区要注意：

不能从一个作业引用另一作业的本地数据区。

你不能生成、删除或分配本地数据区。

本地数据区不与库有关。

本地数据区的内容存在例程步边界之上。因此 TFRJOB、TFRBCHJOB、RRTJOB、RETURN 命令不会影响本地数据区的内容。

可用本地数据区做下列事情：

不用参数列表给过程或程序传送信息

把信息装入到酝数据区提交作业，把信息传给被提交的作业的作业，然后从被提交作业中访问数据

比用其它类型数据区访问过程或程序的性能要好

不用自己生成和删除数据区来存储信息

多数高级语言都能使用本地数据区。SBMxxxJOB 和 STRxxxRDR 命令启动一个本地数据区且把它初始化为空格。仅 SBMJOB 命令可以把提交作业的本地数据内容传送给新作业。

3.6.2 成组数据区

在交互作业成为组作业时（用 CHGGRPA 命令），系统生成成组数据区。对这个组仅可存在一个成组数据区，在组中最后一个作业结束时（用 ENDJOB、SIGNOFF 或 ENDGRPJOB 命令或异常结束）删除成组数据区，在作业不再是成组作业的一部分时，（在 CHGGRPA 中规定 GRPJOB（*NONE））也删除成组数据区。

成组数据区由空格初始化，长度为 512，类型为*CHAR。组中作业可用 CHGDTAARA、RTVDTAARA、DSPDTAARA 命令的参数 DTAARA（*GDA）来使用成组数据区，此数据区可为组中的所有作业使用。

在使用成组数据区时，要注意：

不能用成组数据区做字符变量的子串，但可用子串功能把 512 字节字符移进或移出成组数据区。

成组数据区不能被组外作业引用。

成组数据区与库无关。

成组数据区的内容不能用 TFRGRPJOB 命令修改。

用成组数据区可在同组中各作业之间通讯。例如，在使用 CHGGRPA 命令后，可用下列命令设置成组数据区的值：

```
CHGDTAARA  DTAARA(*GDA)  VALUE('january1988')
```

这个命令可从一个程序中运行也可由工作站用户使用。组中的任何其它 CL 过程或程序可以用下列命令取得数据区的值：

```
RTVDTAARA  DTAARA(*GDA)  RTNVAR(&GRPARA)
```

它把成组数据区的值（JANUARY1988）放到变量&GRPARA 中。

3.6.3 程序初始参数数据区（PIP）

PIP 是由每个预先启动作业在开始时建立的。它的属性 PDA 与一般的数据区不同，它仅有由规定*PDA 来引用。PDA 是 2000 字节长但不限制在其中的参数个数。

RTVDTAARA、CHGDTAARA、DSPDTAARA 命令及 RTVDTAARA 和 CHGDTAARA 宏指令支持数据区名字参数中规定*PDA。

3.6.4 远程数据区

可用 DDM 来访问远程数据区。在一个 AS/400 上的应用程序要修改或取得在远程 AS/400 的数据区，不用修改和重新编译应用程序。要保证访问到正确的数据，做下列之一：

删除标准数据区，再生成一个与其同名的 DDM 数据区

把标准数据区改名

可用下列命令生成 DDM 数据区：

```
CRTDTAARA DTAARA(LOCALLIB/DDMDTAARA) TYPE(*DDM)
   RMTDTAARA(REMOTELIB/RMTDTAARA) RMTLOCNAME(SYSTEMB)
   TEXT('DDM data area to access data area on SYSTEMB')
```

要在 CL 程序中使用远程 AS/400 的数据区，用 RTVDTAARA 命令，规定 DDM 数据区的名字把其当前值放到程序变量中。如果修改值后要把它放回到远程数据区，用 CHGDTAARA 命令，在其中规定用一个 DDM 数据区。

如果在用 DSPDTAARA 时规定 DDM 数据区的名字，就显示 DDM 数据区中的值，而不是远程数据区的值，可用 DLTDTAARA 命令删除 DDM 数据区。

3.6.5 生成数据区

数据区要先生成然后才能使用。一个数据区可以生成为：

最长为 2000 字符的字符串。

根据数据区是仅用在 CL 过程或程序或也用在其它高级语言过程或程序，使用不同属性的十进制值。对 CL 过程或程序，最多 15 位整数 9 位小数，但总共不能多于 15 位。对其它语言，最多 15 位整数 9 位小数，但总共不能多于 24 位。

逻辑值‘0’或‘1’。

在生成数据区时，可规定初值。如果没规定，则用下列值：

十进制用零，字符用空格，逻辑值用‘0’。

要生成数据区，用 CRTDTAARA 命令，如下例：

```
CRTDTAARA  DTAARA(CUST) TYPE(*DEC) +
           LEN(5 0) TEXT('Next customer number')
```

3.6.6 数据区加锁和分配

CHGDTAARA 命令中用*SHRUPD，在命令执行时会锁住数据区。

RTVDTAARA 命令和 DSPDTAARA 用*SHRRD，在命令执行时会锁住数据区。

如果在数据区完成多个操作，要用 ALCOBJ 命令来避免其它用户在操作没完成时访问数据区。例如，同时有作业要读和增加数据区中的值，那么就要用 ALCOBJ 命令来保护读和修改的

值。详细内容请看第四章。

3.6.7 显示一个数据区

可用 DSPDTAARA 命令显示一个数据区的属性（名字、库名、类型、长度、说明）以及值，显示使用 24 个数字格式，可存前置零。

3.6.8 修改一个数据区

可用 CHGDTAARA 命令修改一个数据区的全部或部分值，但不能修改其它属性。新值可以是常量或 CL 变量。如果命令是在 CL 过程中，则程序生成时数据区不一定要存在。

3.6.9 取得一个数据区

RTVDTAARA 命令取得数据区的部分或全部内容，把它复制到一个变量中去。在编译时数据区不一定要存在，CL 变量也不要求与数据区同名，它只是取得数据区的内容但不能修改它。

3.6.10 获得数据区的例子

例 1．假定用名为 ORDINFO 的数据区来跟踪一个文件的状态，数据区的设计为：

位置 1 为 O（打开），P（处理），或 C（完成）
位置 2 为 I（有存货），O（无存货）
位置 3 为定货单的初值

可如下说明字段：

```
DCL VAR(&ORDSTAT) TYPE(*CHAR) LEN(1)
DCL VAR(&STOCKC) TYPE(*CHAR) LEN(1)
DCL VAR(&CLERK) TYPE(*CHAR) LEN(3)
```

要把定货状态放在&ORDSTA 中，写如下命令：

```
RTVDTAARA DTAARA(ORDINFO (1  1))  RTNVAR(&ORDSTAT)
```

要取得库存条件放到&STOCK 中，写如下命令：

```
RTVDTAARA DTAARA(ORDINFO (2  1)) RTNVAR(&STOCKC)
```

要取得定货单的初始值放到&CLERK 中，写如下命令：

```
RTVDTAARA DTAARA(ORDINFO (3  3)) RTNVAR(&CLERK)
```

每个 RTVDTAARA 命令都要访问数据区。要取多个子字段，不如取整个数据区放到一个变量中，然后用子串功能获得各子字段。

例 2．下例把 5 个字符的数据区放到 3 个字符的变量中，做法如下：

生成名为 DA1，5 个字符的数据区（放在 MYLIB 库中）。初值为‘ABCDE’
说明名为&CLVAR1，长为 3 个字符的变量
把 DA1 的后三个字符复制到&CLVAR1 中
所用命令如下：

```
CRTDTAARA DTAARA(MYLIB/DA1) TYPE(*CHAR) LEN(5) VALUE(ABCDE)
.
.
.
DCL VAR(&CLVAR1) TYPE(*CHAR) LEN(3)
RTVDTAARA DTAARA(MYLIB/DA1 (3 3)) RTNVAR(&CLVAR1)
```

现在&CLVAR1 中有‘CDE’。

例 3．下例把 5 位数字的数据区内容放到 5 位数字的变量中。

做法如下：

生成名为 DA2(放在 MYLIB 中)的 5 位数字，两位小数，初值为 12.39 的数据区

说明名为&CLVAR2 长为 5 位数字 1 位小数的变量

把 DA2 的内容复制到&CLVAR2 中

命令如下：

```
CRTDTAARA DTAARA(MYLIB/DA2) TYPE(*DEC) LEN(5 2) VALUE(12.39)
.
.
.
DCL  VAR(&CLVAR2) TYPE(*DEC) LEN(5 1)
RTVDTAARA DTAARA(MYLIB/DA2) RTNVAR(&CLVAR2)
```

现在&CLVAR2 中内容为 0012.3。

3.6.11　修改和取得数据区的例子

下例是用 CHGDTAARA 和 RTVDTAARA 命令做字符子串操作的例子。做法如下：

生成一个名为 DA1(放在 MYLIB 中)，10 字符长，初值为 ABCD5678IJ 的数据区

说明名为&CLVAR1，5 字符长的变量

把 DA1 中从第 5 位开始，5 位长的字符放到&CLVAR1 中

命令如下：

```
DCL VAR(&CLVAR1) TYPE(*CHAR) LEN(5)
.
CRTDTAARA DTAARA(MYLIB/DA1) TYPE(*CHAR) LEN(10) +
     VALUE('ABCD5678IJ')
.
.
.
CHGDTAARA DTAARA((MYLIB/DA1) (5 4)) VALUE('EFG')
RTVDTAARA DTAARA((MYLIB/DA1) (5 5)) RTNVAR(&CLVAR1)
```

现在&CLVAR1 的内容为‘EFG I’。

`第四章　目标和库

目标是命令操作的基本工作单元，例如，程序和文件都是目标，通过目标可以查找、管理和处理系统中的数据。仅要知道所用目标和功能即可，不必清楚它所在的存储地址。

注：目标可放在库中和目录中。（以前说目标只能放在库中）这样的内容是有关放在库中目标的信息，目录的有关内容请看集成文件系统介绍一书。

4.1 目标类型和一般属性

每类目标在系统中都有唯一的用途以及处理这类目标的一系列命令。每类命令都有一些一般属性用来说明目标，这些一般属性在 4.7 表中列出。DSPOBJD 的联机帮助信息也显示这些属性的说明。

4.2 目标完成的功能

目标可以完成多种功能，某些功能是系统自动完成的有些是通过命令完成的。

4.2.1 系统自动完成的功能

自动完成的功能保证用一致的、安全的和恰当的方法处理目标。

这些功能是：

目标类型验证：系统检查目标类型和在目标上做的功能类型以便验证在些目标上是否能做此种处理。例如，在 CALL 命令中规定的不是程序类型，则不能完成调用功能。

目标权限验证：系统检查目标、功能及用户以便验证用户是否用此目标完成功能。例如，USERA 没有权力使用 OBJB，所以他不能请求使用此目标完成任何操作。

损坏目标的检查和通知：系统监控在处理目标时产生的错误并通知你由于不认识目标产生的结果导致的失败。它是通过标准的信息来指出目标已损坏。这种情况发生的比较少，系统能整体性的监视和通知这类故障。

4.2.2 用命令完成的功能

有二类功能是要通过命令来请求的：

为每类目标规定功能，例如，生成、修改和显示

下表给出的一般功能

表 4－1 目标的一般功能

功能	标题
检索库中的一个或多个目标	4.3.4
规定库中目标的权限	4.4.6
说明目标	4.6
显示目标说明	4.7
取得目标说明	4.8
删除系统中无用的目标	4.10
在库间移动目标	4.11
生成重复目标	4.12
目标改名	4.13

删除目标	4.15
分配和再分配目标	4.16
显示目标锁状态	4.16.1
检验目标是否存在	5.1.2

4.3 库

在 AS/400 系统中，目标组成一个叫做库的特别目标，目标是用库来查找的。要访问库中的目标，必须对库和目标有权限，详细内容请看 4.4 和 4.4.3。

如果在同一参数中规定库名做为目标名，这种目标名叫限定名，此时，目标名要这样写：

DISTLIB/ORD040C　　程序 ORD040C 是在 DISTLIB 中。

如果对命令使用提示，而提示的是限定名，那么要给出库名和目标名。在大多数命令中，可以规定一个库名、*CURLIB（作业的当前库）或使用库列表。

4.3.1 库列表

对命令可规定限定名，不能省略库名，如果省略了，发生下列情况：

对生成命令，生成的目标会根据目标类型放在用户的当前库，*CURLIB 中或放在系统库中。例如，程序会放在当前库中，授权表会放在 QSYS 库中。

对不是生成命令的其它命令，系统会用列表来查找目标。AS/400 系统库列表由四部分组成：

系统部分：它包括系统需要的库的列表。

产品库：它包括两个产品库。系统产品库支持与库（不是 QSYS）有关的语言及实用程序来处理它们的命令。

用户命令和菜单也能在 CRTCMD 和 CRTMNU 命令的 PRDLIB 参数中规定产品库来保证找到有关的目标。

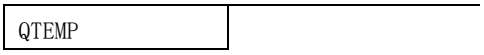
产品库是由系统管理的。在需要时，会自动把产品库（例如 QRPG）放在库列表的保留产品库位置上。产品库可以是当前库的复本或者是库列表中用户部分的一个库。

当前库：当前库可以是但不一定非得是库列表中某一库的复本。在大多数命令时可以用*CURLIB 做为一个库名而不管作业是否规定了当前库。如果库列表中没有当前库而规定了*CURLIB，就用 QGPL 库，可用 CHGCURLIB 或 CHGLIBL 命令来修改作业的当前库。

用户部分：它包括系统用户和应用程序使用的库。库列表中的用户库，产品库和当前库对每个作业可以有不同，但最多 25 个库。

下图给出列表的结构：

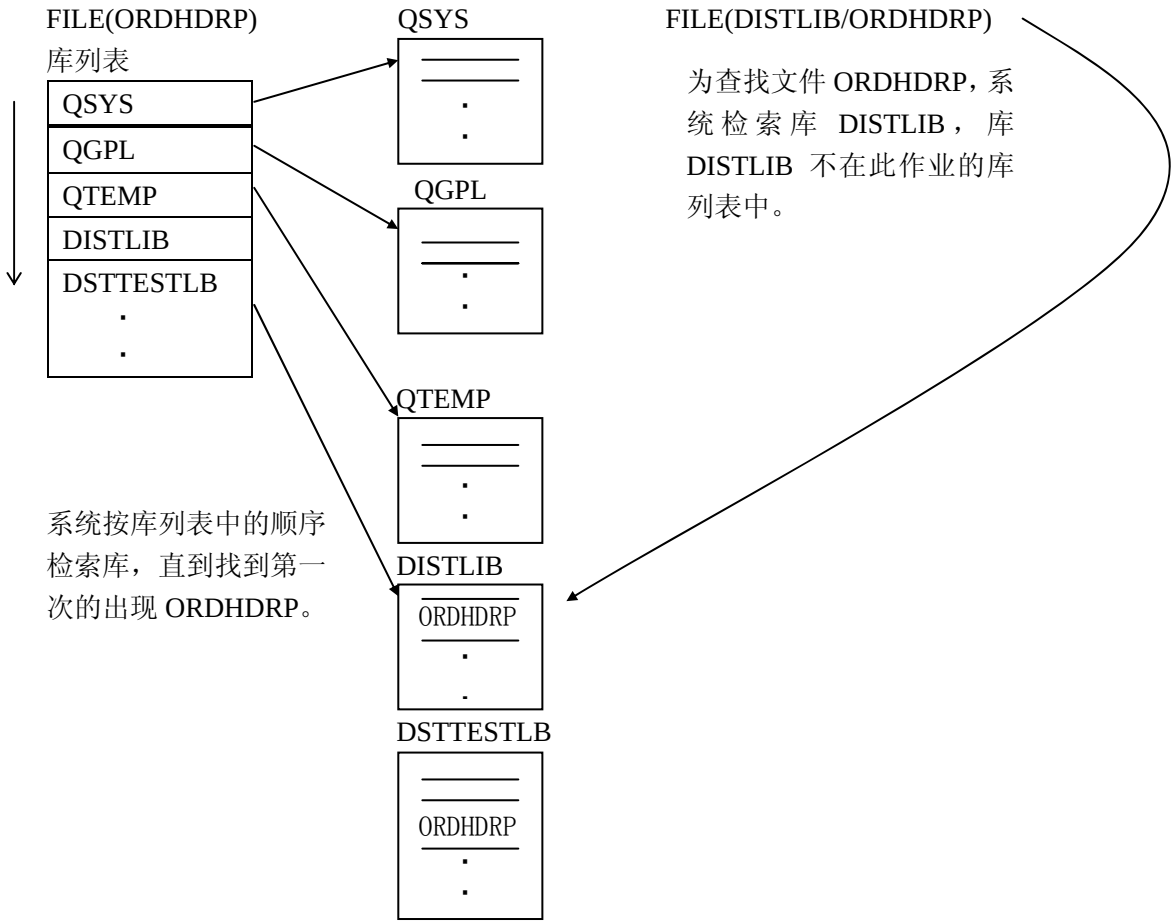
QSYS QSYS2 QUSRSYS QHLPSYS	系统部分
QPDA	产品库 1
QRPG	产品库 2
OELIB	当前库
OELIB QGPL	用户部分



注：在使用 SEU 时，系统把 QPDA 放在产品库 1 中。当用 SEU 做语法检查时，第二个库加到产品库 2 中，例如，要检查 RPG 源码的语法，则 QPDA 是产品库 1，QRPG 是产品库 2。对大多数其它系统功能，不用产品库 2。

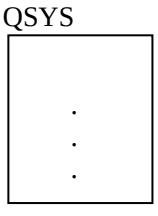
用库列表很容易查找系统中的目标，每个作业都有与之有关的库列表。在用库列表找目标时，按库在库列表中的顺序查找，直到找到一个名字与类型都相符的目标为止。如果在库列表中有多个同名同类型的目标存在，那么在库列表中找到的目标第一次出现的库即为要查找的目标所在库。

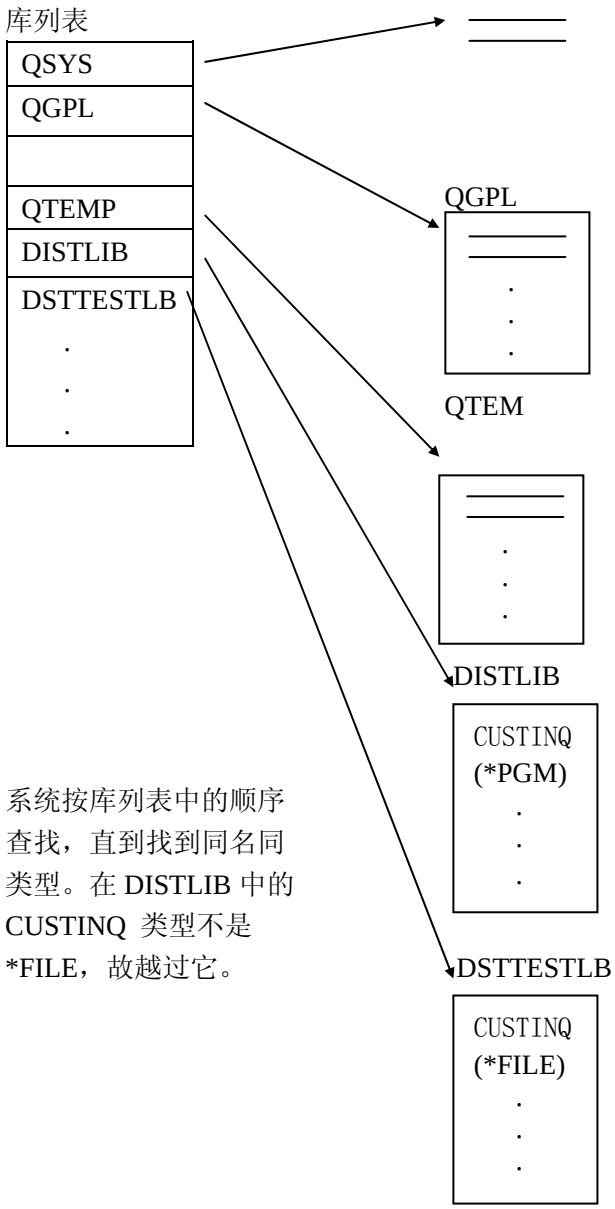
下表给出使用库列表（*LIBL）和规定库名时查找库的情况：



下图给出在库列表中的同名但不同类型的二个目标。系统用下列命令检索：

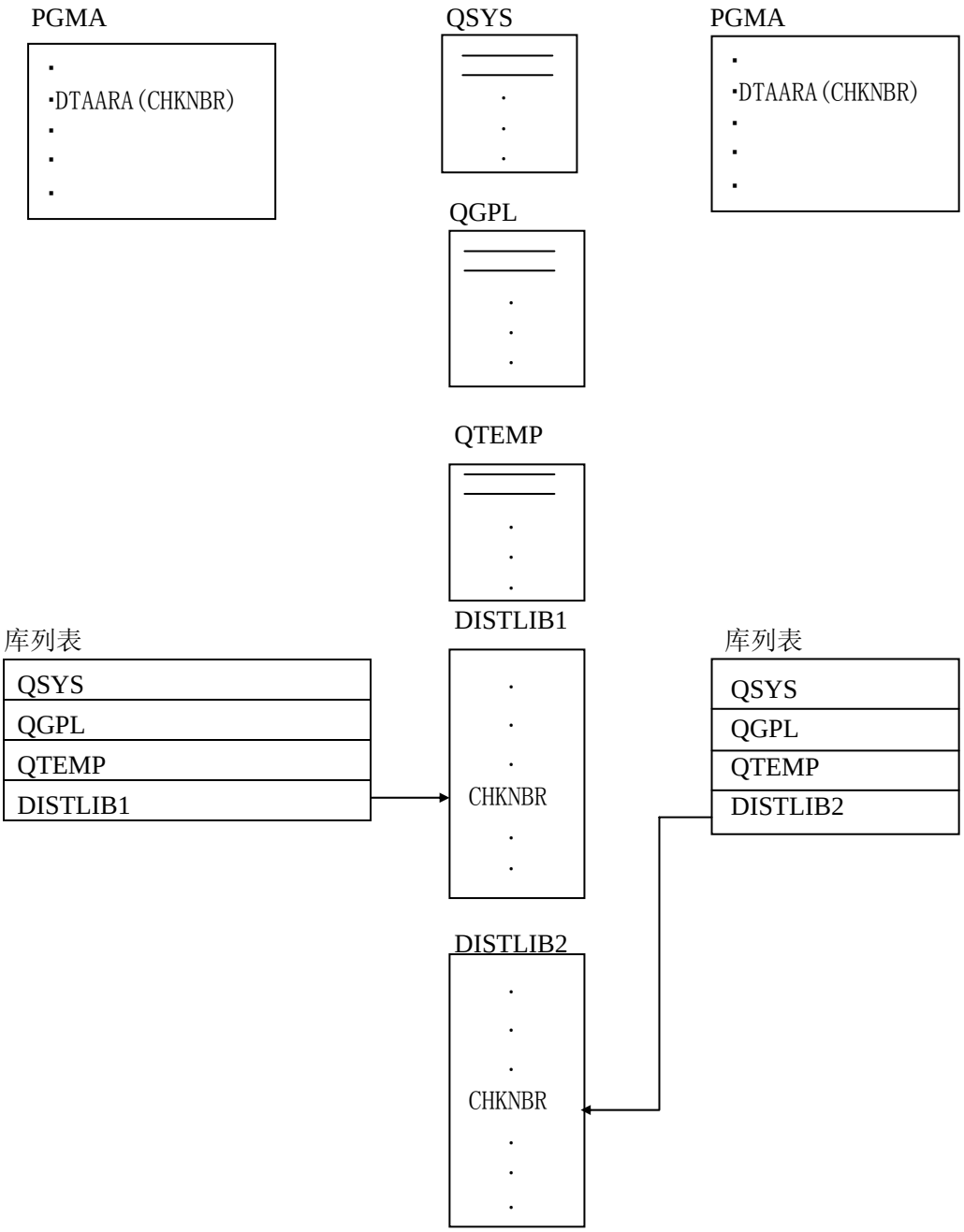
```
DSPOBJD    OBJ(*LIBL/CUSTINQ)    OBJTYPE(*FILE)
```





一般说来，使用库列表比使用限定名更灵活、容易，它的优点不仅在于不用输入库名，更表现在如应用程序使用不同数据，可使用不同库列表而不用修改应用程序本身。例如，CL 程序 PGMA 修改数据区 CHKNBR。如果没规定库名，那么根据使用的库列表不同而修改不同的数据区。如下图，JOBA 和 JOBB 都调用 PGMA：

JOBA 库 JOBB



在下列情况下，使用限定名要更好些：

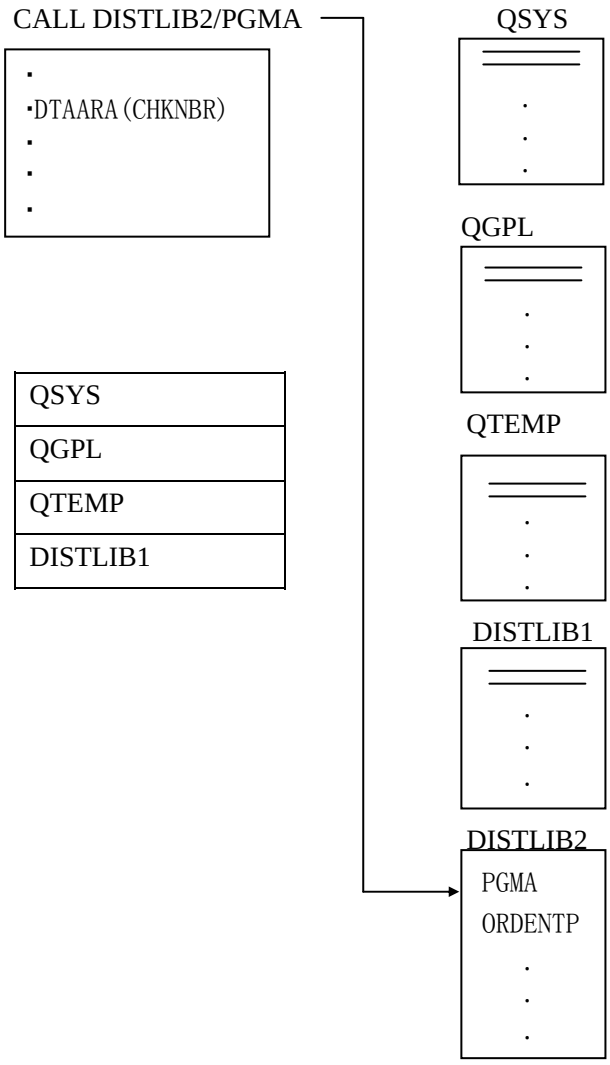
作业使用的目标不在库列表中

在库列表中有多个同名目标，但你只用其中一个库中的目标

由某些原因只使用一个库中的目标

如果用限定名调用程序，但程序打开的文件不是用限定名，这时文件如不在库列表中，则文件打不开。如下图：

JOBA　　库列表



对 PGMA 的调用是成功的。因为在 CALL 中的程序名用限定名，但在程序要打开文件 ORDENTP 时，由于用的不是限定名，而文件又不在库列表中的某一库里，故打开操作失败。如果把 DISTLIB2 加到库列表中或文件用限定名，则能打开文件，某些高级语言不允许用限定名，可用 OVRxxx 命令来规定限定名。

4.3.1.1　作业的库列表

每个作业的库列表由四部分组成：系统部分，用户部分，当前库，产品库。系统部分总是在库列表中。

初始的系统值 QSYSLIBL 中的库组成库列表的系统部分。它的初值为 QSYS，QSYS2，QHLPSYS 和 QULRSYS。系统值 QUSRCIBL 中的库组成库列表中的用户部分。

QSYSLIBL 中可以有 15 个库，QUSRLIBL 可有 25 个库。要修改作业库列表的系统部分，使用 CHGSYSLIBL 命令。要修改 QSYSLIBL 或 QUSRLIBL 的值，用 CHGSYSVAL 命令。在系统值修改后会影响新的作业。

4.3.1.2　修改库列表

对正在运行的作业，可用 ADDLIBLE 和 RMVLIBLE 命令来往库列表中加库或从库列表移出一个库，也可用 CHGLIBL 或 EDTLIBL 命令来修改库列表。这些命令只能改库列表

中的用户部分，不会改系统部分。可用 CHGURLIB 或 CHGLIBL 命令修改当前库，当前库也可在用户配置文件中，在注册用户时，或用 SBMJOB 命令来修改。产品库不能用 CL 命令来增加，这些库是使用它们时用命令或菜单加上去的。它们也不能用 CL 命令修改，但能用修改库列表 API（QLICHGLL）来修改。

在使用这些命令时，它仅影响使用命令的作业，这些修改仅在作业运行期间有效。在用这些命令修改库列表时，库必须已经存在。活动的用户库列表中的库不能被删除。

在作业启动时，库列表中的用户部分由作业描述中的值或在 SMBJOB 命令中规定的值来确定，可以规定*SYSVAL，它使 QUSRLIBL 的值成为作业库列表中的用户部分。如果在作业描述、BCHJOB 或 SBMJOB 都规定了库名，那么用 BCHJOB 或 SBMJOB 中规定的值。下面是一些规则：

可在作业描述中规定库列表。当作业运行时，可替代 QUSRCIBL 中规定的库列表。

在用 SBMJOB 或 BCHJOB 提交作业时，可在命令中规定库列表。它替代作业描述或 QUSRLIBL 中规定的库列表。

在 SBMJOB 命令中，可规定*CURRENT。它指出使用发出命令的作业库列表。

在作业内可使用 ADDLIBLE、RMVLIBLE 或 CHGLIBL 命令，这些命令替代以前规定的库列表。

可用 CHGCURLIB 或 CHGLIBL 命令来修改当前库。

可用下列 CL 程序来修改库列表：

```
PGM  /* SETLIBL - Set library list */
CHGLIBL LIBL(APPDEVLIB QGPL QTEMP)
ENDPGM
```

如果正常使用库列表，可用一个初始程序来建立库列表：

```
PGM  /* Initial program for QPGMR */
CHGLIBL LIBL(APPDEVLIB QGPL QTEMP)
TFRCTL PGM(QPGMMENU)
ENDPGM
```

程序生成时它的用户配置文件改为规定新的初始程序。控制从这个程序传给 QPGMMENU，它显示程序员菜单。要往初始程序规定的库列表中加一个库，可用以下命令：

```
ADDLIBLE LIB(JONES) POSITION(*LAST)
```

如果作业中有些部分要用不同的库列表，可以写一个 CL 程序保存当前库列表，以后再重存回来。程序如下：

```
          PGM
          DCL &LIBL *CHAR 275
          DCL &CMD *CHAR 285
(1)       RTVJOBA USRLIBL(&LIBL)
```

```
(2)     CHGLIBL (QGPL QTEMP)
        .
        .
        .
(3)     CHGVAR &CMD ('CHGLIBL (' *CAT &LIBL *TCAT ')')
(4)     CALL QCMDEXC (&CMD 285)
        .
        .
        .
        ENDPGM
```

⑴保存库列表的命令，它存在变量&LIBL 中，每个库名占 10 个字节，库名之间要有一个空格。
⑵此命令根据下面的功能要求修改库列表。
⑶CHGVAR 在变量&CMD 中建立命令 CHGLIBL。
⑷调用 QCMDEXC 来处理&CMD 中的命令串。在调用 QCMDEXC 之前要用 CHGVAR 命令。这是因为在 CALL 命令中不能做连接动作。

4.3.1.3 设置库列表的考虑

设置库列表和使用它时，要考虑以下问题：

在库列表中的库必须在系统中存在，在系统启动后可以访问 QUSRLIBL 和 QSYSLIBL。如果库列表中的库不存在，则有信息送往 QSYSOPR，且忽略此库。一旦 OS/400 启动后，任何活动作业的库列表中的库都不可以删除。如果在作业描述、BCHJOB 或 SBMJOB 中规定的库列表有不存在的库，则作业不能启动。

库列表中的库必须为使用它们的用户授权。要初始一个库列表，用户必须对这些库有目标操作权限，在增加库和修改库时对库要有*USE 权限。

当一个在授权用户配置文件下运行的程序要往库列表中加一个库，而当前用户在结束程序运行前又不能增加或移出库，那么用户保持*USE 授权来访问库，这种情况仅对规定了*LIBL 时才有效。

库列表尽可能短，这样存好的系统性能。

4.3.2 显示库列表

可用 DSPLIBL 来显示运行作业的库列表。它按库列表中库出现的同样顺序显示，也可用 DSPJOB 命令后选 13 来显示库列表。

4.3.3 使用类属目标名

有时可能需要检索一些名字有相同字符的目标，这类检索叫做类属检索。此时在命令的目标名中写类属名，类属名用*号加一些字符组成。例如，要显示名字用 ORD 开始的目标描述，那么在 DSPJOBD 中用类属名 ORD*。

类属检索，可由下列库限定目标类属名：

某个库：请求的操作仅在规定库的类属名目标上完成。

作业库列表：按库列表中给出的顺序检索库，请求的操作在库列表中库的类

属名目标上完成。

作业当前库：检索作业的当前库。如果没规定，用 QGPL。

库列表中用户部分的所有库：按库列表中列出的顺序检索，其中包括当前库、

有权使用的所有用户库，以及下列以 Q 打头的库都可被检索：

QDSNX
QGPL
QGPL38
QMQMDATA
QMQMPROC
QPFRDATA
QRCL
QS36F
QUSRVxRxMx
QUSER38
QUSRADSM
QUSRBRM
QUSRIJS
QUSRINFSKR
QUSRNOTES
QUSRRDARS
QUSRSYS

系统中你有权使用的所有库：以字母顺序检索。

4.3.4 检索多个或一个目标

在能用类属名的所有命令中，可规定目标名（不用*号）且可检索多个目标。如果规定目标名和*ALL 或*ALLUSR 做库名，系统要检索多个目标，检索结果是你有权的目标名及类型。如果规定类属名，或规定*ALL，*ALLUSR 或库名，可以规定所有支持的目标类型（或*n ALL 目标类型）。

4.4 使用库

用库来把相关目标组合在一起，且用名字来查找目标，一个库是一组目标的目录。

可以使用库来做：

以每个用户的目标分组，这有助于管理系统中的目标。例如，可把用户 JOE 的目标放在 JOELIB 库中。

以每个应用程序所用的目标分组。例如，可把所有定货单文件和程序放在 DISTLIB 中，只要把一个库加到库列表中就能够保证所有定货单文件和程序都在列表中。这样，就不用在每次使用文件和程序时都规定库名。

保证安全，可以规定哪个用户有权使用哪些库，做什么事情。对新生成的目标根据所在库的 CRTAUT 参数，使用自动授权表和分配授权可以使安全措施更容易实现。对这些目标根据 CRTOBJAUD 参数值来查它们的属性。

把要同时保存和重存的目标都放在同一库里，能简化保存/重存操作，可用一个 SAVLIB 来代替多个 SAVOBJ 命令。

使用多个库做测试，详细内容请看附录 A。

使用多个产品库，例如，源文件和某些目标用一个库，程序及文件用一个库，不要经常做保存的目标用一个库，经常要保存的目标用一个库。

用多个库能更容易的使用目标。例如，有两个同名文件放在不同库中，一个用来测试，一个正常使用。只要在程序中不规定库名，就不用对测试和正常使用来修改程序中的文件名，

而用库列表来控制使用哪个库。

有两种类型的库：产品库和测试库。产品库是正常使用的库，在调试方式中，你要注意保护产品库的数据文件不被修改，而在测试库中的文件不用特别规定即可修改。（详细内容请看附录 A）。

4.4.1 生成一个库

用 CRTLIB 命令生成一个库，下面的命令生成一个名为 DISTLIB 的库，它是一个产品库，公共权限为缺省值，它不允许其它用户访问这个库，放这个库中的目标给出缺省的权限为*CHANGE。

```
CRTLIB   LIB(DISTLIB) TYPE(*PROD) CRTAUT(*CHANGE) CRTOBJAUD(*USRPRF) +
         AUT(*EXCLUDE) TEXT('Distribution library')
```

用户不能生成名字以 Q 开头的库，在检索时，会认为此类库为系统库。（例如 QRPG、QPDA 等）。

4.4.2 规定库的权限

下面介绍可以分配给用户使用库的各种权限，详细内容请看安全参考一书。

4.4.2.1 目标权限

一个库的目标操作权给用户显示库描述的权限。

一个库的目标管理权包括下列授权：

分配和取消授权，仅能分配和取消你管理范围内的权限，仅目标主人和有*ALLOBJ 权限的人能分配库的目标管理权限

给库改名

目标存在权和使用权给用户删除目标的权力。

目标存在权和操作权给用户转移库所有者的权力。

4.4.2.2 数据权限

库的加和读权限允许用户在库中生成新目标和往库中移进目标。

库的修改和执行权限允许用户修改库中目标的名字，也授权给有关的用户。

删除权限允许用户从目标中取消一项，库的删除权限不是允许用户删除库中目标，用库中目标的权限来确定目标是否能删除。

执行权允许用户检索库中目标。

4.4.2.3 合并授权

库的*USE 权限（有目标的操作权、读和执行权）包括下列权限：

用库来查找目标

显示库的内容

把库放到库列表中

保存库

从库中删除目标

库的*CHANGE 权限（由目标操作权和所有数据权限组成）包括下列权限：

用库查找目标

显示库的内容
把库放到库列表中
保存一个库
从库中删除目标
往库中加目标

*ALL 权限提供所有目标和数据授权。用户可以删除库，规定库的安全措施，修改库，显示库的描述和内容。

*EXCLUDE 权限不让用户访问目标。

要显示一个库的授权情况，用 DSPOBJAUT 命令。

4.4.3 目标的安全考虑

当系统访问你要引用的目标时，要检索你是否有权使用这个目标以及请求使用它做什么。一般说，有两层授权要考虑：

必须有权用此目标来完成请求的操作。

必须对目标所在的库有权限，如果用库列表，必须对表中的库有权限。

目标权限是由系统的安全功能控制的，它包括：

目标主人和有*ALLOBJ 权限的用户对目标有所有的权限，也能给其它用户分配和取消目标的权限。

在没有分配目标的私人授权时，用户有公共权限。

详细情况请看安全考虑一书，写程序时有关安全方面的考虑也请看此书。

4.4.4 新建目标的缺省公共授权

在库中建目标时，目标的公共授权是由库的 CRTAUT 值设置的：

```
CRTLIB    LIB(TESTLIB) CRTAUT(*USE) AUT(*LIBCRTAUT)
```

它生成了 TESTLIB 库，那么生成放在库中的所有目标都有缺省的公共权限*USE。TESTLIB 的公共授权是由库 QSYS 的 CRTAUT 值确定的。

```
  CRTDTAARA DTAARA(TESTLIB/DTA1) TYPE(*CHAR) +
            AUT(*LIBCRTAUT)

 CRTDTAARA DTAARA(TESTLIB/DTA2) TYPE(*CHAR) +
            AUT(*EXCLUDE)
```

数据区 DAT1 生成在 TESTLIB 中，它的授权是*USE。

数据区 DAT2 生成在 TESTLIB 中，它的授权是*EXCLUDE，这是因为在生成它的命令中已规定 AUT 参数。

可用授权表来实现库中目标的安全：

```
 CRTAUTL AUTL(PAYROLL)
 CRTLIB  LIB(PAYLIB) CRTAUT(PAYROLL) +
          AUT(*EXCLUDE)
```

它生成名为 PAYROLL 的授权表。库 PAYLIB 生成时的公共授权为*EXCLUDE，生成在 PAYLIB 中的目标的安全由缺省规定，是由授权表 PAYROLL 管理的。

```
CRTPF FILE(PAYLIB/PAYFILE) +
      AUT(*LIBCRTAUT)

CRTPF FILE(PAYLIB/PAYACC) +
      AUT(*CHANGE)
```

文件 PAYFILE 生成在 PAYLIB 中，它是由 PAYROLL 管理权限的。在 CRTPF 中对 PAYFILE 规定的公共授权为*AUTL，它指出由授权表 PAYROLL 管理其公共权限。

文件 PAYALL 生成在 PAYLIB 中，由于在 CRTPF 命令中规定了 AUT 参数，故它的公共权限为*CHANGE。

库中的 CRTAUT 值规定了生成在此库中的目标的公共权限。这些值可以是：

*SYSVAL（目标权限由系统值 QCRTAUT 确定），授权表名，*ALL，*CHANGE

*USE，*EXCLUDE。

4.4.5 新建目标的缺省审查属性

在库中生成目标时，目标的审查属性是由库的 CRTOJBAUD 值来设置的。

```
CRTLIB LIB(PAYROLL) AUT(*EXCLUDE) CRTAUT(*EXCLUDE) CRTOBJAUD(*ALL)
```

在库 PAYROLL 中的所有目标在读和修改时都要审查它的权限。

4.4.6 往库中放目标

生成目标时，要放在库里。如果没指定库，放在作业的当前库中，如没有当前库，放在 QGPL 中。在生成库时可用 CRTLIB 中的 CRTAUT 参数来规定生成在此库中的目标的公共权限。一个目标的限定名包括库名和目标名。用下列命令生成物理文件 ORDHDRP 放在 DISTLIB 中：

```
CRTPF      FILE(DISTLIB/ORDHDRP)
```

要能把文件放到库中，必须对库有读和加的权限。

同名同类型的目标不能放在同一库中。如果这样做，系统要给出错读信息。

注：QSYS 库仅用于系统目标，不要把其它的特许程序放到这个库中，因为在安装新目的 OS/400 时会丢失一些内容。

4.4.7 删除和清理库

用 DLTLIB 删除库时，删除了库中目标和库本身。用 CLRLIB 清理库时，删除了库中目标但不删除库。

要删除一个库，必须对库和目标有存在权，对库要有使用权。如果对库中所有目标没有存在权，那么不能删除无权限的目标和库，只删除那些你有权限的目标。如果对库没有存在权而要删除它，不仅是库删不掉，库中目标也一个也删不掉。如果要删除某些你有权的目标，可用删除这类目标的命令，比如 DLTPGM。

不能删除在活动作业库列表中的库，必须等到作业结束才允许删库。这样，必须在下个例程里开始之前删库。

如果一个库是初始库列表（由系统值 QSYSLIBL 和 QUSRLIBL 确定）中的一个库，要按下列步骤删除它：

1. 用 CHGSYSVAL 把库从库列表中移出。
2. 用 CHGLIBL 来修改作业的库列表，也可用 CHGSYSLIBL，ADDLIBLE，EDTLIBL，RMVLIBLE 命令来修改库列表。
3. 用 DLTLIB 命令删除库及目标。

注：不能删除 QSYS 和 QGPL。你不能使用 QRECOVERY 库，它仅是给系统用的。

要清理一个库，对库中目标要有存在权，对库要有使用仅。如果不是这样，则删不掉无权的目标。如果目标已分配给其它人，则删不掉它。

4.4.8 显示库名及内容

可用 DSPLIB 或 WRKLIB 命令来显示或打印你有权的库来找到库中每个目标的基本信息。它包括：

目标名和类型
目标属性
目标大小
目标生成时给出的说明

在 DSPLIB 中，可规定一个库名或多个库名，此时，目标是按库分组的。在每个库中，目标按类型分组，在同类型时，按名字的字母顺序列出，库的顺序为下列之一：

如果在 DSPLIB 中规定库名，按命令中规定库的顺序显示。

如果在 DSPLIB 中规定*LIBL 或*USRLIBL，按作业库列表中的顺序显示。

如果规定*ALL 或*ALLUSR，则以字母顺序显示，对显示的库必须有读的权限。

下列命令显示库 DISTLIB 中的目标：

```
DSPLIB  LIB(DISTLIB)  QUTPUT(*)
```

参数 QUTPUT 中的*号表示，显示的结果在交互处理时在工作站上显示，在批处理时要打印出来。要在交互处理时打印出来，规定 QUTPUT(*PRINT)。

4.4.9 显示和获得库的说明

用 DSPLIBD 和 RTVLIBD 命令来显示和获得库的说明。

库的说明信息包括：

库类型（PROD 或 TEST）
库存的辅助存储池（ASP）
库的生成权限
库的生成目标审查
库的说明

4.5 OS/400 图象民族支持

OS/400 特许程序支持不同的民族语言，这就允许在系统中对不同用户表示不同的语言。

用于用户可读信息的语言（显示、信息、打印、联机信息）是由作业的库列表控制的，往库列表中的系统部分加一个民族语言库，可表示不同的民族语言版本，主语言是支持安装

的每个特许程序运行码和原文数据的民族语言版本。第二语言，是所有特许程序的原文数据。

系统的主语言信息是存在与 IBM 特许程序同一库中。例如，系统的主语言是英语，则 QSYS、QHLPSYS、QSSP 就包含英文的信息。QSYS 和 QHLPSYS 是在库列表的系统部分，而其它特许程序库是在需要时由系统加到库列表中去的。

不是主语言的民族语言版本是安装在第二语言库中的。每个库中包括一个民族语言版本，帮助所有的特许程序。第二语言库的名字为 QSYSnnnn，其中 nnnn 是语言特性码。如法语的特性码为 2928，则第二语言库为 QSYS2928。

如用户要表达系统的主语言信息，则不要特别的操作。如果要用不同主语言的第二语言，则必须修改库列表，把要用的语言库放在库列表中所有民族语言信息的前边。可用下列方法之一把要用的民族语言库放在首位：

可用 CRTSBSD 和 CHGSBSD 的 SYSLIBLE 参数来规定显示信息等所用的语言：

```
CRTSBSD SBSD(QSBSD 2928) POOLS((1 *NOTSG)) SYSLIBLE(QSYS2928)

CHGSYSLIBL  LIB(QSYS2928)
```

可用 CHGSYSLIBL 中的 LIB 参数把语言库放在库列表的顶部：

```
CHGSYSLIBL  LIB(QSYS2928)
```

可在用户配置文件中建一个初始程序来规定语言库放在交互作业的库列表顶部，如果用户不想在每次注册时都用 CHGSYSLIBL 命令，这是一个好的做法。初始程序使用 CHGSYSLIBL 把要用的语言库加到库列表的顶部。

注：不是所有的用户都有权使用 CHGSYSLIBL 命令。

要使一个用户在没被授权的情况下使用 CHGSYSLIBL 命令，可以写一个包括这条命令的 CL 程序。此程序的主人为超级用户，且在生成时用超级用户的权限，有权运行这个程序的用户就能用这条命令。下面是一个例子，它为用户 Frence 设置库列表。

```
PGM
  CHGSYSLIBL  LIB(QSYS2928)  /* Use French information */
ENDPGM
```

4.6 说明目标

在用生成命令建立目标时，可在 TEXT 参数中给出不多于 50 个字符的说明。某些命令会允许使用缺省值*SRCMBRTXT，它指出生成目标的说明就用生成它的源成员的说明，这仅适用于用源文件生成的目标。

如果生成命令的源输入是设备文件或联机文件或不用源文件，缺省值为空白。它可以用 DSPOBJD 和 DSPLIB 命令显示说明，可用 CHGOBJD 或其它很多 CHGxxx 来修改。

4.7 显示目标说明

可用 DSPOBJD 或 WRKOBJ 命令来显示目标说明，它可用来确定系统中哪些目标没有被使用。如果用批处理，说明可以打印或写到一个数据库文件中。如果用交互处理，说明可以显示，打印或写到一个数据库文件中。

可以显示目标的全部、基本或服务属性，它们在下表中给出。

基本属性	全属性	服务属性（见注）
目标名	目标名	目标名
库名	库名	库名
目标类型	目标类型	目标类型
扩充属性	主人	源文件和库
目标大小	主组	成员名
说明（部分）	扩充属性	扩充属性
	用户定义的属性	用户定义的属性
	说明	空闲状态
	生成日期和时间	目标大小
	生成目标的用户	生成日期和时间
	生成目标的系统	文件成员最后修改的日期和时间
	目标定义域	系统级别
	修改日期和时间	编译程序
	是否使用数据采集	目标
	最后使用日期	控制级别
	使用天数计数	由程序修改
	允许由系统修改	是否由用户修改
	目标审查值	特许程序
	目标大小	PTF 号码
	脱机尺寸	APAR ID
	空闲状态	说明
	压缩状态	目标或目标状态说明
	辅助存储池	条件
	目标溢出	
	保存的日期和时间	
	重存的日期和时间	
	保存命令	
	设备类型	

注：

1.由编程支持人员所用的服务信息用来确定生成目标所用系统的级别以及目标是否修改过，这些信息是很有用的，因为它能给出生成目标所用的源文件以及最后修改日期。

2.库中目标仅给出目标名，如果在 DSPOBJD 中用*LIB，目标大小的信息是库的大小，而不是库中所有目标的总尺寸，可用 QLIRLIBD 或 DSPLIB OUTPUT(*PRINT)来查到库的总尺寸。用 DSPOBJD 或 WRKOBJ 命令，可用下列内容列出库中目标：

目标、类属性、类型、一类目标中的名字或类属性

目标以库、类型的字母顺序列出。如果要显示多个目标的全属性或服务属性，可在批作业中用 DSPOBJD 命令，可以输出到假脱机上打印出来，或形成一个数据库文件。如果直接输出到数据库文件，目标的所有属性也写进文件中，可用 DSPFFD 命令来显示文件

QADSPOBJ，来看这个文件的记录格式。

下列命令显示名字以 QRD 开头的目标说明。

```
DSPOBJD    OBJ(DISTLIB/ORD*)   OBJTYPE(*FILE) +
           DETAIL(*BASIC)   OUTPUT(*)
```

显示的基本内容为：

```
                        Display Object Description - Basic
                                                                  Library 1 of 1
Library:     DISTLIB

Type options, press Enter.
  5=Display full attributes   8=Display service attributes

Opt  Object      Type      Attribute                 Size   Text
 _   ORDDTLP     *FILE        PF                     8192   Order detail
 _   ORDHDRP     *FILE        PF                     8192   Order header




                                                                  Bottom
F3=Exit   F12=Cancel   F17=Top   F18=Bottom
```

如果规定*FULL 而不是*BASIC，或在 ORDDTLP 的基本显示前写 5，则显示全属性。

```
                        Display Object Description - Full
                                                                  Library 1 of 1
Object . . . . . :    ORDDTLP        Attribute . . . . . :   PF
  Library  . . . . . :     DISTLIB     Owner  . . . . . . . :   QSECOFR
Type . . . . . :    *FILE          Primary group       :   *NONE
User-defined information:
   Attribute . . . . . . . . . . . . :
   Text  . . . . . . . . . . . . . . :
Creation information:
  Creation date and time . . . . . :   06/08/89  10:17:03
  Created by user  . . . . . . . . . :   QSECOFR
```

```
  System created on  . . . . . :   SYSTEM01
  Object domain  . . . . . :   *SYSTEM
Change/Usage information:
  Change date/time . . . . . :   05/11/90  10:03:02
  Usage data collected . . . . . :   YES
  Date last used . . . . . :   05/11/90
  Days used count  . . . . . :   20
  Date use count reset . . . . . :   03/10/90
  Allow change by program  . . . . . :   YES
Auditing information:
  Object auditing value  . . . . . :   *NONE
Press Enter to continue.
F3=Exit   F12=Cancel                    (C) COPYRIGHT IBM CORP. 1980, 1993.
```

```
                    Display Object Description - Full
                                                                  Library 1 of 1
 Object . . . . . :    ORDDTLP        Attribute  . . . . . :   PF
   Library  . . . . . :      DISTLIB      Owner  . . . . . . . :   QSECOFR
 Type . . . . . :    *FILE

 Storage information:
   Size . . . . . :   8192
   Offline size . . . . . :   0
   Freed  . . . . . :   NO
   Compressed . . . . . :   NO
   Auxiliary storage pool . . . . . :   1
     Object overflowed  . . . . . :   NO
 Save/Restore information:
   Save date/time . . . . . :
   Restore date/time  . . . . . :
   Save command . . . . . :
   Device type  . . . . . :
                                                                      Bottom
 Press Enter to continue.


 F3=Exit   F12=Cancel
                                        (C) COPYRIGHT IBM CORP. 1980, 1993.
```

4.8　得到目标说明

可用 RTVOBJ 命令把目标的说明返回给一个 CL 过程，用变量来返回说明，可用这些说明来确定系统中已用的目标。

返给做为变量的目标说明可以是：

目标所在库名
目标的扩展属性
用户定义的属性
描述说明
目标主人的用户配置文件名
目标主组的名字
ASP　ID
是否溢出所在 ASP
目标生成日期和时间
目标最后保存日期和时间
在 SAVACT(*LIB　*SYSDFN 或*YES)保存操作期间目标最后保存的日期和时间
目标最后重存的日期和时间
生成目标的系统
目标定义域
是否使用数据采集
目标最后使用日期
目标使用的天数
最后计数清零的日期
目标数据的存储状态
目标压缩状态
目标的字节数
最后保存时目标的字节数
保存目标使用的命令
保存目标到带上时的顺序号
保存目标的带或软盘的卷标
保存目标用的设备类型
保存目标使用的备份文件名
保存目标使用的备份文件所在库名
保存目标使用的文件标号
用来生成目标的源文件名
用来生成目标的源文件所在库名
用来生成目标的原成员名
源文件的最后修改日期
特许程序标识、版本
生成目标的控制级
是否可用 QLICOBJD　API 修改的有关信息
指出是否用了 QLICOBJD　API 修改目标的有关信息
程序是否由用户修改过

如果得到的目标是特许程序的一部分，特许程序的名字及版本
生成得到目标的 PTF 号码
授权的程序分析报告（APAR）标识
目标跟踪的类型

4.8.1 RTVOBJ 的例子

在下面的 CL 过程中，用 RTVOBJD 得到目标的说明，假定 MOBJ 在 MYLIB 中：

```
DCL   &LIB       TYPE(*CHAR) LEN(10)
DCL   &CRTDATE   TYPE(*CHAR) LEN(13)
DCL   &USEDATE   TYPE(*CHAR) LEN(13)
DCL   &USECNT    TYPE(*DEC)  LEN(5 0)
DCL   &RESET     TYPE(*CHAR) LEN(13)
.
.
.
RTVOBJD     OBJ(MYLIB/MOBJ) OBJTYPE(*FILE) RTNLIB(&LIB)
            CRTDATE(&CRTDATE) USEDATE(&USEDATE)
            USECOUNT(&USECNT) RESETDATE(&RESET)
```

则下列信息返给程序：

MYLIB 放在&LIB 中
MOBJ 的生成日期放在&CRTDATE 中
MOBJ 的最后使用日期放在&USEDATE 中
使用 MOBJ 的天数放在&USECNT 中，开始计数的日期放在&RESET 中

4.9 目标的生成信息

下列信息是在目标生成时建立的一些信息，它对于目标管理和维护是很用用的。

目标的生成者：

— 生成目标的用户配置文件完成生成操作，如果此用户配置文件有成组配置文件，则组中的用户都拥有这个目标
— 当主人变更时，目标的生成者不能修改目标
— 在重存目标时，生成者是目标，所在介质上的生成者
— 在用 CRTDUPOBJ 命令建目标副本时，目标的生成者是运行此命令的用户
— IBM 支持目标的生成者为*IBM
— 在 V1.3 版本以前已有的目标，生成者为空白

生成目标的系统：

— 当重存目标时，生成系统为中间介质上目标所用系统
— 在 V1.3 版本以前已有的目标，生成系统为空白

4.10 删除系统中不用的目标

目标说明中的信息能帮助你确定及管理系统中不用的目标。要确定不用的目标，看最后使用日期和最后修改日期，修改命令不能改变最后使用日期。

最后修改的日期和时间：

— 在目标生成或修改时，系统会标记时期，指出发生修改的日期和时间

最后使用日期：

— 最后使用日期是在一天中的某个时候，（一天中的第一次使用）则用系统日期
— 不成功的企图使用一个目标，不改变最后使用日期
— 新目标的最后使用日期为空白
— 在重存期间删除和再生成的目标，丢失最后使用日期
— 成员数为零的数据库文件的最后使用日期不改变。例如，在用 CRTDUPOBJ 命令复制目标时，文件中无成员，则不改变最后使用日期
— 数据库文件的最后使用日期是最近使用的文件成员的日期
— 对逻辑文件，最后使用日期是最后一次使用逻辑成员的日期
— 对物理文件，最后使用日期是通过物理文件或逻辑文件访问得到数据空间数据的最后时间。

表 4-3 对不同类型目标引起最后使用日期更改的操作

目标类型	命令及操作
所有类型	CRTDUPOBJ 以及象 CPYLIB 之类的复制命令，GRTOBJAUT 命令
修改请求说明	CHGCMDCRQA
图表格式	DSPCHT
C 场所说明	RTVCLDSRC 或在 C 程序中引用
级别	用于启动一个作业
命令	运行，在 CL 程序编译时，对 SEU 提示时，在检查方式下调系统时
通讯边界信息(CSI)	当调用 CMINIT 来从边界信息目标初始化为不同转换字符值时
连接列表	当连接列表到达 Vary on pending 状态边界时
交叉系统产品映象	在 CSP 应用程序中引用时
控制器描述	在控制器到达 Vary on pending 状态边界时
设备描述	当设备到达 Vary on pending 状态边界时
数据区	RTVDTAARA、DSPDTAARA
数据队列	在每个作业仅一次修改下列 API 信息时：QSNDDTAQ，QRCVDTAQ，QMHRDQM，QMHQRDQD
文件(数据库文件)	在关闭、清理，初始化重组时，APYJRNCHG，RMVJRNCHG
文件资源	在打印期间引用时
格式定义	在打印期间引用时
图形符号集	用 GDDM*或 PGR 图形应用程序引用时，在交互装入或用 GSLSS
作业描述	用来建立一个作业时
作业调度	系统间作业调度入口提交一个作业
作业队列	在队列中移出一项或往队列中放入一项时
线路描述	当线路到达 Vary on pending 边界状态时
场所	取得场所 API QLGRTVLC 在作业启动时，如果用户配置文件 LOCALE 值包括个有效的*LOCALE 目标的路径名时
菜单	用 GO 命令显示菜单时
信息文件	在从信息文件中获得信息时（除 QCPFMSG，##MSG1，##MSG2，QSSPMSG 外），例如建立作业日志时，显示信息队列时，在需要 QHST

	记录的信息时，或程序收到一个除标识信息以外的信息时 合并信息文件命令（MRGMSGF）。信息文件*QCPFMSG，##MSG1，##MSG2，或 QSSPMSG 除外
信息队列	除了 QSYSOPR 和 QHST 外，在发送信息，接收信息，列出信息
网络接口描述	在网络接口描述到达 Vary on pending 边界时
结点列表	仅由影响所有目标类型的命令和操作修改
输出队列	当在队列中放入一项或移出一项时
复盖	在打印期间引用时
页定义	在打印期间引用时
页段	在打印期间引用时
面板组	在用 HELP 键请求对提示或面板帮助时，更改所用的日期 在从面板组显示或打印一个面板时
打印描述组	在打印期间引用时
产品可用性	仅由影响所有目标类型的命令和操作来更新
产品装入	仅由影响所有目标类型的命令和操作来更新
程序	RTVCLSRC 命令，在运行非系统程序时，与其它程序连编生成程序目标时
PSF 配置	在打印操作期间引用时
查询定义	在用来成报告时，在输入输出时
查询管理格式	在用来成报告时，在输入输出时
查询管理查询	在用来成报告时，在输入输出时
检索索引	在联机帮助信息中使用 F11 键时，在用 STRSCHIDX 命令时
服务存储	运行 VARCFG，不用网络服务描述目标
SQL 包	仅由影响所有目标类型的命令和操作更新
子系统描述	在子系统启动时
拼写辅助字典	在用来生成另外的字典时，在恢复时，在拼写检索期间在字典中找到一个词且字典不是 IBM 支持的拼写辅助字典
表	程序使用做传输时
用户配置文件	当作业用配置文件做初始化时，在配置文件是成组文件且作业用组中成员启动时，GRTUSRAUT 命令
工作站用户	仅由影响所有目标类型的命令和操作更新
	在用户从 WRKJOBSCDE 面板上选择选项 10=立即提交时

下面是在目标说明中提供的有用信息：

所用天数计数：

- 在最后使用日期更新时计数增加
- 在重存系统中已有目标时，使用的天数来自系统中的目标。如果在重存时没有则计数为零。
- 在重存操作期间删除和再生成的目标会丢失所用天数计数。
- 新目标的使用天数为零。

注：AS/400 系统不能决定新老设备文件之间的不同。如果往系统中重存一个已有的同名的目标，如果不删除，系统则用原有目标的天数计数。

— 数据库文件的使用天数是其中成员使用天数的总和。如果总和溢出，则给出最大值。

使用天数清零：

— 在用 CHGOBJD 或 QLICOBJD API 把使用天数清零时，要记录日期，用户就会知道使用天数有多长。

— 对一个文件的使用天数清零，其中所有的成员使用天数也清零。

能删除使用天数和最后使用日期的一般情况如下：

目标损坏要重存。

当系统不在限制状态时重存一个程序。

DSPOBJD 命令可以显示目标的全部说明，它可以把输入写到一个输出文件中，要得到这个说明，用 RTVOBJD 命令。

RTVMBRD 和 DSPFD 命令提供文件中成员的信息。

下列类型的目标不能修改其使用信息：

警告表（*ACRTBL）	授权表（*AUTL）
配置表（*CFGL）	服务级别描述（*COSD）
数据字典（*DCADCT）	文本（*DOC）
双字节字符集字典（*IGCDCT）	
双字节字符集分类（*IGCSRT）	
双字节字符集表（*IGCTBL）	编辑说明（*EDTD）
结束寄存器（*EXITRG）	格式控制表（*FCT）
过滤器（*FTR）	文件夹（*FLR）
网间包交换描述（*IPXD）	日志（*JRN）
日志接收器（*JRNRCV）	库（*LIB）
方式描述（*MODD）	网络服务描述（*NWSD）
网 BIOS 描述（*NTBD）	产品定义（*PRDDFN）
引用码传输表（*RCT）	对话描述（*SSND）
S/36 机器说明（*S36）	用户队列（*USRQ）

4.11 从一个库往另一库中移动目标

可用 MOVOBJ 命令在库间移动目标。一个目标暂时不用，用一个新版的目标代替超期的目标时用此命令。例如，可把新的主文件临时放在一个库中，与老主文件不在一个库，由于老文件的数据是正常复制到新文件中，那么在没有生成新文件时就不能删除老文件。然后，可删除老文件把新文件移到老文件所在的库中。

仅可移动你有目标管理权的目标。对移出的库要有删除和执行权限，对移入的库要有增加和读的权限，可以从临时库 QTEMP 中移出目标，但不可以移进目标，也不能移动非空的输出队列。

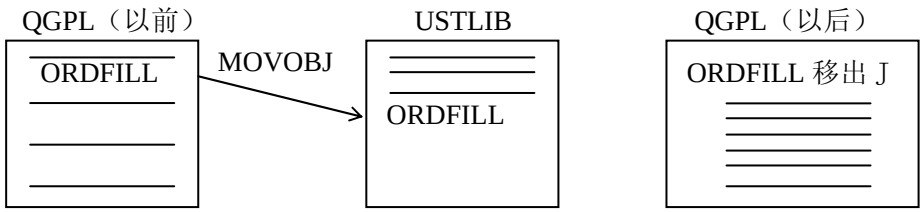
对日志和接收器要移回生成它们所在的库时要有限制，如果日志目标由 RCLST 命令放到了 QRCL 库中，它们必须移回原库才能使用。

下面是不能移动的目标类型：

授权表（*AUTL）	服务级别描述（*COSD）
配置表（*CFGL）	连接列表(*CNNL)
控制器描述(*CTLD)	数据字典(*DTADCT)
双字节字符集字体表(*IGCTBL)	设备描述(*DEVD)

显示工作站信息队列(*MSGQ)　　文本(*DOC)
编辑描述(*EDTD)　　结束寄存器(*EXITRG)
文件夹(*FLR)　　网间包交换描述(*IPXD)
作业调度(*JOBSCD)　　库(*LIB)
线路描述(*LIND)　　方式描述（*MODD）
网 BIOS 描述 (*NTBD)　　网络接口描述(*NWID)
SQL 程序包(*SQLPKG)　　S/36 机器说明(*S36)
系统历史日志(QHST)　　系统操作员信息队列(QSYSOPR)
用户配置文件(*USRPRF)

下例中，一个文件从 QGPL 库移动到 DISTLIB 中，与其它的定货单文件组成一组。



要移动目标，必须规定 TOLIB 以及目标类型（OBJTYPE）：

```
MOVOBJ  OBJ(QGPL/ORDFILL)  OBJTYPE(*FILE)  TOLIB(DISTLIB)
```

在移动目标时，要注意不要移动其它目标依赖的目标。例如，CL 过程要依赖同一库中另一过程中所用的命令定义。生成模块和运行过程要是同一个库，在编译时和运行时，要在某个规定的库或库列表中找命令定义。如果规定库名，命令定义必须运行时与编译时在同一库中，如果规定*LIBL，命令定义可在编译与运行时在库列表中的库间移动。同样，应用程序也依赖于某个库中的一定目标。

一个目标引用另一目标时可能会依赖目标的位置（虽然可用*LIBL 来定位目标）。这样，如果要移动目标，就要修改一些引用关系。下面给出一些目标间的引用关系：

子系统描述引用作业队列，级别，信息队列和程序
命令定义引用程序，信息文件，帮助面板组和包括 REXX 过程的源文件
设备文件引用输出队列
设备描述引用传输表
作业描述引用作业队列输出队列
数据库文件引用其它数据库文件
逻辑文件引用物理文件和格式选择
用户配置文件引用程序，菜单，作业描述，信息队列和输出队列
CL 程序引用显示文件，数据区和其它程序
显示文件引用数据库文件
打印文件引用输出队列

注：在从系统库 QSYS 中移出目标时要小心。这些目标是系统完成操作必不可少的，系统必须能找到这些目标。对 QGPL 库也是同样。特别是对一些作业和输出队列，

移出时要注意。

MOVOBJ 一次只移出一个目标，用一个命令移出多个目标，请看 QUSRTOOL 中的 MOVLIBOBJ 命令。

4.12 生成重复的目标

可用 CRTDUPOBJ 命令来生成已有目标的一个复本。复本与原目标有相同的类型和权限，在同一个 ASP 中，使用这个命令的用户有二个同样的目标。

注：1、如果对已日志的文件生成重复目标，则对复本日志并不活动，但可在其后对其做日志。

2、如果要重复的目标没有成员，则最后使用日期字段为空，使用天数为零。

要对目标进行重复，必须对目标有管理和使用权，对放复本目标的库有使用和增加权，对原目标所在库有使用权，对处理用户配置文件有增加权。

只能重复作业队列，信息队列，输出队列和数据队列的定义。作业队列和输出队列不能重复到 QTEMP 中，对物理文件或备份文件，能规定文件中的数据是否可重复。

下面是不能重复的目标：

AS/400 先进的 36*机器，*CODS，*CFGL，*CNNL，*CTLD，*DATCT，*DEVD，*DTAQ，*IPXD，*JOBSCD，*JRN，*JRNRCV，*LIB，*LIND，*MODD，*NWID，*NWSD，*RCT *SVTSTG，*SPADCT，*SQLPKG，*S36 QSYSOPR QHST，*USRPRF，*USRQ

在某些情况下，可能仅重复一个文件中的某些数据，那么可用 CRTDUPOBJ 与 CPYF 命令的一些选择值来做。

下列命令生成物理文件的一个复本，且重复文件中的数据。

```
CRTDUPOBJ  OBJ(ORDHDRP)  FROMLIB(DSTPRODLLIB)  OBJTYPE(*FILE) +
           TOLIB(DISTLIB2)  NEWOBJ(*SAME)  DATA(*YES)
```

在生成目标复本时，要考虑目标间的引用顺序。多数是以名字来引用的，且多数都要有库名，而复本目标可能包含要引用不同于原目标所在的库中的目标。对于除文件外的所有类型，引用对其它目标的引用重复到复本目标中，对文件，复本目标共享原文件的格式。

在 FROM-LIB 中的物理文件且有逻辑文件使用它，则在 TO-LIB 中必须也有。要比较 FROM 库和 TO 库中的记录格式名和记录级 ID。如果物理文件不匹配，则不能重复逻辑文件。如果在 FROM 库中逻辑文件使用一个格式选择，则假定在 TO 库存中也存在这个格式选择。

4.13 重命名目标

可用 RNMOBJ 命令重命名一个目标。此时，对目标要有管理权，对其所在的库要有更新和执行权。要给授权表改名，对其要有授权表管理权，对 QSYS 库要有更新和读数。

下面的目标不能改名：

*COSD，*DTADCT，*IGLTBL，*MSGQ，*DOC，*EXITRG，*FLR，*JOBSCD，*JRN，*JRNRCV，*MODD，*NWSD，*SOLPKG，*S36，QHST，QSYS，QTEMP，QSYSQPR，*USRPRF

也不能给非空的输出队列改名，也不能给 IBM 提供的命令改名。

要重命名目标，必须给出原名，改后的名及类型。

下面是改名的例子：

RNMOBJ OBJ(QGPL/ORDERL) OBJTYPE(*FILE) NEWOBJ(ORDFILL)

新目标不能用限定名，因为新老目标要在同一库中，如果在执行 RNMOBJ 时，目标正在使用，则不能执行且系统给出信息。

在给目标改名时，要小心不要改有其它目标引用的目标。例如，一个 CL 程序要依赖于程序中使用的命令定义，在运行和编译时命令定义要同名。如果不同名，由于找不到命令程序就不能运行，应用程序也是同样情况。

不能给有日志，日志接收器，数据字典或 SQL 程序包的库改名。一个目标引用另一目标要依赖于目标名和库名。如果改名，就要修改相关的引用涉及的内容。

如果给物理文件或逻辑文件改名，文件中的成员名不改，但可用 RNMM 命令给它们改名。

注：在给 QSYS 库中目标改名时要小心，这些目标是运行系统必须的，在执行系统任务时必须能找到它，对 QGPL 库存中的目标也同样。

4.14 目标压缩和解压缩

可用 CPROBJ 命令来压缩选择的目标以便节省系统的磁盘空间。也可用 DCPOBJ 命令来对已压缩的目标解压。能压缩和解压目标类型为*PGM，*SRVPGM，*MOUDLE，*PNLGRP，*MENU（仅 UIM 菜单）和*FILE（仅显示文件和打印文件）。数据库文件是不允许压缩的。客户目标通讯 AS/400 系统支持的目标可以压缩和解压。要看或取得目标压缩的状态，可用 DSPOBJ 命令（选*FULL 参数）或 RTVOBJD 命令。

4.14.1 目标的压缩

可用 CPROBJ 或 DCPOBJ 命令来压缩和解压上节中给出的目标。在下列情况都为真时才能压缩目标：

系统能获得目标的排它锁。

压缩的大小能节省一磁盘空间。

下面是对目标压缩的一些限制：

在 V1.3 版本之前生成的程序不能压缩。

在 V.3.6 版本之前生成的程序，服务程序和模块在没转换前不能压缩。

在 QSYS 和 QSSP 库中的 IBM 提供的程序在程序的页池值不是*BASE 时不能压缩。用 DSPPGM 命令来看程序的页池值，在除 QSYS 和 QSSP 库外中的程序都可压缩而不考试页池值。

仅属性为 UIM 的菜单可压缩。

仅 DSPF 和 PRTF 的文件可压缩。

只有系统在限制状态（所有子系统都结束）时才可压缩系统库中的目标和程序。

压缩的程序不能在系统中运行，程序会异常结束。

如下表，在非限制状态用多个作业运行压缩会比较快。

表 4—4 用多个作业压缩目标

目标类型	IBM 支持	用户支持
*HLE	作业 3：QSYS	作业 7：USRLIB1
*MENU	作业 2：QSYS	作业 8：
*MODULE	不能用	作业 10：USRLIB1
*PGM	仅在限制状态	作业 5：USRLIB1

*PNLGRP	作业 1：QSYS 作业 4：QHLPSYS	作业 6：USRLIB1
*SRVPGM	作业 11：QSYS	作业 9：USRLIB1

4.14.2 临时解压目标

压缩的目标在使用时会由系统自动解压，在发生下列情况前一直会是临时解压状态：

系统做 IPL。这时会删除临时解压的目标（保留压缩目标）

用 RCLTMPSTG 命令释放临时解压的目标。假如目标没有用到规定日期，要删除解压目标。

临时解压目标用了两天以上或在同一 IPL 中用了五次以上，则要永久解压

用 DCPOBJ 命令解压目标，这时是永久解压

系统对目标有排它锁

注：1、类型为*PGM，*SRVPGM 和*MODULE 的目标不能临时解压。如果调用压缩程序或调试程序，它会自动地永久解压。

2、打开压缩的文件时会自动解压。

3、如果检索得到压缩文件的描述，文件会临时解压，检索文件的两个例子是：

用 DSPFFD 命令显示文件的字段级信息

用 DCLF 命令说明文件

4.14.3 自动解压目标

随 OS/400 或其它 IBM 特许程序来的压缩目标在安装特许程序后会由系统解压。解压只在有足够的可用空间时才做，系统作业调用 QDCPOBJx 来自动开始解压。QDCPOBJ 的作业数会根据处理数加 1，这些作业是系统作业，优先级为 60。用户不能修改、结束或挂起这些作业。QDCPOBJx 作业可以是下列状态之一，它是来自 WRKACTJOB 命令：

RUN（运行）：作业正在解压目标。

EVTW（文件等待）：作业没解压目标。有多个目标要解压时作业才活动（也即，要安装另外的特许程序）。

DLYW（延迟等待）：作业临时停止，下列情况可导致 QDCPOBJx 临时停止：

— 系统在限制状态下运行（即执行了 ENDSYS 或 ENDSBS *ALL 命令）

— 此时正从处理特许程序显示上安装特许程序。作业是在延迟等待状态，最多等 15 分钟才开始解压目标。

LCKW（锁等待）：作业等待一个内部锁。在有一个 QDCPOBJ 作业在 DLYW 状态时发生这种情况。

如果在现有的系统上安装操作系统，有下列的存储要求：

要能启动 QDCPOBJx 作业，起码要有 250MB 的可用存储。系统可用存储要大于 750MB，才能提交作业来解压要安装的系统目标。

系统可用存储若小于 250MB，则不能提交作业，目标要在用时才解压。

系统可用存储在 250MB 到 750MB 之间，仅解压经常使用的目标。

经常使用的目标是指那些至少使用过 5 次且最后使用日期在前 14 天之内，其余不常用的目标保持压缩状态。

如果操作系统是从安装特许内码显示上安装的特许内码和用选项 2 初始化的系统，则必须有至少 1000MB 的可用存储。

如果在 QDCPJOBx 活动时系统中断，则在下一次 IPL 时重新启动作业。

4.15 删除目标

要删除一个目标，可用 DLTxxx 命令，也可用 WRKLIB 显示中的处理目标的删除选项，删目标必须对目标有存在权，对库有执行权，只有权授表的主人或有*ALL 权的人才能删除授权表。

在删目标时，要保证目标没有人用。一般讲，如果有人在用目标。它是不能删掉的。如果在调用程序前用 ALCOBJ 命令分配的程序也能删除。

生成程序命令和生成设备文件命令中都有 REPLACE 参数，它让你能继续使用被替代的老程序，这些再生成程序的老版本都放在 QRPLOBJ 库中。

删系统库中目标时要非常小心。

大多数删除命令中可在目标名栏中填一个类属名，要用 DSPOBJD 命令看一个所要删的目标类属名，详细内容请看 4.3.3。删库的内容请看 4.4.7。

4.16 分配资源

系统中目标的分配要保证集中性，并保证最大可能的一致性。在同一时间对目标完成几个操作，那么对目标要有一定的保护。比如，同时有两个用户读一个目标或一个用户仅可读目标而另一用户能读和更新目标。

AS/400 根据用目标完成的功能来分配目标。例如：

如果一个用户显示或转储一个目标，另一用户可读此目标。

如果一个用户修改、删除、改名或移动目标，其它人不许使用此目标。

如果一个用户保存一个目标，另一用户可读它，但不能更新或删除它。

如果一个用户重存一个目标，没有人可以读或更新此目标。

如果一个用户为了输入打开一个数据库文件，另一用户可读此文件。

如果一个用户为了输出打开一个数据库文件，另一用户可更新此文件。

如果一个用户打开一个设备文件，另一用户仅可读此文件。

一般讲，目标是根据需要才分配的，即在作业步需要目标时，才分配它，使用它，再分配它以使其它作业可以用此目标。请求目标的第一个作业分配目标，在程序中，可以处理例外发生，即处理不能分配请求的目标，详细情况请看第七章和第八章。有时在作业需要一个目标之前分配目标，这样在用目标时不用再等，可保证完成所需功能，这叫目标的预先分配，可用 ALCOBJ 命令预先分配目标。

目标是根据要它们做什么来分配的（读或修改）和它们是否可以共享。文件和成员总是以*SHRRD 来分配的。而数据用规定锁状态来分配锁的级别。锁状态标识使用的目标及是否可共享。有下列五个锁状态：

*EXCL（排它）：目标对请求它的作业保留排它使用。其它的作业不可使用它，但假如目标已分配给其它作业，你的作业也不能得到排它使用目标。这种锁状态用在完成功能前不想让其它作业访问此目标时。

*EXCLRD（排它但允许读）：目标已分配给请求它的作业，但其它作业可以读它。此锁用在不想让其它用户对目标做除读以外的操作时。

*SHRUPD（共享更新）：目标可对其它作业共享读或更新。即另外用户可对同一目标请求共享读锁状态或共享修改锁状态。在用户想要修改一个目标的同时也让另外用户读或修改同一目标时用此状态。

*SHRNUP（共享不更新）：如果作业请求一个共享不更新或共享读锁状态，则目标可与其它作业共享，在用户不修改目标且要保证其它用户也不修改目标时用此状态。

*SHRRD（共享更新）：如果用户不请求排它使用目标，则目标可与其它作业共享，另

外作业可以请求排它允许读、共享更新、共享读或共享不更新锁状态。

注：库的分配不限制在库中目标上完成的操作。即如果一个作业把排它读或共享更新锁状态放在一个库中，那么其它作业就不能往库中放进目标或从库中移出目标，而其它作业仍能更新库中的目标。

下表给出对一个目标有效的锁状态组合：

表 4—5　有效锁状态的组合

如果一个作业有这种锁状态：	另一个作业可能有的锁状态：
*EXCL	None
*EXCLRD	*SHRRD
*SHRUPD	*SHRUPD or *SHRRD
*SHRNUP	*SHRNUP or *SHRRD
*SHRRD	*EXCLRD, *SHRUPD, *SHRNUP, or*SHRRD

对多数类型的目标可以规定五个锁状态，但不是对所有目标。

下表给出对某些类型目标有效的锁状态。

表 4—6　特别目标类型有效的锁状态

目标类型	*EXCL	*EXCLRD	*SHRUPD	*SHRNUP	*SHRRD
设备描述		x			
库		x	x	x	x
信息队列	x				x
面板组	x	x			
程序	x	x			x
子系统描述	x				

要分配一个目标，必须有目标存在权、目标管理权或操作权。分配过的目标在作业结束时会自动取消分配。在任何时候取消分配一个目标，用 DLCOBJ 命令。

可在程序调用前分配它以保护其不被删掉。要不允许程序同时被不同作业运行，在作业调用程序前必须给程序加排它锁。下列目标不能用 ALCOBJ 命令分配：

AS/400 先进的 36 机器，AS/400 先进的机器配置

不能用 ALCOBJ 和 DLCOBJ 来分配 APPC 设备描述。

下例中是一个更新二个文件成员的批作业。文件成员在更新时，可由另一程序读，但在作业运行时其它程序不可更新成员，每个文件的第一个成员用排它允许读锁状态预先分配好。

```
//JOB   JOBD(ORDER)
  ALCOBJ  OBJ((FILEA *FILE *EXCLRD) (FILEB *FILE *EXCLRD))
  CALL  PROGX
//ENDJOB
```

分配给你的目标在使用完它们后要立即取消分配，这样其它用户可以使用它们。在作业前结束时分配过的目标会自动取消分配。如果 FILEA 和 FILEB 中的第一个成员没有预先分配，那么排它允许读限制就不会起作用。在使用文件时，你可能想预先分配它们，这时，应

该保证在使用它们 时没有被修改过。

注：假如一个目标已被分配多次（由多个分配命令），用一个 DLCOBJ 命令不能完全取消分配，每个分配命令要用一个取消分配命令。

对一个没加锁或没有分配所需锁状态的目标用 DLCOBJ 命令不算错误。

下例是修改一个目标的锁状态：

```
PGM
ALCOBJ  OBJ((FILEX *FILE *EXCL)) WAIT(0)
CALL PGMA
ALCOBJ  OBJ((FILEX *FILE *EXCLRD))
DLCOBJ  OBJ((FILEX *FILE *EXCL))
CALL PGMB
DLCOBJ  OBJ((FILEX *FILE *EXCLRD))
ENDPGM
```

文件 FILEX 用排它分配给 PGMA，用排它允许读分配给 PGMB。

用记录锁来分配文件中的数据记录。可用生成文件命令中的 WAITFILE 参数来规定超时前程序等待文件多长时间。

生成文件命令中的 WAITRCD 参数规定等待记录锁的时间。在 CRTCLS 命令中的 DFTWAIT 参数规定等待其它目标的时间。详细内容请看备份和恢复一书。

对由落实控制提供的记录锁的详细内容，请看数据库程序设计一书。

4.16.1 显示目标的锁状态

可用 WRKOBJLCK 或 WRKOBJ 命令显示目标的锁状态。

WRKOBJLCK 命令显示一个目标的所有锁状态。它显示挂起锁和等待锁的情况。对数据库文件，WRKOBJLCK 命令显示的是文件层的锁而不是记录层的锁，也用 WRKOBJLCK 命令显示数据库文件成员的锁。

如果用 WRKOBJ 命令，在显示作业菜单中可以选择锁选项。这些选项包括显示规定作业的所有锁状态，作业挂起的锁，作业等待的锁。但如果作业正等待一个数据库记录锁，那它不出现在目标锁显示中。

下面的命令显示逻辑文件 ORDFILE 在系统中所请求的所有锁状态：

```
WRKOBJLCK  OBJ(QGPL/ORDFILL)  OBJTYPE(*FILE)
```

显示结果为：

```
                          Work with Object Locks
                                                          System:   SYSTEM01
Object:   ORDFILL       Library:   QGPL          Type:   *FILE-LGL

Type options, press Enter.
  4=End job   5=Work with job   8=Work with job locks
```

```
Opt      Job             User            Lock         Status
 _       WORKST04        QSECOFR         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                         *SHRRD       HELD
                                                                        More...
F3=Exit    F5=Refresh    F6=Work with member locks    F12=Cancel
```

第五章　在 CL 过程和程序中处理目标

5.1　在 CL 程序中访问目标

在CL程序命令和过程中引用目标的原则与在每个命令和分别处理不是在程序中使用目标的原则是一样的。目标名可用限定或非限定名，非限定目标名用库列表来查找。

在 CL 过程和程序中引用的大多数目标在运行引用它们时才访问。对限定名（库名/目标名）的目标，在命令运行引用它时必须在指定的库中，但在程序生成时不一定要在此库中，即 CL 语句中用的大部分目标是简单地基于运行时目标的位置。在 5.1.1 中给出一些特例。

如果在 CL 源语句中不用限定名而用库列表，就不用对所有目标做运行时的考虑。如果在编译时用库列表，那么在运行时目标可以在库列表中的任一库里，这就要求在不同库中不能有重复的目标名。如果用库列表，可以在过程生成和命令处理之间把目标移到不同的库中。

在命令运行时所引用的目标必须存在，而在编译时程序 PAYROLL 即时不存在，CL 程序也能编译成功：

```
PGM /*TEST*/
    DCL...
    MONMSG...
    .
    .
    .
    CALL PGM(QGPL/PAYROLL)
    .
    .
```

```
.
ENDPGM
```

实际上，在程序 TEST 活动时，PAYROLL 不一定存在，但在处理 CALL 命令时要存在。在调用程序中被调用程序的生成一定要放在 CALL 命令之前：

```
PGM /*TEST*/
 DCL...
 .
 .
 .
 MONMSG
 .
 .
 .
 CRTCLPGM PGM(QGPL/PAYROLL)
 CALL PGM(QGPL/PAYROLL)

 .
 .
 .
 ENDPGM
```

对于生成命令，比如 CRTCLPGM、CRTDTAARA，在编译和运行时所引用的目标是生成命令定义的，而不是生成的目标。如果用生成命令，生成命令定义在编译时必须在限定命令所用的库中。（换句话说，如果用*LIBL，它必须在库列表中）。

5.1.1 例外情况：访问命令定义、文件和过程

从引用命令定义或文件的源语句生成 CL 程序时要下面两个东西存在：

在文件生成时目标必须存在

在命令引用它们做处理时目标必须存在

即在用 DCLF 时，在生成引用此文件的程序时必须先生成文件。

5.1.1.1 访问命令定义

在生成和命令运行时都可发生访问命令定义，要做语法检查，在使用命令的程序生成期间命令必须存在。在生成期间，命令要存在于引用的库存中，并且在执行时的同一库中。如果不用限定名，那么在运行和生成时要在库列表中的同一库中。

命令名在程序中可用限定名：

如果命令定义在程序运行时不是用库列表访问的

在有多个命令定义用同一名字时

命令名在程序处理和生成时必须一致。如果不同，则会出错。如果修改了命令中的缺省参数，则在命令执行时用新的缺省值。详细内容请看 9.9 及 CHGCMD 的联机帮助信息。

5.1.1.2 访问文件

DCLF 命令的程序或模块在编译时要访问文件。因此，在使用它的程序或模块编译前此文件必须存在。在模块运行和生成程序或服务程序时文件不必须存在。

把描述记录格式及其中字段的 DDS 写进源文件中，然后生成它。即用 CRTDSP 把这些信息编译生成文件目标。在 DDS 中说明的的字段可以是输入字段，输出字段或既输入又输出字段。在程序编译时，这些字段做为变量说明给 CL 程序，程序从显示中通过这些变量管理数据。

也可用 DDS 来生成物理文件。CL 变量的说明使程序包含整个记录，这个变量与文件同名，与记录同长。

除了某些特别的 CL 命令外，CL 程序不能管理除显示文件和数据库文件外的其它类型文件。

在生成文件后可以删除 DDS，但不提倡这样做。在 CL 程序或模块编译后可以删除它所引用的文件。（在命令引用它时要存在，例如在程序中处理 DCLF、SNDF 或 RCVF）。

在命令定义中讲述的使用限定名的原则也适用于文件。详细信息请看 5.2。

5.1.1.3 访问过程

过程是由 CALLPRC 规定的。在生成调用它的模块时不必须要存在。它的存在与否对使用它的程序或服务程序不是必须的。被调用的过程可以是：

在 CRTPGM 或 CRTSRVPGM 中 MODULE 参数里规定的模块

在 BNDSRVPGM 参数中规定的服务程序，此服务程序必须在运行时可用

BNDDIR 参数中规定的联编目录中列出的服务程序或模块

5.1.2 检查目标是否存在

在程序中要使用一个目标前，要检查它是否存在，你是否有权使用它。这在一个功能同时使用多个目标时很有用。

用 CHKOBJ 来检查目标存在与否，可在过程和程序的任何地方使用这个命令，它有下列格式：

```
CHKOBJ  OBJ（库名/目标名）  OBJTYPE（目标类型）
```

其它可选参数检验目标的权限。如果要检查权限且打开文件，那么数据权和操作权都要被检查。在执行这个命令时，有信息会生成告诉你检查的结果，可以监控这些信息并处理它们。例如：

```
            CHKOBJ  OBJ(OELIB/PGMA)  OBJTYPE(*PGM)
             MONMSG  MSGID(CPF9801)  EXEC(GOTO NOTFOUND)
             CALL OELIB/PGMA
             .
             .
             .
NOTFOUND:    CALL FIX001 /*PGMA Not Found Routine*/
             ENDPGM
```

在此例中，MONMSG 仅检查‘目标没找到的’的逃逸信息。对 CHKOBJ 可能送出的所有信息请看联机帮助。在 2.5.10 和第七、八章也有有关的内容介绍。

CHKOBJ 命令不分配目标。对多数应用检查目标的存在性是一个足够的功能，它还要运行一个分配的请求，ALCOBJ 命令即可检查是否存在又可分配目标。

用 CHKTAP 和 CHKDKT 命令可保证带或软盘放在设备上且准备好，它们也产生一些信息，你可在 CL 程序中监控它们。

5.2 在 CL 过程中处理文件

CL 过程和程序中支持两类文件，即显示文件和数据库文件。你可以把显示送往工作站，也可从工作站接收输入给过程和程序使用，或从数据库文件读数据给过程和程序使用。

注：在过程和程序的数据库文件是通过 DCLF 和 RCVF 命令来使用的。在 AS/400 数据库程序设计一书中有打开和关闭数据库文件的说明，它包括 OPNDBF 和 CLOF 命令，它使后来的用高级语言写的程序或过程可以使用数据库文件。

要在 CL 过程和程序中使用文件，必须：

1、用 DDS 源语句规定显示文件或数据库文件的格式，标识字段和条件。

2、用 CRTDSPF，CRTPF 或 CRTLF 生成文件，CL 过程和程序不支持子文件。（信息子文件除外）。

3、对数据库文件，用 ADDPFM 或 ADDLFM 往文件中加成员。如果在生成时已加了成员，则不须这步，在处理过程和程序时必须有成员。但在生成过程和程序时不必须有成员。

4、用 DCLF 在 CL 过程中说明文件，并用适当的数据管理命令引用记录格式。

5、生成 CL 模块。

6、生成 CL 程序。

在一个 CL 过程中仅可用一个显示文件或数据库文件。对于所用的同一个命令，显示文件和数据库文件是很类似的，但也有一点不同：

下列语句仅适用于 CL 过程和程序中使用的数据库文件：

— CL 过程和程序仅可用单格式的数据库文件

— 文件可以是物理文件或逻辑文件，逻辑文件可以引用多个物理文件成员

— 仅可用 RCVF 命令做输入操作，SNDF、SNDRCVF、ENDRCV、WAIT 和 RCVF 中的 DEV 参数不能用于数据库文件

— 在 CL 过程和程序中引用的物理文件不必须用 DDS 来建立。如果不用 DDS，则文件与记录格式同名，且只有一个与文件同名的字段，它与文件的记录等长。（CRTPF 中的 RCDLEN 参数）

— 文件生成时不必须有成员，但在程序处理文件时要有一个成员

— 在处理第一个 RCVF 时为输入打开文件，此时文件必须存在且有一个成员

— 文件一直保持打开，直到过程或 OPM 程序返回或到了文件末。在到达文件末时，送出 CPF0864 信息，不允许对文件做其它操作，过程和程序要监控这些信息，并在到达文件末时来取适当的动作

下列语句仅适用于用在 CL 过程和程序中的显示文件：

— 显示文件可有 99 个记录格式

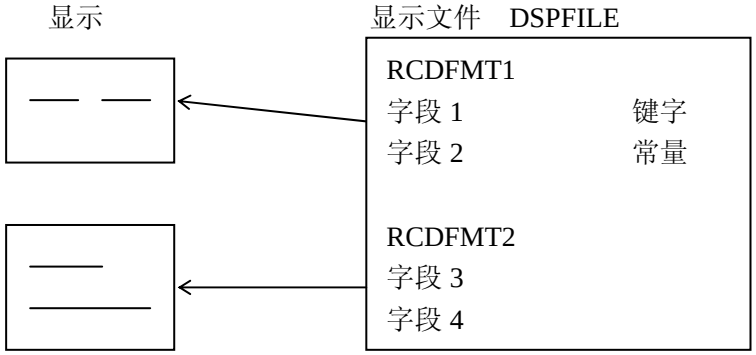
— 对显示可使用所有数据管理命令（SNDF、RCVF、SNDRCVF、ENDRCV 和 WAIT）

— 必须用 DDS 定义显示文件

— 在处理第一个 SNDF，SNDRCVF，或 RCVF 时，为输入/输出打开文件，它一直保持打开直到过程或 OPM 程序返回

注：在第一次发生发送或接收时才打开这两类文件。这样，可在过程和程序中间生成要使用的文件，且在第一次发送或接收前完成替换工作。

显示文件的格式做为 DDS 中的记录格式，每个记录格式可以有字段、条件指示器或常量。一个显示文件可以有几个记录格式。显示文件名，记录格式名和字段名要唯一。虽然在 CL 过程和程序不要求，但其它的高级语言程序可能需要它。



有关 DDS 的详细信息请看 DDS 参考手册。

CL 过程和程序能使用几个调用数据管理的命令，这些命令让你用显示设备接收或发送数据，也可引用数据库文件从其中读记录，这些命令是：

DCLF（说明文件）：定义在过程和程序中使用的显示文件或数据库文件，文件中的字段自动说明为过程和程序中的变量。

SNDF（发送文件）：往显示中送数据。

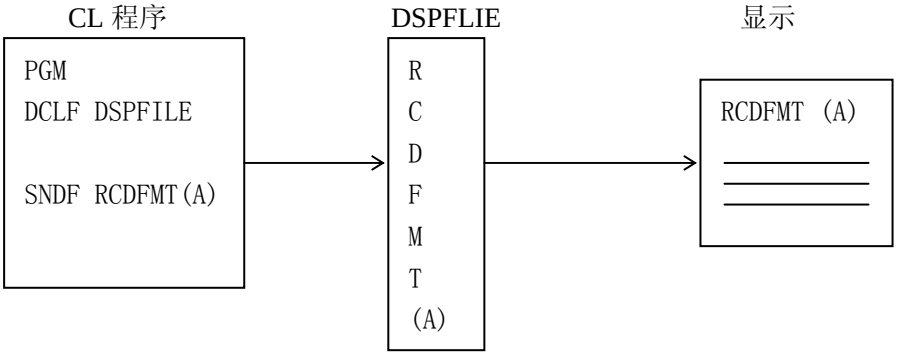
SNDRCVF（发送/接收文件）：往显示中送数据，然后请求输入，接收从显示中来的数据。

RCVF（接收文件）：接收从显示文件或数据库文件中来的数据。

OVRDSPF（用显示文件替换）：允许在运行时用一个显示文件替换过程和程序使用的文件。

OVRDBF（用数据库文件替换）：允许在运行时用一个数据库文件替换过程和程序使用的文件。

这些命令用由 DDS 提供的显示功能让运行程序与设备通讯，并从数据库文件读记录。DDS 提供写菜单和完成基本应用程序间对数据的请求，这是多种 CL 应用的特点。



显示中或记录格式中的字段是在文件的 DDS 中定义的。CL 过程和程序要用这些字段，必须用 DCLF 来引用文件，所用的字段和指示器会自动做为过程和程序的变量。可在任何 CL 命令中使用这些变量，但它们的主要用途是与显示传送信息，在运行时不用 DCLF 命令。

字段的显示格式和选项是在设备文件规定的，是通过指示器来控制的。在 DDS 中可用 99 个指示器，且 CL 也支持它。指示器变量在 CL 过程和程序是用名为&IN01—&IN99 的逻辑格式说明的。它让你显示字段及控制数据管理显示功能，从设备上提供对过程和程序响应的信息。在数据库文件中不用指示器。

5.2.1 在 CL 过程中引用文件

在 CL 模块和程序生成时在编译 DCLF 命令中访问文件，同时文件中的每个字段都做变量说明。如果在编译时文件用限定名，那么在运行时文件也必须在这个库中，如果在编译时用库列表，那么在运行时要在库列表中的一个库里。

5.2.2 打开及关闭文件

在用 CL 编程时，在一个过程或 OPM 程序中只可引用一个文件。引用的文件在第一个发送、接收或发送/接收操作时，会隐含打开。打开的显示文件在它所在程序返回控制时关闭。打开的数据库文件在到达文件末时关闭，或在把控制返回时关闭，它一旦关闭，就不能在同一个调用过程或 OPM 程序中再次打开。

在打开数据库文件时，打开文件中的第一个成员，除非先用 OVRDBF 命令规定了不同的成员（MBR 参数）。假如过程和程序由于错误而结束，则文件关闭。由于文件在过程和程序完成操作前一直保持打开，那么就很容易在运行的过程和程序中打开文件，然后可与另外的过程和程序共享打开数据路径：

文件用 SHARE(*YES)打开或修改过

用 SHARE(*YES)代替文件生效

文件可用这种方式在任何两个过程和程序间共享。详细内容请看 CL 参考手册中的有关生成命令的 SHARE 参数说明。在 CL 过程和程序打开一个显示文件时，总是用于即输入又输出。而在 CL 过程和程序打开一个数据库文件时，总是仅用于输入。

在使用文件的 CL 过程和程序的 RCLRSC 中不要规定 LVL(*CALLER)。如果规定了，所有由过程和程序打开的文件会立即关闭。任何试图访问文件的企图都异常结束。

5.2.3 说明文件

用 DCLF 命令来说明 CL 过程和程序中的显示文件和数据库文件。它不能说明磁带、软盘、打印和混合文件。在过程和程序中仅允许有一个 DCLF 命令，它有下列参数：

```
DCLF    FILE(库名/文件名)
        RCDFMT(记录格式名)
```

注：在模块或程序编译前文件必须存在。

在过程和程序使用的文件必须在 DDS 中规定输入/输出字段，这些字段做为程序变量。在用 DCLF 时，编译程序把文件中每个字段和指示器说明为变量。对字段，CL 变量的名为字段名前加&号。对可选指示器，为指示器前加&IN。例如，在 DDS 中定义了字段 INPUT

和指示器 10，那么 DCLF 命令自动把它们说明为&INPUT 和&IN10，这是在编译时完成的。在一个命令中可以规定 50 个记录格式名，但不能有一个是变量。对数据库文件只能规定一个记录格式。如果用下列的 DDS 生成显示文件 CNTRLDSP 放在库存 MCGANN 中：

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R MASTER
     A                                        CA01(01 'F1 RESPONSE')
     A            TEXT          300      2  4
     A            RESPONSE       15   1  8  4 BLINK
     A
     A
```

从显示文件中可有三个变量&IN01，&TEXT，&RESPONSE，在 CL 过程中要引用这个显示文件，可输入下列 DCLF 源语句：

```
DCLF MCGANN/CNTRLDSP
```

编译程序将分开解释所有显示文件变量，在输出的编译清单中有下面显示的内容：

```
   」
500-                DCLF         MCGANN/CNTRLDSP
      QUALIFIED FILE NAME    'MCGANN    '  'CNTRLDSP   '

        RECORD FORMAT NAME   'MASTER    '

          CL VARIABLE     TYPE     LENGTH     PRECISION (IF *DEC)
           &IN01          *LGL        1
           &TEXT          *CHAR     300
           &RESPONSE      *CHAR      15
```

5.2.4　用显示文件接收和发送数据

在 CL 过程和程序中用 SNDF，RCVF，SNDRCVF 命令使显示文件接收和发送数据。在运行 SNDF 时，与记录格式中输出和输出/输入字段相关的变量的内容由系统格式化后送到显示设备上。同样，在运行 RCVF 时，记录格式中输入和输入/输出字段的值放在相应的 CL 变量中。SNDRCVF 命令把 CL 变量的内容送往显示然后从显示中得到更新后的字段。一般讲，在显示文件中定义为区位十进制的字段在 CL 过程和程序中定义为*DEC 类型，*DEC 字段内部解释为压缩十进制。CL 命令认为压缩和区位数据类型是等价的。在显示文

件中有重叠的字段由于显示位置重合引起其后定义的 CL 变量不能是重叠的。有浮点数据的记录格式不能用在 CL 过程和程序中。

注：如果对一个工作站，SNDRCVF 或 RCVF 命令指出 WAIT(*NO)，或用有 INVITE DDS 键字的记录格式发出 SNDF 命令，则用 WAIT 命令接收数据。

除了信息子文件外，任何试图发送或接收子文件的记录都产生运行时错误，大多数在 DDS 中规定的显示文件的功能都可用，某些功能（例如使用可变的起始行号）不可用，详细内容请看第八章。

下例给出生成程序员菜单的必须步骤，用 SNDRCVF 发送和接收数据。屏幕如下所示：

```
Operator Menu

  1. Accounts Payable
  2. Accounts Receivable
  90. Signoff

Option:
```

首先，输入下列 DDS 源语句，记录格式是 MENU，OPTION 是一个仅输入字段，它用 DSPATR(MDT)，它让系统检查此字段值的有效性。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R MENU
     A                                    1  2'Operator Menu'
     A                                    3  4'1. Accounts Payable'
     A                                    5  4'2. Accounts Receivable'
     A                                    5  4'90. Signoff'
     A                                    7  2'Option'
     A            OPTION         2Y 01   + 2VALUES(1 2 90) DSPATR(MDT)
     A
     A
```

用 CRTDSPF 命令生成显示文件，在 CL 程序设计中，显示文件的名字（INTMENU）可与记录格式（MENU）同名，但有些信息不可以这样做，比如 RPG/400，也可用 SDA 来生成显示文件。

下一步，进入 CL 源语句来运行此菜单，CL 源语句为：

```
            PGM /* OPERATOR MENU */
            DCLF INTMENU
 BEGIN:     SNDRCVF RCDFMT(MENU)
```

```
IF COND(&OPTION *EQ 1) THEN(CALL ACTSPAYMNU)
IF COND(&OPTION *EQ 2) THEN(CALL ACTSRCVMNU)
IF COND(&OPTION *EQ 90) THEN(SIGNOFF)
GOTO BEGIN
ENDPGM
```

在编译以上源语句时，DCLF 命令自动地把字段 OPTION 说明为一个 CL 变量。SNDRCVF 命令缺省为 WAIT(*YES)，即程序要等待由它接收的输入。

5.2.5 控制菜单的 CL 程序

下例给出用 CL 过程显示和控制菜单。用其它方法生成和控制菜单，请看应用显示编程一书。

此例中 CL 过程叫做 ORD040C，它控制显示菜单且根据从菜单选择的选项来决定调用哪个 HLL 过程，它在显示器上给出菜单，菜单显示如下：

```
Order Dept General Menu

 1 Inquire into customer file
 2 Inquire into item file
 3 Customer name search
 4 Inquire into orders for a customer
 5 Inquire into an existing order
 6 Order entry
98 End of menu

Option:
```

显示文件的 DDS 如下：

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* MENU ORDO4OCD ORDER DEPT GENERAL MENU
     A
     A          R MENU                       TEXT('General Menu')
     A                                  1  2'Order Dept General Menu'
     A                                  3  3'1 Inquire into customer file'
     A                                  4  3'2 Inquire into item file'
     A                                  5  3'3 Customer name search'
     A                                  6  3'4 Inquire into orders for a custom+
```

```
A                                        er'
A                                    7  3'5 Inquire into existing order'
A                                    8  3'6 Order Entry'
A                                    9  2'98 End of menu'
A                                   11  2'Option'
A            RESP           2Y001 11 10VALUES(1 2 3 4 5 6 98)
A                                        DSPATR(MDT)
A
A
```

CL 源语句如下：

```
        PGM /* ORD040C Order Dept General Menu */
        DCLF FILE(ORD040CD)
START:  SNDRCVF RCDFMT(MENU)
        IF (&RESP=1) THEN(CALLPRC CUS210)
        /* Customer inquiry */
        ELSE +
          IF (&RESP=2) THEN(CALLPRC ITM210)
          /*Item inquiry*/
          ELSE +
            IF (&RESP=3) THEN(CALLPRC CUS220)
            /* Cust name search */
            ELSE +
              IF (&RESP=4) THEN(CALLPRC ORD215)
              /* Orders by cust */
              ELSE +
                IF (&RESP=5) THEN(CALLPRC ORD220)
                /* Existing order */
                ELSE +
                  IF (&RESP=6) THEN(CALLPRC ORD410C)
                  /* Order entry */
                  ELSE +
                    IF (&RESP=98) THEN(RETURN)
                    /* End of Menu */
        GOTO START
        ENDPGM
```

DCLF 命令指出在处理 SNDRCVF 命令时哪个文件有需要的字段属性。SNDRCVF 命令往显示上发送菜单且接收从显示中选择的选项。如果从菜单上选 98，ORD040C 返回到调用它的过程，ELSE 语句处理替换的响应。

注：此菜单用 CALL 运行，用 GO 命令运行的菜单请看应用显示编程一书。

5.2.6　在 CL 过程中替换显示文件

可用 OVRDSPF 命令代替在 CL 过程和程序中使用的显示文件或修改显示文件的某些参数，这在对 CL 过程和程序编译时用的文件修改或移动时很有用，OVRDSPF 的初始参数为：

```
OVRDSPF    FILE(被替换的文件名) TOFILE(新文件名)  DEV(设备名)
```

OVRDSPF 仅对 CL 过程和程序生成时 DCLF 规定的显示文件有效。程序运行时和生成时所用的文件类型必须一致。

必须在被替换的文件打开前用 OVRDSPF 命令。

在下列情况下，文件可以被替换：

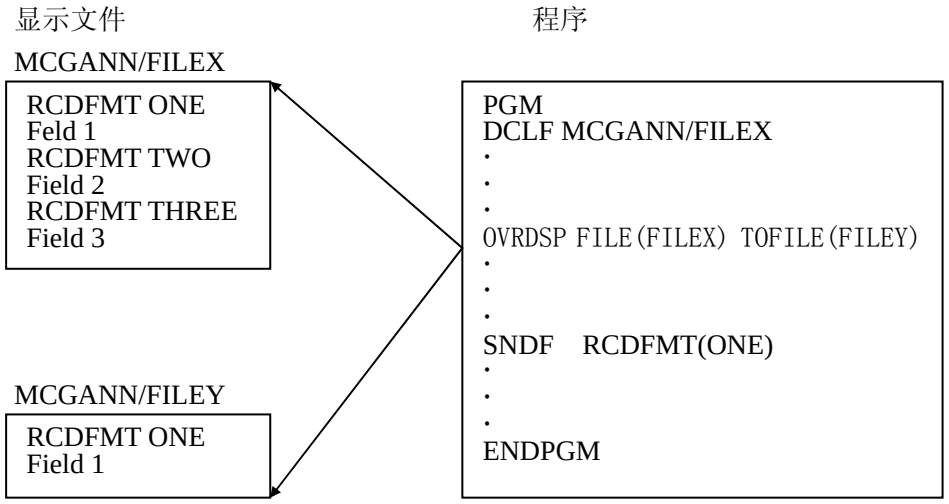
在有 OVRDSPF 命令的过程和程序中打开

在由 CALLPRC 命令把控制转给的另外过程和程序中打开

在由 CALL 命令把控制转给的另外过程和程序中打开

有关替换文件的内容请看数据管理一书。

在替换成不同的文件时，仅在 SNDF、RCVF 或 SNDRCVF 命令中引用 的记录格式名需要在替换文件中，在下面的解释中，显示文件 FILEY 不用记录格式 TWO 和 THREE。



要保证原文件和替换文件有相同的字段定义和指示器名，且要顺序胡同。如果规定 LVLCHK（*NO）要导致非期望结果。在用 OVRDSPF 时另一个要考虑的是 SNDF，RCVF 和 SNDRCVF 中的 DEV 参数。如果规定 DEV（*FILE），系统自动地用被替换文件的当前设备操作，如果在 DEV 中规定一个特别设备，可以发生以下情况之一：

如果用一个显示设备显示文件，假如不是用在 RCVF、SNDF 或 SNDRCVF 命令中规定的设备替换显示文件，则出错。

如果用多个设备显示文件，假如在 RCVF，SNDF 或 SNDRCVF 规定的设备不是 OVRDSPF 命令规定的那些，则出错。

5.2.7　处理多设备显示文件

系统中正常的操作方式是工作站用户注册来请求一个交互作业，同时可有多个用户这样做，他们使用过程的一个逻辑考贝，其中包括过程使用的显示文件，每个请求都用这种使用方式来调用作业，这时不考虑使用多设备显示文件。

在一个用户请求调用一个作业，通过一个显示文件与多个显示工作站通讯时，发生多设备显示的配置，而一个 CL 过程仅能处理一个显示文件，这个显示文件或其中不同的记录格式能被送往几个设备显示。显示设备文件所用的命令主要有：

ENDRCV（结束接收）：它结束已经满足的输入请求。

WAIT（等待）：在命令中规定 WAIT(*NO)时，由前面一个或多个 RCVF 或 SNDRCVF 命令从任何显示设备上接收请求的用户数据。或由一个或多个以前的 SNDF 命令送出包括 INVITE DDS 键字的记录格式。

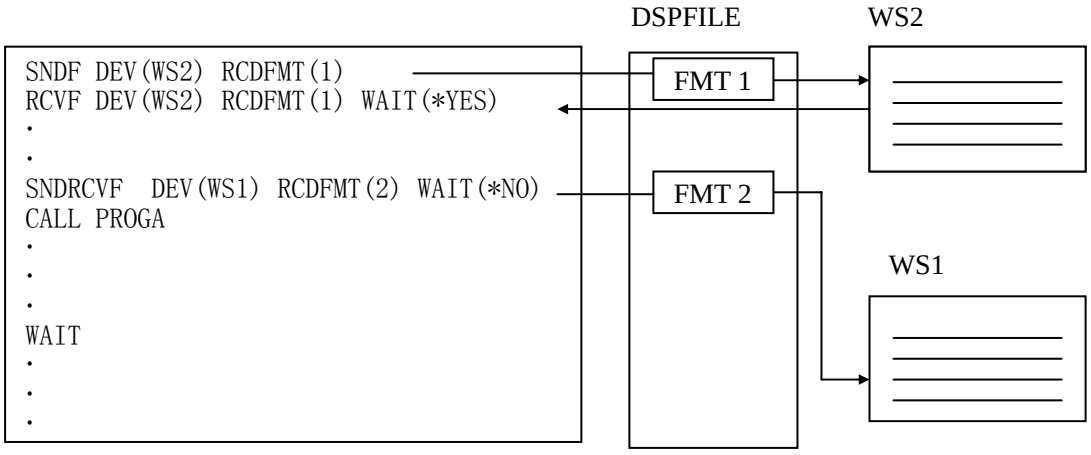
如果用多设备显示文件，设备名必须在 CRTDSPF 中的 DEV 参数中规定，或在修改显示文件时用 CHGDSPF 规定，或在替换命令中规定。设备数必须小于或等于 CRTDSPF 中的 MAXDEV 参数中规定的数。多设备显示配置影响 SNDRCVF 和 RCVF 命令，需要用 WAIT 和 ENDRCV 命令。在多设备用 RCVF 或 SNDRCVF 时，缺省值 WAIT(*YES)能避免进一步的操作，要等到从 DEV 中规定的设备上有输入属性字段返给程序。由于响应可能延迟，也常规定 WAIT(*NO)。这样可以过程和程序在满足接收操作前能继续运行其它的命令。

如果用 RCFVF 或 SNDRCVF 且规定 WAIT(*NO)，过程和程序继续运行直到处理 WAIT 命令。

在记录格式的 DDS 有 INVITE 键字且使用 SNDF 命令，它等价于用 SNDRCVF 且规定 WAIT(*NO)。对 SNDRCVF 和 RCVF，忽略 INVITE 键字。

要访问一个数据记录必须有 WAIT 命令。如果没有可用数据，过程要挂起，直到从显示设备上接收到数据或等待时间超过在 CRTDSPF、CHGDSPF 或 OVRDSPF 中 WAITRCD 参数中规定的时间限制。如果超时，则发出 SPF0889 信息。

在 ENDJOB、ENDSYS、PWRDWNSYS 和 ENDSBS 命令中的控制选项也要有 WAIT，以便能满足取消作业的条件。这时，发出 CPF0888 信息且没有返回数据。如果 WAIT 不与前面的接收请求有关，(例如规定 WAIT(*NO))，则发生处理错误。典型的多设备显示配置如图所示：



在上面的例子中，两个命令给出用缺省值的典型顺序。处理等待从 WS2 的接收操作完成，由于在 DEV 中规定了 WS2，则在 WS2 没响应前不执行 RCVF 命令，即使前面的从其它工作站的请求（没给出）已满足也是这样。但在 SNDRCVF 命令中规定了 WAIT(*NO)，所以不等待从 WS1 的响应，继续处理，调用 PROGA，然后停在 WAIT 命令处，等待满足工作站的没完成的请求或直到功能超时。

WAIT 有下格式：

```
WAIT  DEV(CL 变量名)
```

如果规定了 DEV 参数，CL 变量名是响应的设备名（缺省值名*NONE）。如果有几个接收请求，变量就用遇到 WAIT 命令后的第一个能响应的设备名，然后继续处理，接收的数据放在与设备显示字段相关的变量中。

与 RCVF 一起用 WAIT(*YES)能等待从 某个设备来的数据。在起动设备请求的操作和 RCVF 中规定的记录格式名必须相同。在某些情况下，几个接收请求都没完成，但不能进一步处理没有回答的设备显示。在下例中，有三个命令规定了 WAIT(*NO)。但只能在 WS3 回答后才能继续处理 LOOP：

```
          PGM
          .
          .
          .
          SNDF DEV(WS1) RCDFMT(ONE)
          SNDF DEV(WS2) RCDFMT(TWO)
          SNDRCVF DEV(WS3) RCDFMT(THREE) WAIT(*NO)
          RCVF DEV(WS2) RCDFMT(TWO) WAIT(*NO)
          RCVF DEV(WS1) RCDFMT(ONE) WAIT(*NO)
          CALL...
          CALL...
          .
          .
          RCVF DEV(WS3) RCDFMT(THREE) WAIT(*YES)
LOOP:     WAIT DEV(&WSNAME)
          MONMSG CPF0882 EXEC(GOTO REPLY)
          .
          .
          .
          GOTO LOOP
REPLY:    CALL...
          .
          .
          .
          ENDPGM
```

CL 过程和程序也支持 ENDRCV 命令，它让你能取消没满足的输入请求。SNDF 或

SNDRCVF 也能取消没满足的输入请求，但如果在处理 SNDF 或 SNDRCVF 时有可用数据，则送出 CPF0887 信息。这时，必须用 WAIT 或 RCVF 命令接收数据，或用 ENDRCV 命令明确地取消这个请求，才能再执行 SNDF 或 SNDRCVF 命令。

5.2.8 从数据库文件接收数据

用来从数据库文件接收数据的命令只有一个 RCVF。

在运行 RCVF 命令时，读文件的访问路径中的下一条记录，把在数据库记录格式中定义的字段值放在相应的 CL 变量中。CL 不支持区位十进制或二进制数，这样，文件中定义为区位十进制或二进制的字段在 CL 过程和程序中要定义为*DEC 字段，*DEC 字段做为压缩十进制。如果需要，RCVF 命令会完成从区位十进制或二进制到压缩十进制的转换。有浮点数据的数据库文件不能在 CL 过程和程序中使用。在到达文件末尾时，有 CPF0864 信息送出，这时，说明为记录格式的 CL 变量不因 RCVF 命令的执行而有变化。程序员要监控这个信息，且要采取相应的动作。如果在文件末时要运行 DRCVF 命令，则再次发送 CPF0864 信息。

5.2.9 在 CL 过程和程序中替换数据库文件

可用 OVRDBF 命令来替换 CL 过程和程序中命名的数据库文件或修改已有的数据库文件的某些参数，这在过程和程序生成之后修改文件名或移出文件是很有用的。它可也用来访问文件中除第一个成员以外的其它成员。

此命令的初始参数为：

OVRDBF FILE(被替换的文件名) TOFILE(新文件名) MBR(成员名)

这条命令仅在 CL 过程和程序引用编译时用 DCLF 说明的数据库文件的模块或程序时是有效的。程序处理使用的文件必须和引用的模块或程序生成时所用的文件类型相同。

OVRDBF 命令必须在被替换文件打开之前处理。（第一次使用 RCVF 命令时打开文件），在下列情况下文件会被替换：

文件在有 OVRDBF 命令的过程或 OPM 程序中打开

或 文件在由 CALL 命令把控制传给另外的程序时被打开

或 文件在由 CALLPRC 命令把控制传给另外的过程时被打开

有关 OVRDBF 的详细内容请看数据管理一书。

在替换不同文件时，替换文件必须仅是只读记录格式。用 DDS 定义的多记录格式的逻辑文件如果被定义为只引用一个物理文件成员，则可用做替换文件，在 DDS 中定义的仅一个记录格式的逻辑文件可以定义为引用多个物理文件成员。在程序生成时，格式名不必非得与引用它的格式名相同，但必须保证替换文件的格式与原文件相同。如果规定 LVLCHK（*NO）将导致非期望结果。详细内容请看 DB2 数据库程序设计一书。

5.2.10 引用的显示命令输出文件

有些 IBM 显示命令允许允许将命令的输出放在数据库文件中（QUTFILE 参数）。文件中的数据可直接输入到 CL 过程和程序中且处理它，这些命令的详细内容请看 DB2 数据库程序设计一书。

下面的 CL 过程接收两个参数：用户名和库名，过程确定库中所有程序名、文件名和数据区名，并且正常分配用户权限。

```
            PGM PARM(&USER &LIB)
            DCL &USER *CHAR 10
            DCL &LIB *CHAR 10
(1)         DCLF QSYS/QADSPOBJ
(2)         DSPOBJD OBJ(&LIB/*ALL) OBJTYPE(*FILE *PGM *DTAARA) +
            OUTPUT(*OUTFILE) OUTFILE(QTEMP/DSPOBJD)
(3)         OVRDBF QADSPOBJ TOFILE(QTEMP/DSPOBJD)
(4)         READ:  RCVF
(5)         MONMSG CPF0864 EXEC(RETURN)  /* EXIT WHEN END OF FILE REACHED  */
(6)         GRTOBJAUT OBJ(&ODLBNM/&ODOBNM) OBJTYPE(&ODOBTP) +
            USER(&USER) AUT(*CHANGE)
            GOTO READ                       /*GO BACK FOR NEXT RECORD*/
            ENDPGM
```

（1）QSYS 中的 QADSPOBJ 文件是 IBM 支持的由 DSPOBJD 命令使用的文件，它是在生成输出文件时由命令引用的主要文件，它由 CL 编译程序引用来确定记录格式并且说明记录格式中字段所用的变量。

（2）DSPOBJD 命令生成名为 DSPOBJD 的文件，放在库 QTEMP 中，它与 QADSPOBJ 有相同格式。

（3）OVRDBF 命令用 DSPOBJD 命令生成的文件替换说明的文件（QADSPOBJ）。

（4）RCVF 命令从 DSPOBJD 文件中读一个记录，字段的值复制到相应的 CL 变量中。它隐含由 DCLF 命令说明了，由于使用了 OVRDBF 命令，读 QTEMP/DSPOBJD 文件而不是读 QSYS/QADSPOBJ 文件。

（5）监控 CPF0864 信息，它指出已到达文件末，把控制返回给调用它的过程。

（6）处理 GRTOBJAUT 命令，它使用库名、目标名及类型变量，它们是由 RCVF 命令读到的。

注：QUSRTOOL 中也有几个能生成输出文件的命令（CVTxxx）。

第六章　先进的程序设计方法

这章介绍更先进的程序设计方法，包括：

从高级语言程序（包括 CL 程序）引出的特别函数

用提示和程序员菜单来输入程序源语句

先进函数命令请看系统 API 参考。

这章包括 GUPI，它是 IBM 做的在客户写的程序中使用的变量，在这章末尾给出几个样板程序。

6.1　使用 QCAPCMD 程序

处理命令（QCAPCMD）应用程序接口（API）完成命令串的分析处理，可使用这 API 做：

在运行命令串之前做语法检查

提示命令及接收修收的命令串

使用来自高级语言的命令

显示命令的帮助信息

详细内容请看系统 API 参考。

6.2 使用 QCMDEXC 程序

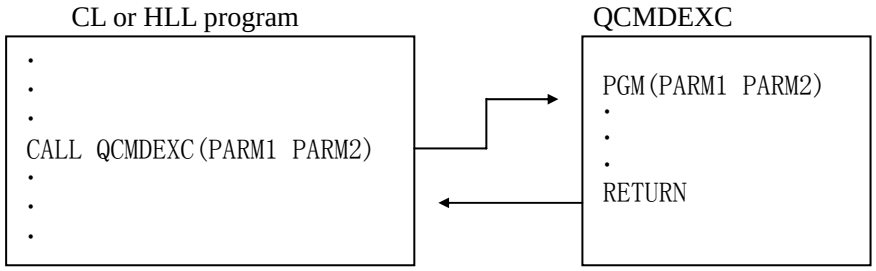
执行命令（QCMDEXC）是 IBM 提供的运行一个简单命令的程序，用它来激活另外的命令：

从用 HLL 写的程序中。

从用 CL 写的过程中。

从在编译时并不知道运行什么命令或不知用什么参数的程序中。

QCMDEXC 程序是用 HLL 或 CL 过程和程序调用，运行 CALL 命令并做为 CALL 的参数传送。



在命令运行后，控制返回给 HLL 或 CL 过程和程序。

命令也可不在程序中运行，这样，在命令中就不能使用变量。另外，仅能用在 CL 过程和程序中的命令也不能用 QCMDEXC 程序运行。调用 QCMDEXC 程序的格式如下：

```
CALL  PGM(QCMDEXC)  PARM(命令  命令长度)
```

在第一个参数中输入做为字符串的要运行的命令,如果命令中有空格,必须用引号括起。字符中的最大长度为 6000 个字符，其中不包括引号。第二个参数（长度）是做为命令传递的字符串长度，它必须是 15 位长 5 位小数的压缩十进制值。下面是替换库列表的调用 QCMDEXC 程序的命令：

```
CALL PGM(QCMDEXC) PARM('CHGLIBL LIBL(QGPL NEWLIB QTEMP)' 31)
```

可把上述语句用在 HLL 或 CL 程序中来替换程序运行时的库列表。用这种方法在运行时没有灵活性，要在运行时灵活替换库列表：

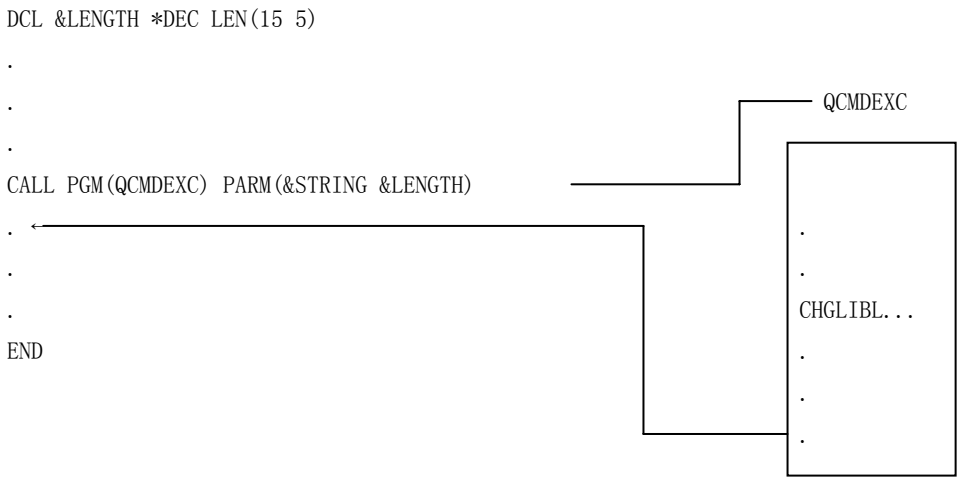
1、在参数列表中把常量换成变量。

2、在调用 HLL 或 CL 程序时规定变量的值。

如下所示：

```
CALL PGM(PGMA) PARM('CHGLIBL...' 3000)
```

```
PGM PARM(&STRING &LENGTH) /* PGMA */
DCL &STRING *CHAR LEN(3000)
```



传给 QCMDEXC 的命令长度，是命令串的最大长度，传递的命令串有引号，长度是串的确切长度。命令串可以传给一个变量，命令长度是这个变量的长度，不用减少命令长度使其与变量中实际的长度相等。

用下列命令可在 CL 过程和程序中运行同一个命令。这是由于被传送的数字字母是 15 位长 5 位小数的压缩十进制值：

```
CALL QCMDEXC (&STRING 3000)
```

不是所有的命令都可用 QCMDEXC 运行，它必须在 CALL 的环境中是有效的，这些命令不能是下列之一：

输入流控制命令（BCHJOB，ENDBCHJOB，和 DATA）

仅用在 CL 程序中的命令

要确定命令能否在 QCMDEXC 中调用，看 CL 参考手册中的语法图，在其右上角有一小框表示命令运行的环境。例如：JOB：I 指出命令可在交互环境下运行，JOB：B 表示可在批作业下运行，EXEC 指出命令可在 QCMDEXC 中运行。

可在命令前写？号请求提示，或在交互作业中调用 QCMDEXC 时选择提示。

在用 QCMDEXC 程序时如果有错，会送出信息，可用 MONMSG 命令监控此信息。详细内容请看第七章和第八章。

如果有语法错，会发送 CPF0006 信息。如果在命令处理期间有错，则送出逃逸信息。在从运行 QCMDEXC 程序的命令中监控此信息，这与在 CL 过程和程序中监控命令信息的方式相同。

6.2.1 在 DBCS 数据中使用 QCMDEXC 程序

可用 QCMDEXC 来请求一个双字节字符集数（DBCS）进入命令中，它的格式为：

```
CALL  QCMDEXC(‘命令’  命令长度  IGC)
```

第三个参数 IGC 告诉系统要接收 DBCS 数据。下列 CL 程序要求用户提供 DBCS 的信息正文，然后系统发送此信息：

```
PGM
```

```
    CALL QCMDEXC ('?SNDMSG' 7 IGC)
 ENDPGM
```

解释如下：

？指出要求系统给出 SNDMSG 命令的提示。

7 是 SNDMSG 命令加上？号的总长度。

IGC 要求 DBCS 数据。

在运行完 QCMDEXC 程序后，可在显示中使用 DBCS 转换：

```
                              SEND MESSAGE (SNDMSG)

TYPE CHOICES, PRESS ENTER.

MESSAGE TEXT . . . . . .  _____________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
TO USER PROFILE . . . . . .  __________  NAME, *SYSOPR, *ALLACT...

                                                                  BOTTOM
F3=EXIT   F4=PROMPT   F5=REFRESH   F10=ADDITIONAL PARAMETERS  F12=CANCEL
F13=HOW TO USE THIS DISPLAY   F24=MORE KEYS
```

6.3 使用 QCMDCHK 程序

QCMDCHK 是 IBM 提供的用于检查一个命令语法的程序，并且可提供命令的提示，但不运行命令。如果要求提示，则命令串与做为通过提示输入的修改值一起返给调用的过程和程序。

从 CL 过程和程序或 HLL 过程和程序都可以调用 QCMDCHK。

典型的使用 QCMDCHK 的方式是：

给用户提示一条命令然后存储起来以备后来处理

确定用户规定的选项

记录处理的命令。首先，用 QCMDCHK 提示，然后用 QCMDEXC 执行，最后记

录执行的命令。

命令的格式为：

```
CALL  PGM(QCMDCHK)  PARM(命令  命令长度)
```

第一个参数是要检查或提示的包括命令的字符串，如果它是变量且要求提示，则由工作站用户进入的命令放在这个变量中。

第二个参数是要检查的命令串的总长度。如果命令串是用引号括起的，长度为串的确切长度。如果命令串传到变量中，则是 CL 变量的长度，它必须是 15 位长 5 位小数的压缩十进制值。

QCMDCHK 对传给它的命令串做语法检查，它检查所有必须的参数及参数允许的值，但它不检查处理的环境，它也不检查命令定义语句。如果检查到有语法错误，则发出 CPF0006 信息，可以监控此信息，CPF0006 是在标识错误的一个或多个诊断信息前发送的。在下例中，由于值 123 对 CRTCLPGM 命令中的 PGM 参数是非法的，故控制转给 ERROR。

```
CALL QCMDCHK('CRTCLPGM PARM(QGPL/123)' 22)
MONGMSG CPF0006 EXEC(GOTO ERROR)
```

可在命令前写一个问号或在命令串的一个或多个键字名前放一个选择提示符来要求对命令做提示。

如果检查和提示时无错，更新过的命令放在第一个参数规定的变量中，并从命令串中去掉提示符。如下例所示：

```
DCL &CMD *CHAR 2000
 .
 .
CHGVAR &CMD '?CRTCLPGM'
CALL QCMDCHK (&CMD 2000)
```

在调用 QCMDCHK 程序运行后，变量&CMD 中有通过提示进入的所有值，如下所示：

```
CRTCLPGM PGM(PGMA) SRCFILE(TESTLIB/SOURCE) USRPRF(*OWNER)
```

此时已去掉命令名前的引号。

在用 QCMDCHK 要求提示时，命令串会传给变量。否则，更新后的命令串不返给过程和程序。必须保证放命令串的变量足够长，能装下从提示中返回的更新过的命令，如果不够长，则发送 CPF0005 信息，且变量中的字符串不改变，如果没有选择提示，提示符仅返回用户输入的内容。

变量长度是由第二个参数决定的，它不是变量实际的长度。在下例中，由于规定的长度太短，会发送 CPF0005 信息。（虽然变量说明的长度足够长）。

```
DCL &CMD *CHAR 2000
 .
 .
```

```
CHGVAR &CMD '?CRTCLPGM'
CALL QCMDCHK (&CMD 9)
```

如果在运行 QCMDCHK 时用 F3 或 F12 从提示中退出，则会有信息 CPF6801 送给调用 QCMDCHK 的过程和程序，且变量中的命令串不做修改。

如果在 PARM、ELEM 或 QUAL 命令定义语句中规定了 PASSATR(*YES)，则用 CHGCMD DFT 命令修改缺省值，缺省值为高亮度就象用户规定的值一样。如果修改过的缺省改回原值，那么它不再高亮显示。

6.4 在 CL 过程和程序中使用信息子文件

在 CL 过程和程序中，信息子文件是文件的一类。要使用子文件信息支持，就要用子文件信息控制记录来运行 SNDF 或 SNDRCVF。用 DDS，则支持 SFLPGMQ 数据，且 SFLINZ 总是活动的。

在 CL 过程和程序中使用信息子文件时，必须命名过程和程序，不能在 DDS 中对 SFLPGMQ 规定星号（*）。在规定一个过程和程序名字时，送往过程和程序信息队列的所有信息都取出来放到信息子文件中，与当前请求有关的所有信息都从 CALL 信息队列取出放到信息子文件。

信息子文件让控制的过程和程序显示一个或多个错误信息。

6.5 在运行时允许用户修改 CL 命令

对大多数 CL 过程和程序，工作站用户用规定参数值传给过程和程序或用显示上的输入字段输入值来给过程和程序提供输入，也可用下列方法提示工作站用户给过程和程序输入值 ：

如果在 CL 过程和程序的源码中，在 CL 命令前写？号，系统显示命令的提示，在过程和程序中已规定的参数值填进去且不允许工作站用户修改，详细内容请看 6.5.1。

如果调用 QCMDEXC 程序且要求选择提示，系统显示命令提示，但不需要在 CL 源程序中规定运行时要处理的 CL 命令，详细内容请看 6.2。

6.5.1 在 CL 过程和程序中使用提示符

可在交互处理 CL 过程和程序中请求提示。如下例：

```
PGM
.
.
.
?DSPLIB
.
.
.
ENDPGM
```

在程序处理时，DSPLIB 命令出现在显示上，它等待用户输入必须的参数值然后按执行键。

在源过程中规定的值都不能由操作员或用户直接改变。例如：

```
PGM
.
.
.
?SNDMSG TOMSGQ(WS01 WS02)
.
.
ENDPGM
```

在调用过程出现 SNDMSG 命令时，用户可以输入 MSG，MSGTYPE 和 RPYMSGQ 参数，但不可以修改 TOMSGQ 的值。下面是对在处理时使用 CL 过程中提示的限制：

在调用提示时，在提示中不能输入变量名或参数值的表达式

命令提示不能嵌在 IF、ELSE 或 MONMSG 命令中

正确	不正确

IF (&A=5) THEN(DO)
?SNDMSG
ENDDO
	IF (&A=5) THEN(?SNDMSG)

下列命令不能使用提示符：

CALLPRC	ELSE	PGM
CALL	ENDDO	RCVF
CHGVAR	ENDPGM	RETURN
COPYRIGHT	ENDRCV	SNDF
DCL	IF	SNDRCVF
DCLF	GOTO	WAIT
DO	MONMSG	

在批作业中不能使用提示。

当你在 CL 源文件成员中的命令前写？时，那么可能在编译时出现诊断信息但不影响编译成功。此时，要小心检查这个信息，看它在运行时是否会通过提示进入的值来更正这个错误。

除上面列出的命令外，可对所有其它你有权使用的命令在交互环境下使用提示，这就能在工作站对这些命令使用提示而不用去查命令手册。

如果在运行时用 F3 或 F12 取消了提示，则会发出 CPF6801 信息，可以监控此信息，在对命令提示时，过程和程序不能接收你输入的命令串，建议先用 QCMDCHK 提示，然后 QCMDEXC 运行，也可使用 QCAPCMD 来提示和运行命令。

6.5.2 CL 命令的选择性提示

可以对命令中的某些参数要求提示，这在使用某些长命令且不想对某些参数要提示时很有用。

选择性提示可用在交互提示中或写在过程和程序的 SEU 源文件中。可在 SEU 中输入选择性提示的源语句但不能用 SEU 在输入命令时使用它。

可用选择性提示做：

选择要提示的参数

决定哪个参数受保护

从提示中省略参数

下面是对选择性提示的限制：

命令名或标号前必须有？号

— 一个或多个选择性提示为？-（问号和减号）

— 要避免 CPF805 信息（虽然编译成功但诊断有问题）

位置参数前不能有选择性提示字符

用选择性提示的参数必须用键字格式

选择提示符和键字之间不允许有空格

选择提示符仅适用于参数，即不能对其值做选择提示

？-不允许复盖程序

如果需要一个参数，必须使用？？选择性提示

由于在提示命令时输入位置是高亮的故可以知道哪个参数是必须的。在选择性提示和正常提示的前边有>号的是用户规定的值，如果一个用户定义的值前没有>号，则把命令缺省值传送给命令处理程序。

如果在命令定义语句的 PARM、ELEM 或 QUAL 中规定 PASSATR(*YES)，且已用 CHGCMDDFT 命令修改了缺省值，缺省值做为用户定义的值（前有>号）。如果这些值又改回了原值，则去掉>号。在使用选择性提示时可用 F8 键来再次显示它们的初值。

如果参数的值用 CL 变量规定且通过选择性提示显示，可在提示中修改它，在命令运行时用修改了的值，在过程和程序中变量的值不改变。如果 CL 过程有下列语句：

```
OVRDBF ?*FILE(FILEA) ??TOFILE(&FILENAME) ??MBR(MBR1)
```

在显示上给出 FILE，TOFILE 和 MBR 三个参数，FILE 参数的值不可改变，而 TOFILE 和 MBR 的值可改，假定&FILENAME 的值为 FILE1，你改成 FILE2，在命令运行时，使用 FILE2，而变量&FILENAME 的值不变，下表给出一些选择性提示符和产生的结果：

输入的内容	显示的值	是否保护	如果没规定传给 CPP 的值	用>作标识
??KEYWORD()	缺省	No	缺省	No
??KEYWORD()	值	No	值	Yes
?*KEYWORD()	缺省	Yes	缺省	No
?*KEYWORD()	值	Yes	值	Yes
?<KEYWORD()	缺省	No	缺省	No
?<KEYWORD()	值	No	缺省	No
?/KEYWORD()	缺省	Yes	缺省	No
?/KEYWORD()	值	Yes	缺省	No
?-KEYWORD()	无	N/A	缺省	N/A
?-KEYWORD()	无	N/A	值	N/A
?&KEYWORD()	缺省	No	缺省	No
?&KEYWORD()	值	No	缺省	No
?%KEYWORD()	缺省	Yes	缺省	No
?%KEYWORD()	值	Yes	缺省	No

输入的内容	按 F5 键或给空格时显示的值	说　明
??KEYWORD()	缺省	用命令缺省值做正常键字提示
??KEYWORD(VALUE)	值	用程序规定的缺省值做正常键字提示
?*KEYWORD()	缺省	仅用一个命令缺省值给出保护的提示
?*KEYWORD(VALUE)	值	仅用程序规定的值给出保护的提示，例如做为信息给出的值不能修改
?<KEYWORD()	缺省	用命令缺省值做正常键字提示
?<KEYWORD(VALUE)	值	用程序规定的缺省值做正常键字提示
?/KEYWORD()	缺省	保留给 IBM 用
?/KEYWORD(VALUE)	值	保留给 IBM 用
?&KEYWORD()	缺省	用命令缺省值做正常键字提示
?&KEYWORD(VALUE)	值	用程序规定的缺省值做正常键字提示
?%KEYWORD()	缺省	仅用命令缺省值给出保护提示
?%KEYWORD(VALUE)	值	仅用程序规定的值给出保护的提示，例如做为信息给出的值不能修改

选择性提示可与 QCMDEXC 和 QCMDCHK 程序一起使用。

调用的格式为:

```
CALL PGM(QCMDEXC or QCMDCHK) PARM(command command-length)
```

下表是选择性提示符的简单介绍

选择的提示符	说　明
??	显示参数是输入属性
?*	显示参数但不是输入属性，用户定义的值传给 CPP
?<	显示参数是输入属性，命令的缺省值传给 CPP，除非修改了显示的参数值
?/	保留给 IBM 用
?-	不显示参数，规定的值或缺省值传给 CPP，在提示替换程序中不允许
?&;	除非用 F9 键，否则不显示参数，一旦显示，它是输入属性，命令的缺省值传给 CPP，除非修改了显示的参数值
?%	除非用 F9 键，否则不显示参数。一旦显示，它不是输入属性，命令缺省值传给 CPP

6.5.3　CL 过程和程序中使用 QCMDEXC 和提示

可用 QCMCEXC 程序调用提示，它允许修改除命令名外的所有值，这比直接使用提示要灵活的多，直接使用提示仅可输入没在源码中规定的值。如果象下面那样直接调用命令提示：

```
?OVRDBF    FILE(FILEX)
```

你可以规定除 FILE 外的其它参数值。但如果在程序执行过程中用 QCMDEXC 调用命令：

```
CALL QCMDEXC PARM('?OVRDBF FILE(FILEX)' 19)
```

你可以规定任何参数，包括 FILE。在此例中，FILEX 是缺省值。带有可修改规定值的提示也可与前面介绍的选择性提示一起使用，但每个要用的键字必须明确的选择，提示直接与命令一起调用：

```
OVRDBF ??FILE(FILEX) ??TOFILE(*N) ??MBR(*N)
```

6.6 使用程序员菜单

程序员菜单可用 QPGMMENU 程序直接调用，或用 STRPGMMNU 命令也可，可用命令来进一步规定使用的缺省值。另外，STRPGMMNU 也支持程序员菜单的其它选项。

6.6.1 使用 STRPGMMNU 命令

此命令可在完成下列任务：

完成与 QPGMMENU 的相同功能

填写标准的输入字段，有四个命令参数允许你在菜单底部填入标准的输入字段：— 源文件

— 源库

— 目标库

— 作业描述

可用一个或多个参数来控制菜单的初值。可以把它设计为注册的初始程序的一部分，或是用户调用的自己的功能。

下例给出一个程序，它对每个需要不同初值的应用程序有各自的功能：

```
        PGM
        CHGLIBL    LIBL(PGMR1 QGPL QTEMP)
LOOP:
        STRPGMMNU SRCLIB(PGMR1) OBJLIB(PGMR1) JOBD(PGMR1)
        MONMSG     MSGID(CPF2320) EXEC(GOTO END) /* F3 or F12 to leave menu */
        GOTO LOOP
END:    ENDPGM
```

控制程序员菜单选项：

其它参数帮你控制菜单和它的功能。例如，可以规定 ALWUSRCHG(*NO)来避免用户修改菜单中的值。由于使用菜单的用户可以调用 STRPGMMNU 命令和修改值，所以不必考虑安全方面的问题。（用户也可用 F10 键调用命令入口来启动这个功能）。如果用 STRPGMMNU 来显示菜单，可防止用户调用 QPGMMENU 程序（由授权规定），但不能防止用户从其它请求来调用 STRPGMMNU 命令。

恰当的菜单生成选项：

EXITPGM 和 DLTOPT 参数允许提供自己的菜单生成选项支持（选项 3）。当用选项 3 时也可调用用户程序，下面介绍 EXITPGM 参数的典型用法。

6.6.1.1 EXITPGM 参数

EXITPGM 参数可有下列用途：

用选项 3 来提交作业，修改生成命令用的缺省值：

例如，不用 F4 键，EXITPGM 参数将修改一个或多个生成命令来规定你自己要求的缺省值。如用 F4 键，EXITPGM 能提交命令如同由程序员输入一样（无参数修改）。

要修改参数而不管程序员使用 F4 键与否：

这要求检索&RQSDTA512 参数的值（它传给出口程序）来看是否已经被用以及是否是适合要求的值。

要修改 SBMJOB 命令的其它参数：

能修改 SBMJOB 命令的用户参数来规定作业描述的值来替代*CURRENT 的值，也可用 RTVJOBA 命令来取得一个或多个作业描述的值，输入一些特性做规定的值。

要实施本地程序设计规范：

假如已有命名标准，要求所有物理文件名必须是 7 个字符且以 P 结尾，那么出口程序要校验试图使用 CPTPF 命令而名字不符合这个标准的情况。

有助于替代已有的目标，例子请看 QUSRTOOL 库中的文件 QATTINFO 内成员 SBMPARMS，以及 STRPGMMNU 出口程序。

6.7 DBCS 数据的应用程序设计

在设计处理双字节字符集数据的程序或把字母数字应用程序转换为双字节程序时，必须要特别考虑一些情况。

6.7.1 设计 DBCS 应用程序

设计 DBCS 应用程序方法与设计字母数字程序一样，下面是要特别注意的问题：

要标识在数据库文件中使用的 DBCS 数据

设计的显示和打印要能使用 DBCS 字符

如果需要，对交互应用提供输入数据的双字节转换

在显示文件中，用 DDS 键字 IGCCNV 来规定 DBCS 转换

写一个能由程序显示的双字节错误信息

规定扩展字符处理，这样系统能打印和显示所有 DBCS 数据，确定哪 些 DBCS 字符必须要定义

6.7.2 转换字母数字程序处理 DBCS 数据

如果字母数字应用程序使用外部描述的显示文件，那么仅改此文件就可以让应用程序处理 DBCS 字符，要转换应用程序，依下列步骤做：

1、生成字母数字应用程序的一个拷贝。

2、把字母数字常量和字母修改为 DBCS 常量和字母。

3、把文件中的字段类型修改为下列之一：

DBCS—open（O）

DBCS—only（J）

DBCS—either（E）

不用修改字段长度。

4、把转换好的显示文件放在另外库中，各字与原来的相同。

5、要在作业中使用转换后的文件，用 CHGLIBL 修改作业用的库列表，把 DBCS 显示文件所在的库放在字母数字显示文件所在库的前边。

6.8 在 CL 程序中使用 DBCS 数据

下面的程序给出在一个 CL 程序中使用不同的键盘转换。注意在这个程序中怎么样在说明中使用 DBCS 数据，而命令本身是字母数字的。运行时，程序给出对 DDS 显示文件如何用不同的键盘转换。

```
PGM
        DCLF      IGCTEST
START:      CHGVAR      &OUTPUTA        'ABCDEFGHIJ'

            CGVAR       &OUTPUTJ        'ABCD'
            CGVAR       &BOTHJ          'ABCD'
            CGVAR       &OUTPUTE        'EFGH'
            CGVAR       &OUTBUTO        'AB C D F,

LOOP:  SBDRCVF

            IF  &  IN01  RETURN
            CHGVAR      &OUTPUTA        &INPUTA
            CHGVAR      &OUTPUTJ        &INPUTJ
            CHGVAR      &OUTPUTE        &INPUTE
            CHGVAR      &BOTHE          &INPUTE
            CHGVAR      *OUTPUTO        INPUTO

          GOTO      LOOP
                                                    RV2W505-0
    ENDPGM
```

6.9 样板 CL 程序

下面的样板程序给出灵活的、简单的、多样的 CL 程序，也解释所完成的功能和可用它的用户。

6.9.1 起动所用的初始程序（程序员）

```
PGM
CHGLIBL LIBL(TESTLIB QGPL QTEMP)
CHGJOB OUTQ(WSPRTR)
TFRCTL QPGMMENU
ENDPGM
```

测试库放在库列表的第一位，打印机选择一个输出队列且显示程序员菜单。

6.9.2 把目标从测试库移到产品库中

```
PGM PARM(&OBJ &OBJTYPE &OPER)
DCL &OBJ *CHAR LEN(10)
DCL &OBJTYPE *CHAR LEN(7)
DCL &OPER *CHAR LEN(1)  /* R=Replace M=Move */
IF ((&OPER *NE 'M') *AND (&OPER *NE 'R')) THEN(DO)
        SNDPGMMSG MSG('Operation code must be "R" or "M" ')
        RETURN
        ENDDO
IF ((&OBJTYPE *NE *PGM) *AND (&OBJTYPE *NE *FILE) *AND (&OBJTYPE +
    *NE *DTAARA)) THEN(DO)
        SNDPGMMSG MSG('Object' *BCAT &OBJ *BCAT ' must be *PGM, +
                  *FILE, or *DTAARA')
        RETURN
        ENDDO
CHKOBJ BLDLIB/&OBJ OBJTYPE(&OBJTYPE)
MONMSG MSGID(CPF9801) EXEC(DO)
        SNDPGMMSG MSG('Object or object type does not exist +
                  in BLDLIB')
        RETURN
        ENDDO
IF (&OPER *EQ 'M') THEN(DO)
        MOVOBJ BLDLIB/&OBJ OBJTYPE(&OBJTYPE) TOLIB(PRODLIB)
        MONMSG MSGID(CPF3208) EXEC(DO)
                       SNDPGMMSG MSG('Object' *BCAT &OBJ *BCAT ' +
                                 already exists in PRODLIB')
                       RETURN
                       ENDDO
        CHKOBJ PRODLIB/&OBJ OBJTYPE(&OBJTYPE)
        MONMSG MSGID(CPF9801) EXEC(DO)
        SNDPGMMSG MSG('Object or object type does not +
                  exist in PRODLIB')
                  RETURN
                  ENDDO
        ENDDO
RETURN
ENDPGM
```

目标名、类型和操作码是从另外的过程和程序传来的。要检查操作码和目标类型是否正确，目标是否在测试库中存在。如果目标库中没有此目标，则把它从测试库移到目标库中，然后确认移动，可对移动命令加一些目标的权限，或处理其它请求和其它类型目标。

6.9.3 在应用程序中保存规定的目标

```
PGM
SAVOBJ OBJ(FILE1 FILE2) LIB(LIBA) OBJTYPE(*FILE) DEV(TAP01) +
  CLEAR(*YES)
SAVOBJ OBJ(DTAARA1) LIB(LIBA) OBJTYPE(*DTAARA) DEV(TAP01)
SNDPGMMSG MSG('Save of daily backup of LIBA completed') +
  MSGTYPE(*COMP)
ENDPGM
```

这个程序保证有规律的重复过程有一致的命令入口，也可加一个 SAVOBJ 命令。这个程序依赖于操作员选择正确的软盘或磁带来为每个应用程序做周期的备份。也可以由对每个保存操作分配唯一的软盘或磁带来控制。如果要在每周分别保存工资文件，可以给不同的软盘或磁带名，然后写一个程序来比较软盘或磁带名是否与这周用的相同。

6.9.4 恢复非正常结束的系统（系统操作员）

```
PGM
DCL &SWITCH *CHAR LEN(1)
RTVSYSVAL SYSVAL(QABNORMSW) RTNVAR(&SWITCH)
IF (&SWITCH *EQ '1') THEN(DO)  /*CALL RECOVERY PROGRAMS*/
        SNDPGMMSG MSG('Recovery programs in process. +
          Do not start subsystems until notified') +
          MSGTYPE(*INFO) TOMSGQ(QSYSOPR)
        CALL PGMA
        CALL PGMB
        SNDPGMMSG MSG('Recovery programs complete. +
          Startup subsystems') +
          MSGTYPE(*INFO) TOMSGQ(QSYSOPR)
        RETURN
        ENDDO
ENDPGM
```

6.9.5 提交作业（系统操作员）

```
PGM  /*DAILYAC*/
SBMJOB JOB(DAILYACCRC) JOBD(ACCRC2) +
             CMD(CALL ACCRC305 PARM(DAILY))
SNDPGMMSG    MSG('Daily Accounts Receivable job DAILYACCRC +
             submitted to batch') MSGTYPE(*COMP)
ENDPGM
```

系统操作员不用填写提交作业的所有参数，只调用此程序既可。

6.9.6 等待从设备显示上输出超时

```
            PGM
            DCLF FILE(QGPL/MENU)
START:      SNDRCVF DEV(*FILE) RCDFMT(MENUFMT) WAIT(*NO)
            WAIT
            MONMSG MSGID(CPF0889) EXEC(SIGNOFF)
            CHGVAR VAR(&IN99) VALUE('0')
            IF COND(&IN01) THEN(GOTO CMDLBL(START))
OPTION1:    /* OPTION 1-ORDER ENTRY */
            IF COND(&OPTION *EQ '1') THEN(DO)
            CALL PGM(ORDENT)
            GOTO CMDLBL(START)
            ENDDO
OPTION2:    /* OPTION 2-ORDER DISPLAY */
            IF COND(&OPTION *EQ '2') THEN(DO)
            CALL PGM(ORDDSP)
            GOTO CMDLBL(START)
            ENDDO
OPTION3:    /* OPTION 3-ORDER CHANGE */
            IF COND(&OPTION *EQ '3') THEN(DO)
            CALL PGM(ORDCHG)
            GOTO CMDLBL(START)
            ENDDO
OPTION4:    /* OPTION 4-ORDER PRINT */
            IF COND(&OPTION *EQ '4') THEN(DO)
            CALL PGM(ORDPRT)
            GOTO CMDLBL(START)
            ENDDO
OPTION9:    /* OPTION 9-SIGNOFF */
            IF COND(&OPTION *EQ '9') THEN(SIGNOFF)
OPTIONERR:  /* OPTION SELECTED NOT VALID */
            CHGVAR VAR(&IN99) VALUE('1')
            GOTO CMDLBL(START)
            ENDPGM
```

这个程序对每个用户选择一个选项规定一个时间限制，如果时间到，则注销此用户。显示文件用下面的命令生成：

```
CRTDSPF FILE(MENU) SRCFILE(QGPL/QDDSSRC) SRCMBR(MENU)  +
    DEV(*REQUESTER) WAITRCD(60)
```

显示文件使用*REQUESTER 设备，在发出 WAIT 命令时，它等待 WAITRCD 中规定的

时间（60 秒）。下面是显示文件的 DDS：

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ... ... 8

0100       A                                      PRINT CA01(01)
0200       A          R MENUFMT                   BLINK
0300       A                                      TEXT('Order Entry Menu')
0400       A                                    1 31'Order Entry Menu'
0500       A                                    2  2'Select one of the following:    '
0600       A                                    3  4'1.  Enter Order'
0700       A                                    4  4'2.  Display Order'
0800       A                                    5  4'3.  Change Order'
0900       A                                    6  4'4.  Print Order'
1000       A                                    7  4'9.  Sign Off'
1100       A                                   23  2'Option:'
1200       A            OPTION         1   I 23 10
1300       A  99                                  ERRMSG('Invalid option selected.')

                          * * * * * *  E N D  O F  S O U R C E  * * * * *
```

程序执行 SNDRCVF　WAIT(*NO)来显示菜单和用户选择的选项，然后用 WAIT 命令来接收用户选择。如果用户输入 1—4 选项，则调用相应的程序。如果选择 9，则注销用户，如果输入无效选项，则显示‘Invalid option selected’信息。

用户可选择另外有效的选项，如果在 60 秒内没响应，则发送 CPF0889 信息且由 MONMSG 命令发出注销命令。

可用有 INVITE 键字的记录格式的 SNDF 命令代替 SNDRCVF WAIT(*NO)，功能是一样的。

6.9.7　取得程序属性

DSPPGM 命令可用来显示一个程序的属性。要把某些属性（例如程序类型、源成员、说明、生成日期）取出放在 CL 变量中。可用 DSPOBJD 命令产生一个输出文件，然后用 CL 过程和程序中的 DCLF 和 RCVF 命令来读这个输出文件，要取得 DSPPGM 的另外一些属性（比如 USRPRF），也可使用 QUSRTOOL 工具（RTVPGMATR）或 QCLRPGM（程序信息 API）

6.10　从磁带或软盘上装入和运行应用程序

LODRUN 命令可从其它用户的软盘或磁带上装入和运行应用程序或软件。

在运行 LODRUN 命令时：

检索介质上用户写的应用程序，它必须是名为 QINSTAPP。如果用磁带，磁带首先要卷到头

如果在用户系统中的 QTEMP 中已有 QINSTAPP，删掉它

用 RSTOBJ 命令把 QINSTAPP 重存到 QTEMP 中

控制系统转给 QINSTAPP 程序，它可以把其它应用程序重存到用户系统上且运行这些程序

6.10.1　编写应用程序者的责任

提供 QINSTAPP 程序用户的责任是编写及支持它，QINSTAPP 不是由 IBM 提供的。设计这个程序可完成多种任务，程序可以做：

重存及运行其它程序或应用

重存一个库

删除其它程序或应用

改正应用中存在的问题

下面给出 QINSTAPP 程序的例子，它使用 LODRWN 命令把程序保存到磁带或软盘上且装入到系统中，LODRWN 把控制从系统传给程序，然后完成程序功能。

```
PGM          PARM(&DEV)  /* "Device" 是仅允许的值                    */
DCL          VAR(&DEV)   TYPE(*CHAR) LEN(10)
DCL          VAR(&MODEL) TYPE(*CHAR) LEN(4)

/* 检查型号、版本等                                                  */
RTVSYSVAL    SYSVAL(QMODEL) RTNVAR(&MODEL)
IF           (&MODEL *EQ 'xxxxx') THEN...

/* 为新应用程序安装一个库                                            */
RSTLIB       SAVLIB(NEWAPP) DEV(&DEV) ENDOPT(*LEAVE) +
               MBROPT(*ALL)
/* 安装启动新应用程序的命令                                          */
             RSTOBJ OBJ(NEWAPP) SAVLIB(QGPL) DEV(&DEV) +
               MBROPT(*ALL)

END:         ENDPGM
```

第七章　定义信息

在 AS/400 系统中，过程和程序之间、作业之间、用户之间以及用户和过程和程序之间是通过信息通讯的。信息可预先定义，也可立即生成。

预先定义的信息在使用它的程序外生成和存在，它放在信息文件中，且有一个信息号，系统预先定义的信息是：

```
CPF0006  Errors  occurred  in  command
```

立即信息是由发送者在发送时建立的，它不放在信息文件中，下面是立即信息的例子：

```
From  . . . :   QSYSOPR      06/12/94   10:50:54
System going down at 11:00; please sign off
```

系统中有一组预先定义的信息，它在系统程序之间及系统和用户之间通讯，每个特许程序都有一个信息文件存在与特许程序所在的同一库中。例如：系统信息存在 QSYS 库中的

QCPFMSG 文件中。

信息文件中的每个预先定义的信息都由 7 个字符做唯一标识，且由信息描述来定义。信息描述包括，信息正文和帮助正文、严重级别、有效及缺省的回答值和几个其它属性。

在系统中所有发送和接收信息都是通过信息队列传送的。

发出给请求者的响应信息自动显示在请求者的显示屏上，所有其它信息，用户、程序或过程必须从队列中接收信息或显示它。系统中有几个 IBM 支技的信息队列，详情请见 7.4。

系统也写一些信息发给日志，作业日志包括作业进入的有关的信息，历史日志包括作业、子系统和设备状态信息。详情请看 8.7。

可生成自己的信息队列和信息描述，由生成预先定义的信息，可以使用几个过程和程序中的相同信息，但只能定义一次，也可以修改且把预先定义的信息转换成非英语的其它语言，这不影响它们的使用。如果在过程和程序中定义了信息，在修改这些信息后，要重编译模块或程序。

可生成自己的信息和信息文件，系统信息处理功能允许你做：

生成和修改信息队列（CRTMSGQ，CHGMSGQ）
处理信息队列（WRKMSGQ）
生成和修改信息文件（CRTMSGF，CHGMSGF）
增加信息描述（ADDMSGD）
修改信息描述（CHGMSGD）
去掉信息描述（RMVMSGD）
发送立即信息（SNDMSG）
发送中断信息（SNDBRKMSG）
发送程序信息（SNDPGMMSG）
发送用户信息（SNDUSRMSG）
显示信息和信息描述（DSPMSG，DSPMSGD）
处理信息（WRKMSG）
处理信息描述（WRKMSGD）
使用 CL 过程和程序来做：

— 把信息发送给工作站用户或系统操作员（SNDUSRMSG）
— 从信息队列接收信息（RCVMSG）
— 往信息队列发送信息回答（SNDRPY）
— 从信息文件取得信息（RTVMSG）
— 从信息队列去掉信息（RMVMSG）
— 监控送往调用信息队列的逃逸、注意、状态信息（MONMSG）

使用系统回答列表来规定对作业发送的预先定义的查询信息的回答(ADDRPYLE)，
修改回答列表（CHGRPYLE），去掉回答列表（RMVRPYLE），处理（WRKRPYLE）。

在发送信息时，定义为下列类型之一：

信息（*INFO）：传送有关功能的条件信息
查询（*INQ）：传送有关请求回答的信息
通知（*NOTIFY）：要过程和程序采取更正动作的条件说明或从调用过程和程序给出的回答。一个过程和程序可以监控它调用的过程和程序的通知信息。
回答（*RPY）：对接收到的查询或通知信息的响应
发送拷贝（*COPY）：由发送者复制的查询或通知信息
请求（*ROS）：来自接收过程和程序的功能请求

完成（*COMP）：传送工作的完成状态

诊断（*DIAG）：在应用程序或输入数据中处理系统功能时的错误信息

状态（*STATUS）：说明由过程和程序所做工作的状态。过程和程序可把监控来自它调用的过程和程序的状态信息显示在屏幕上，也能通知用户采取相应的处理操作

逃逸（*ESCAPE）：说明过程和程序必须异常结束的信息。过程和程序可以监控来自它调用的过程和程序或机器的逃逸信息，在送出此信息后，控制并不转给发送它的程序

这章介绍：

如何生成自己的信息文件

如何往信息文件加一个信息描述

信息队列的类型

如何生成信息队列

7.1 生成信息文件

要生成自己预先定义的信息，必须首先生成放信息的文件。用 CRTMSGF 命令来生成信息文件，然后用 ADDMSGD 来描述信息且把它们放到信息文件中。

在 CRTMSGF 命令中，用 SIZE 参数规定以 K 字节为单位的最大尺寸，下列公式是计算所用最大尺寸的：

S+(I×N)，

其中： S=初始存储空间

I=每次增加的存储空间

N=要增加的存储空间的次数

S、I、N 的缺省值为 10，2 和*NOMAX。

例如，S=5、I=1、N=2，在文件到了初值 5K 时，系统自动地加 1K，最多能到 7K。如果规定 N=*NOMAX，信息文件的最大尺寸为 16M。

在对信息文件规定了最大尺寸而文件满时，你不能修改信息文件的尺寸，要生成另外的信息文件然后在新文件中生成信息，可用 MRGMSGF 命令从一个信息文件往另一个中复制信息描述。要避免这样做，在生成信息文件时要认真计算最大尺寸，或就规定 N=*NOMAX。

7.1.1 确定信息文件的大小

可用下列公式确定信息文件的大小（括号中给出的是 ADDMSGD 的参数）：

信息索引=42 字节+信息长度

联机帮助信息（SECLVL）=16 字节+帮助信息长度

格式（FMT）=14 字节+（3×FMTS 的个数）

类型和长度（TYPE 和 LEN）=48 字节

特别值（SPCVAL）=2+(64×SPCVAL 的个数)

范围（RANGE）=64 字节

关系（REL）：关系的长度

缺省（DFT）：缺省回答的长度

缺省程序，日志问题和转储表（DFTPBM，LOGPRB，DMPLST）

=35+(2×DMPLST 的个数)

ALROPT=12 字节

信息文件中的最小项是 59 字节，最大项是 5764 字节，下表给出可能的最大项：

信息索引	42 字节	类型和长度	48 字节
信息正文	148 字节	20 个特别值	1282 字节
信息帮助正文	3016 字节	20 个值	640 字节
99 个格式	311 字节	缺省回答值	32 字节
缺省程序和转储列表	233 字节	警告选项	12 字节

在下例中，用 CRTMSGF 命令生成信息文件 USRMSG：

```
CRTMSGF    MSGF(QGPL/USRMSG) +
           TEXT('Message file for user-created messages')
```

如果生成用 RPG 写的 DSPLY 操作码中的信息文件，文件名必须是 QUSERMSG。

7.2 往文件中加信息

可用 ADDMSGD 命令描述预先定义的信息且加到生成的信息文件中。在命令中，可以规定信息标识，信息文件的名字，及信息描述。在信息描述中可以规定：

信息正文（必须的）及可选的替换变量
信息帮助正文及可选的替换变量
严重码
替换变量所用信息数据的格式说明
回答值的有效性检查
回答的缺省值
逃逸信息的缺省信息处理动作
生成级别
报警选项
编码字符集 ID（CCSID）

放在信息描述的每项将在下面详细说明。

下列命令也用来处理信息描述：

CHGMSGD：修改信息描述
DSPMSGD：显示信息描述
RMVMSGD：取消信息描述
RTVMSGD：取得信息描述
MRGMSGF：把一个信息文件合并到另一个信息文件中。
WRKMSGD：显示信息文件中的信息列表且允许加、改或删除信息描述。

7.2.1 分配信息标识

在 ADDMSGD 命令中规定的信息标识用来引用一个信息，也做为这个信息描述的名字，它必须是 7 个字符。PPPmmnn

PPP 是产品或应用代码，mm 是组号，nn 是字类型号，mmnn 也可进一步分组，它由 0—9 数字和 A—F 字母组成。例如：

CPF1234　是 CPF 的的信息 1234

在生成自己的信息时，建议用 U 做第一个字符。例如：

USR3564

码的第一个字符必须是字母，第二、三个字符可以是字母数字，组码必须是 0—9 或 A—F，此时 A—F 做字符对待。

在信息标识中用 nn=00 时要特别小心，如果要监控这类信息它表示要监控这组所有的信息。详细请看 8.3 的内容。

7.2.2 定义信息和信息帮助

可用 ADDMSGD 命令定义二级信息。信息正文是必须的，也要标识发生 CL 信息的条件。信息帮助是可选的，它进一步解释这个条件或解释要采取的改正动作。要得到信息帮助，工作站用户要在信息显示时，把光标移到信息上接按帮助键。帮助信息可用三个格式控制符来为工作站格式化，这些字符使用户更容易读帮助信息。

这三个控制字符必须用空格与信息上下文分开。

&NЬ（此时Ь为空格）：正文在新行开始显示（第 2 列）。如果正文多于一行，那么下一行从第 4 列开始直到结束，或遇到另外一个控制符。

&PЬ：从新行开始正文，从第 6 列开始。如果正文多于一行，下行从第 4 列开始直到结束或遇到另一控制符。

&BЬ：从新行第 4 列开始，如果正文多于一行，下行从第 6 列开始直到结束或遇到另一控制符。

7.2.3 分配一个严重码

在 ADDMSGD 命令中分配给信息的严重码指出信息的重要程序，严重码越高情况越严重。下面给出使用的严重码及其说明（它们的分配与 IBM 预先定义信息是一致的）。

00：信息，仅用于信息类。没发现错误，也不要回答，它指出功能正在处理或已完成。

10：警告，存在潜在的错误条件。过程和程序可取缺省值，例如提供丢失的输出，操作结果假定为成功。

20：错误，检查到一个错误，但可能提供自动恢复，继续处理。可能用缺省来代替遇到的输入，操作结果可能不对，或仅完成部分功能。例如，处理列表中的某些项因为其它项失败而导致错误。

30：严重错误，其错误程度不能做自动恢复，也没有可处理的缺省值。如果错误发生在源数据中，则跳过整个输入记录。如果在过程和程序处理期间发生错误，它导致过程和程序异常结束，操作结果是不正确的。

40：过程或功能异常结束。由于过程和程序不能处理无效数据，或用户取消过程，故操作结束。

50：作业异常结束，作业结束或没起动，过程步异常结束或启动失败，作业级请求不能按要求完成或取消作业。

60：系统状态。仅由系统操作员发布，它给出设备、子系统或系统的状态或警告信息。

70：设备完整性。仅由系统操作员发布。它指出正在管理设备或某些设备不可操作，用户可以从失败中恢复，也可请求服务人员帮助。

80：系统警告。它发布的是立即信息，它虽然不致于立即停止系统，这个警告条件如果不处理会使情况更严重。

90：系统完整性。仅由系统操作员发布，它给出一个子系统或系统不可操作的条件。

99：动作。需要人为干预。例如要回答信息，改打印格式或换软盘等。

对 SEV 参数的详细介绍，请看 CL 参考手册。

7.2.4 定义替换变量

在 ADDMSGD 命令中的 FMT 参数，可对第一或第二级信息规定替换变量。例如：

```
File  &1  not  found
```

其中&1 是替换变量，当显示或取得信息时，&1 要用没有找到的文件来代替，这个名字是由信息发送者提供的。例如：

```
File  DRDHDRP  not  fonud
```

它与 File not found 比较，替换变量使信息更多，规定更多含义。

替换变量必须以&开头后跟 n，n 是从 1—99 的数字。例如：File &1 not found 中的替换变量定义为 FMT((*CHAR 10))。在为替换变量分配替换变量时，必须从 1 开始连续分配，但也不一定要用所有的替换变量。例如：

```
File  &3  not  available
```

也是有效的。虽然在此信息中没有用&1 和&2，但此时必须在 ADDMSGD 中的 FMT 参数中定义&1、&2 和&3。FMT 可以是：

```
FMT((*CHAR 10)(*CHAR 2)(*CHAR 10))
```

此时，第一个值为&1，第二个为&2，第三个为&3。&1 和&2 必须在&3 前面。另外，在发送信息时，在 SNDPGMMSG 命令的 MSGDTA 参数要包括在 FMT 参数中说明的所有数据，要发送上面给出的信息，MSGDTA 参数必须至少 22 个字符长。

对前面的信息，也可如下规定 FMT 参数：

```
FMT((*CHAR 0)(*CHAR 0)(*CHAR 10))
```

由于在信息中没有用&1 和&2，可以把它们的长度规定为零，然后不发送信息。（这时，SNDPGMMSG 命令的 MSGDTA 参数仅要 10 个字符长）。

这种情况，也可用 DMPLST 参数来规定，它规定在往该监控它的程序发送逃逸信息时，DMPLST 参数规定的数据可以是空的。

替换变量在信息中规定的顺序不必须与 FMT 中规定的相同。例如，在 FMT 中规定 3 个值：

```
FMT((*CHAR 10)(*CHAR 10)(*CHAR 7))
```

在信息中使用的替换变量可以是：

```
object  &1  of  type  &3  in  labrary  &2  is  not  available
```

如果是用 CL 过程和程序发送信息，可用下列的信息数据连接这些值：

```
SNDPGMMSG .....MSGDTA(&OBJ *CAT &LIB *CAT &OBJTYPE)
```

必须在 ADDMSGD 命令中规定替换变量的信息数据字段的格式，即数据类型和长度，其中长度可选。信息数据字段的有效数据类型为：

*QTDCHAR：引号内的字符串。字符串要放在单引号内，不删除前导和末尾空白。如果在信息描述中没有规定长度，发送者确定字段的长度。

*CHAR：字符串。不用放在引号中，删去末尾空白。如果在信息描述中没规定长度，则发送者确定字段长。

*CCHAR：可转换为字符串。不用放在引号中，删去末尾空白，长度由发送者确定。如果这类数据送往 CCSID 而标志不是 65535 或 65534 的信息队列时，数据要从 CCSID 信息数据转换成 CCSID 信息队列。对这类数据在用接收或显示功能从信息队列得到数据时，也发生这种转换。处理 CCSID 的详细信息，请看应用开发的有关资料。

*HEX：十六进制。字符串前用 X，字符串放在单引号中。串中的每个字节都要转换成二个十六进制数。（0—9、A—F）。如果在信息描述中没有规定长度，则由发送者确定长度。

*BIN：二进制。格式为有符号的十进整数的一个二进整数（2 或 4 字节长）。如果没规定长度，则用 2 字节。

*DEC：十进制。格式为有符号有小数位和十进制的压缩十进制数。必须规定长度，小数位的缺省值为零。

*SYP：系统指针。系统目标的 16 字节指针，在一个信息或信息帮助中，10 个字符的目标名格式化为 CHAR 类型。

*SPP：空间指针。程序目标的 16 字节指针，在转储中，目标中的数据格式化为相同的十六进制类型数据。SPP 不能用作信息的转换变量，它仅能用做 ADDMSGD 中 DMPLST 参数的一部分。

下列数据类型仅用在 IBM 支持的信息描述中不能用作其它信息：

*ITV：时间间隔。8 字节，它包括等待超时条件的最接近的整秒数。

*DTS：日期和时间标志，8 字节的系统日期和时间标志，日期格式化成系统值 QDTAFMT 和 QDTASEP 规定的样子，时间的格式为 hh:mm:ss。

7.2.5 规定对回答的有效性检查

在 ADDMSGD 命令中，可对查询和通知信息规定对回答类型做有效性检查，可规定：

回答类型（TYPE）：

— 十进制（*DEC）

— 字符（*CHAR）

— 名字（*NAME）

回答的最大长度（LEN）：

— 对十进制，15 个数字（9 位小数）

— 对字符和字母，32 个字符

— 对名字，10 个字符

注：如果没规定任何有效性检查（VALUES，RANGE，REL，SPCVAL，DFT）对类型*CHAR 和*ALPHA 回答的最大长度为 132 个字符。

回答所能用的值：

— 值列表（VALNES）

— 特殊值（SPCVAL）

— 值的范围（RANGE）

— 回答值必须符合的一个简单关系（REL）

注：特别值是能接收的但不是其它有效性检查值能符合的值。

当显示工作站用户回答信息时，键盘在下档，这时进入的是小写字母。如果程序要求用大写字母，可做下面之一：

用 SNDUSR MSG 命令，要支持一个转换表能把小写转为大写

在 VALUES 参数中规定大写字符，它要求工作站用户只能输入大写字符

用 VALUES 规定大写字符且用 SPCVAL 把小写转换成大写

如果输入的是所有字母（A—Z），可规定 TYPE(*NAME)，它在检查时把字符转换成大写

7.2.6 发送立即信息和处理回答

用此例完成下列功能：

往 QSYSOPR 发送立即查询信息

要求回答 Y 或 N

保证有效的回答能输入进来

如果操作员在 120 秒之内没回答，做超时处理

```
            PGM
            DCL         &MSGKEY *CHAR LEN(4)
            DCL         &MSGRPY *CHAR LEN(1)
SNDMSG:     SNDPGMMSG   MSG('.... Reply Y or N') TOMSGQ(QSYSOPR) +
                          MSGTYPE(*INQ) KEYVAR(&MSGKEY)
            RCVMSG      MSGTYPE(*RPY) MSGKEY(&MSGKEY) WAIT(120) +
                          MSG(&MSGRPY)
            IF          ((&MSGRPY *EQ 'Y') *OR (&MSGRPY *EQ 'y')) DO
            .
            .
            GOTO        END
            ENDDO       /* Reply of Y */
            IF          ((&MSGRPY *EQ 'N') *OR (&MSGRPY *EQ 'n')) DO
            .
            .
            GOTO        END
            ENDDO       /* Reply of N */
            IF          (&MSGRPY *NE ' ') DO
            SNDPGMMSG   MSG('Reply was not Y or N, try again') +
                          TOMSGQ(QSYSOPR)
            GOTO        SNDMSG
            ENDDO       /* Reply not valid */
        /* Timeout occurred */
            SNDPGMMSG   MSG('No reply from the previous message +
                          was received in 120 seconds and a 'Y'' +
                          value was assumed') TOMSGQ(QSYSOPR)
```

```
                .
                .
END:            ENDPGM
```

在此例中，不能用 SNDUSRMSG 来代替 SNDPGMMSG，因为它不支持超时处理。（它要等待回答或取消作业）。

SNDPGMMSG 命令发送信息且规定 KEYVAR 参数，它返回一个信息引用键，它唯一标识这个信息，因此正确的回答能与 RCVMSG 命令匹配。KEYVAR 必须定义为长度为 4 的字符字段。

RCVMSG 命令规定从 SNDPGMMSG 的 MSGKEY 参数来的信息引用键，用来接收特别的信息，回答传给 MSG 参数，WAIT 规定等待多长时间回答才超时。

在接收回答时，过程检查回答的 Y 和 N。通常由操作员输入下个小写的 y 和 n。如果输入的是 Y、N 以外的非空白值，过程会发出不同信息然后重复要求查询信息。如果操作员输入的是空格，没有回答送给过程，返给过程空白，则发生超时，（按操作员不回答处理）。过程会给操作员发送信息，指出没收到回答，因此用缺省值 Y，由于这个缺省回答没显示，就不能确定什么时候看信息队列来确定信息回答与否及是否超时，信息一旦送出，过程不从信息队列取消它，第二个信息将缩减这种关系，且给出这种情况下的跟踪报告。

如果发生超时且操作员回答了信息，则忽略此回答，操作员接收不到忽略回答的指示。

7.2.6.1 发送 DBCS 立即信息

要发送 DBCS 正文的立即信息，正文限制在 37 个双字节字符再加上转换控制符，它保证能适当的显示信息。

7.2.7 定义回答的缺省值

ADDMSGD 命令允许对回答规定缺省，缺省回答必须符合有效性检查的值或规定为特殊值，在告诉用户（用 CHGMSGQ 命令）对所有送往用户信息队列的查询信息要用缺省回答时，用此缺省值，在删除没有回答的查询信息时也用缺省的回答。例如，用户用 DSPMSG 命令显示信息，用 F13 键取消没回答的查询信息来删除所有信息或用 F11 键来删除某个信息。

在把作业属性的 INQMSGRPY 设为*DFT 时用缺省回答，在设置*SYSRPYL 选项时也用缺省回答，可用系统回答列表来修改回答的缺省值。

7.2.8 对逃逸信息规定缺省的信息处理

对每个生成的做为逃逸信息发送的信息，可以设置一个缺省信息来处理在发送此信息且没有其它方式处理时采取的动作。

缺省信息处理动作可由下列内容组成：

缺省程序名：可调用它来采取缺省动作。下列参数可传送给缺省程序：

- 调用信息队列名，这个参数由多个字段组成，标识信息发送到哪去，有关内容请看系统 API 参考一书。
- 信息引用键（4 个字符），逃逸信息的引用键是在调用信息队列中。

转储清单：信息数据字段号的清单（与替换变量同样号码）。它指出转储了哪些目标，另外，可以转储下列内容：

- 作业的数据区
- 作业内部机器数据结构

— 作业

作业的转储清单用 DSPJOB 规定 JOB(*)　　QUTPUT(*PRINT)来产生的。

如果在信息描述中没规定缺省动作，将收到一个作业转储清单。(假定规定了 DSPJOB JOB(*)　QUTPUT(*PRINT))。

仅在完成信息过滤后没处理逃逸信息时才使用规定的缺省动作。详细内容请看 8.3.1。

7.2.8.1　缺省程序例子

下面的程序是缺省程序的例子，用在发送给这个信息后跟一个逃逸信息，它可以是 OPM CL 程序，也可以是 ILE 程序。

```
            PGM         PARM(&MSGQ &MRK)
            DCL         VAR(&MRK) TYPE(*CHAR) LEN(4)
            DCL         VAR(&MSGQ) TYPE(*CHAR) LEN(6381)
            DCL         VAR(&QNAME) TYPE(*CHAR) LEN(4096)
            DCL         VAR(&MODNAME) TYPE(*CHAR) LEN(10)
            DCL         VAR(&BPGMNAME) TYPE(*CHAR) LEN(10)
            DCL         VAR(&BLANKMRK) TYPE(*CHAR) LEN(4) VALUE(' ')
            DCL         VAR(&DIAGMRK) TYPE(*CHAR) LEN(4) VALUE(' ')
            DCL         VAR(&SAVEMRK) TYPE(*CHAR) LEN(4)
            DCL         VAR(&MSGID) TYPE(*CHAR) LEN(7)
            DCL         VAR(&MSGDTA) TYPE(*CHAR) LEN(100)
            DCL         VAR(&MSGF) TYPE(*CHAR) LEN(10)
            DCL         VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
            DCL         VAR(&OFFSET) TYPE(*DEC)
            DCL         VAR(&LENGTH) TYPE(*DEC)

            /* 检查 OPM 程序类型                                    */

            IF          (%SST(&MSGQ 277 1) *EQ '0') THEN(DO)
              CHGVAR      VAR(&QNAME) VALUE(%SST(&MSGQ 1 10))
              CHGVAR      VAR(&MODNAME) VALUE('*NONE')
              CHGVAR      VAR(&BPGMNAME) VALUE('*NONE')
             ENDDO
            ELSE DO
            /* 不是 OPM 程序，允许使用长过程名                      */
              CHGVAR      VAR(&OFFSET) VALUE(%BIN(&MSGQ 278 4))
              CHGVAR      VAR(&LENGTH) VALUE(%BIN(&MSGQ 282 4))
              CHGVAR      VAR(&QNAME) VALUE(%SST(&MSGQ &OFFSET &LENGTH))
              CHGVAR      VAR(&MODNAME) VALUE(%SST(&MSGQ 11 10))
              CHGVAR      VAR(&BPGMNAME) VALUE(%SST(&MSGQ 1 10))
             ENDDO
GETNEXTMSG: CHGVAR      VAR(&SAVEMRK) VALUE(&DIAGMRK)
            RCVMSG      PGMQ(*SAME (&QNAME &MODNAME &BPGMNAME)) +
                          MSGTYPE(*DIAG) RMV(*NO) KEYVAR(&DIAGMRK)
```

```
IF          (&DIAGMRK *NE &BLANKMRK) THEN(GOTO GETNEXTMSG)
ELSE DO
    RCVMSG      PGMQ(*SAME (&QNAME &MODNAME &BPGMNAME)) +
                MSGKEY(&SAVEMRK) RMV(*NO) MSGDTA(&MSGDTA) +
                MSGID(&MSGID) MSGF(&MSGF) MSGFLIB(&MSGLIB)
    SNDPGMMSG  MSGID(&MSGID) MSGF(&MSGLIB/&MSGF) +
                MSGDTA(&MSGDTA) TOPGMQ(*PRV (&QNAME +
                &MODNAME &BPGMNAME))
  ENDDO
ENDPGM
```

程序用 FILO 顺序接收所有的诊断信息，然后它发送最后一个诊断信息作为逃逸信息，允许前面的程序监控它。

7.2.8.2　规定报警选项

在 ADDMSGD 命令中，可规定报警选项，它允许信息生成报警，这个信息能生成 SNA 报警并发送问题管理焦点，生成的信息报警可用 ADDALRD 来定义。详细内容请看 DSNX 支持一书。

7.2.9　信息描述的例子

下例中，ADDMSGD 生成一个信息用在定货输入的应用程序中。在输入的客户号没有找到时发出此信息，信息为：

customer　number　&1　not　found

所用的 ADDMSGD 命令为：

```
ADDMSGD   MSGID(USR4310) +
          MSGF(QGPL/USRMSG)  +
          MSG('Customer number &1 not found') +
          SECLVL('Change customer number') +
          SEV(40) +
          FMT((*CHAR 8))
```

此信息加到 QGPL 库中的 USRMSG 文件中。

可用 DSPMSGD 或 WRKMSGD 命令打印或显示信息描述。

SECLVL 参数给出很简单的正文，要使它出现在附加的信息显示上，可以规定 SECLVL（‘message　text’）。在把光标放在信息下按帮助键时，会出现这里规定的正文。

7.2.10　定义 DBCS 信息

要定义正文为 DBCS 字符的信息，用 ADDMSGD 命令写一段 CL 过程和程序，把定义的信息放在信息文件中然后正常发送。在写程序 时，按下列步骤做：

1、源文件要能处理 DBCS 字符，在 CRTSRCPF 命令中规定 IGCDTA(*YES)。

2、用 SEU 输入源程序，有 DBCS 的 CL 命令只能通过 SEU输入。由于这个原因，双字节信息必须在 CL 程序中生成。

3、DBCS 信息限制长度为 37 个字符。这样才能显示或打印完整的信息。在使用 MONMS 命令时，CMPDATA 参数限制为 6 个双字节字符。

4、如果双字节信息文件代替一个字母数字信息文件（这样转换的信息文件只能送往双字节工作站），要用类似下面的命令来替换字母数字信息文件。

```
OVRMSGF MSGF(QCPFMSG) TOMSGF(DBCSLIB/QCPFMSG)
```

双字节信息只能在双字节显示工作站上显示。

7.3 系统信息文件的检索

在从信息文件取得一个信息时，系统用下列两步来检索：

1、系统完成有效的信息文件名替换。

2、如果没有替换信息文件名，在使用信息时，用信息文件名和库名来检索。

7.3.1 检索信息文件

当没有替换信息文件时，用规定的信息文件名和库名（在发送信息文件时）来从信息描述中检索信息文件。

在替换了信息文件名但在替换文件中没有信息标识时，也用规定的信息名和库名来检索信息文件。

系统根据是否规定了库名（*USRLIB 或*LIBL）来检索，下面说明对*CURLIB 和*LIBL 检索的路径：

规定*CURLIB 或规定一个库名：系统规定库中的信息文件名或作业的当前库。

规定*LIBL：在作业的库列表中检索信息文件名，在找到第一个信息文件后停止检索。

如果找到了信息文件，但对信息标识没有描述，则用 QCPFMSG 中的 CPF2457 的属性和正文来做为信息描述。

如果没找到信息文件，系统试图从此时发送的信息文件中取得信息。

注：可以找到一个信息文件，但由于损坏或权限方面的问题不能访问。

7.3.2 替换信息文件

可在过程和程序中替换信息文件，生成、删除、显示信息文件的替换与其它类型的替换很类似，区别仅是替换信息文件的名字，而不是属性，而替换的原则有明显的区别。

要替换一个信息文件，用 OVRMSGF 命令，被替换文件在 MSGF 参数中规定，替换文件在 TOMSGF 参数中规定。

例如，要用 USRMSGF 替换 QCPFMSG，使用下列命令：

```
OVRMSGF MSGF(QCPFMSG) TOMSGF(USRMSGF)
```

在替换或显示预先定义的信息时，检索替换文件的信息描述。如果在文件中找不到信息描述，则检索被替换的文件。

下面是替换信息文件的基本原因：

给回答或转储清单提供缺省值，由于原信息描述不合适，故可以生成一个信息文件带有修改缺省回答和转储清单的信息描述，可以建立几个操作环境，每个有不同的缺省回答。

修改信息的严重级别

提供缺省程序

修改信息正文。如果正文为空，对用户来说即无发送信息。例如，不能用 CPYF 命令指出发送给用户的信息。

为把信息传送给不同语言，用英文写的信息文件能用其它语言写的信息文件替换。（如果修改了所有信息，可用在库列表中修改其位置，而不用替换文件）。

选择信息文件取得信息的另一个方法是修改文件所在库在库列表中的位置。如果用这种方法，在找到第一个匹配文件后就停止检索。如果信息没在文件中，检索停止。

例如：假定信息文件 USRMSG 在库 USRLIB1 中，另一个信息文件 USRMSG 在 USRLIB2 中，要用在 USRLIB1 中的信息文件，那么在库列表中 USRLIB1 要放在 USRLIB2 前。系统检索正确名字的第一个信息文件，如果文件不包含信息，停止检索，如果用 OVRMSGF 命令，系统检索替换文件。如果没信息，它检索被替换的文件。

7.3.2.1 替换信息文件的例子

假定要修改一个作业中所用的 IBM 支持的信息，比如把信息 CPC2191

```
Object XXX in YYY type *ZZZ deleted
```

改为：

```
Object XXX in YYY deleted
```

在显示 CPF2191 的详细描述中可看到 FMT 参数是如何规定的。

首先要生成一个信息文件：

```
CRTMSGF MSGF(USRMSG/OVRCPF)
```

然后用 CPC2191 做为基本信息，把它加到信息文件中：

```
ADDMSGD  MSGID(CPC2191) MSGF(USRMSG/OVRCPF) +
         MSG('Object &1 in &2 deleted') +
         SEV(00) FMT((*CHAR 10) (*CHAR 10))
```

在运行作业时，用 OVRSMGF 命令替换信息文件：

```
OVRMSGF MSGF(QCPFMSG) TOMSGF(USRMSG/OVRCPF)
```

如果要修改此信息让它可用在所有作业中，可用 CHGMSGD 命令，而不用替换系统信息文件。如果用此命令修改了 IBM 提供的信息，那么在安装新版本系统时要重新修改。要再修改信息，可把修改放在输入流中或放在可在任何时候都能运行的程序中，也可替换信息文件，可在作业运行时使用下列命令：

```
OVRMSGF  MSGF(MSGFILE1)  TOMSGF(MSGFILE2)
OVRMSGF  MSGF(MSGFILE2)  TOMSGF(MSGFILE3)
```

首先，MSGFILE1 被 MSGFILE2 代替，其次，MSGFILE2 被 MSGFILE3 代替。当发送信息时，文件出现的顺序是：

1、MSGFILE3

2、MSGFILE2

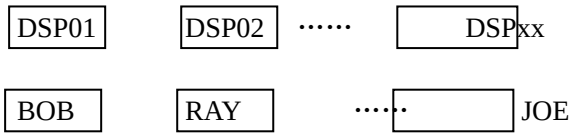
3、MSGFILE1

可保护信息文件不被替换，即在 OVRMSGF 命令中规定 SECURE 参数。

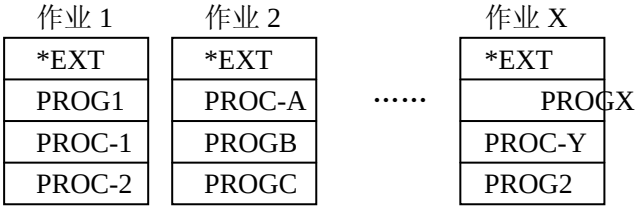
7.4 信息队列的类型

系统中的信息都是送往信息队列中，与这个信息队列相关的系统用过程或程序从信息队列接收信息。同样，对信息的回答送回到请求回答的用户或程序的信息队列中。

下图给出 IBM 支持的信息队列。每个显示工作站（DSP01 和 DSP02）和每个用户配置文件（BOB 和 RAY）都有信息队列：

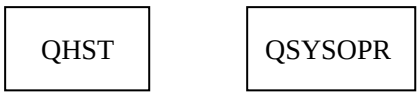


每个在系统中运行的作业都有作业信息队列，对每个作业分配一个外部信息队列（*EXT），在作业中每个 OPM 程序或 ILE 过程调用都有自己的调用信息队列。



外部信息队列调用信息队列。

系统历史日志（QHST）和系统操作员（QSYSOPR）也有信息队列：



这些信息队列的用途：

工作站信息队列用来在工作站用户之间及工作站与系统操作员之间接收和发送信息，队列的各字与工作站同名。在工作站介绍给系统时系统自动建立这些队列。

用户配置文件信息队列用在用户之间通讯。在生成用户配置文件时自动在 QUSRSYS 库中生成。

作业信息队列用来接收处理的请求，发送处理请求的结果信息，信息送往作业的请求者。每个作业都有信息队列，且仅对作业的文件存在。它由外部信息队列和一组调用堆栈项组成。

QSYSOPR 用来接收和回答系统信息，显示工作站用户和应用程序的信息。

历史日志用来发送系统中作业的历史日志信息。

除这些信息队列以外，你也可以生成自己的信息队列来给系统用户及应用程序之间发送信息。

7.4.1 生成或修改信息队列

要生成自己的信息队列，用 CRTMSGQ 命令，也可用 CHGMSGQ 修改你自己信息队列的以下属性：

对信息队列的修改是否立即存盘，立即存盘保证在系统失败时不丢失信息，但这样会降低系统性能。

信息交付信息队列的方式，在生成信息队列时，到达方式定义为保持。在显示工作站注册时，用户信息队列设为在用户配置文件中规定的方式。在 CHGMSGQ 命令中可以规定的交付类型为：

—中断：作业中断、调用一个程序来交付信息。如果用户程序没在 CHGMSGD 命令中请求中断或规定为*SAME，则 DSPMSG 自动显示此信息，作业的中断信息可由 CHGJOB 命令中的 BRKMSG 参数控制。

—通知：用引起注意灯或报警方式通知用户在队列中有信息用户，然后可用 DSPMSG 命令看信息。

—保持：信息放在队列中，直到用户用 DSPMSG 命令显示。

—缺省交付：所有信息都被忽略，需要回答的信息用缺省值回答。

对中断信息的处理：

—自动运行 DSPMSG 命令，对交互作业，如果严重码足够高则显示在屏幕上。对批作业，如果严重码足够高，则列在打印机的假脱机文件中。

—调用中断处理程序，必须用 CHGMSGD 命令规定要调用的程序，且把交付方式定为中断。

中断和通知信息过滤器的严重码：严重码等于或大于规定的最小值的信息都要显示。在生成队列时，最小严重码为 00，要修改它可用 CHGMSGQ 命令。

在用 DSPMSG 显示信息队列中的信息时，SEV 参数（严重码过滤器）用来过滤要给出的信息。用的过滤器不是在生成信息队列时规定的严重码过滤器，要用这个过滤器应规定 DSPMSG SEV(*MSGQ)。可用 DSPMSG 命令来确定用于中断和通知信息的当前严重码，它在显示的标题行给出。

与信息队列有关的 CCSID（编码字符集标识）：送往队列的信息转换为这个 CCSID。如果信息队列的 CCSID 是 65534 或 65535 就不做转换。如果信息队列 CCSID 是 65534，每个信息包含由发送者建立的自己的 CCSID。

标准信息队列的允许报警：允许报警规定在往生成时允许报警的队列发送报警信息时是否产生报警结果。

注：在生成工作站设备描述时，系统会为它建立信息队列来接收所有与设备有关的信息。

对工作站打印机、带设备及 APPC 设备，在生成设备描述时，用 MSGQ 参数来规定信息队列。如果对这些设备没有规定信息队列，用缺省值 QSYSOPR，所有其它设备在生成时都分配 QSYSOPR。

在用户配置文件中定义的信息队列，做为用户信息队列。在用这个用户配置文件注册系统时，用户信息队列设为在用户配置文件中规定的交付方式。如在某个工作站注册时交付方式是中断或通知，然后又注册了另一个工作站，那么并不改变新注册的交付方式，用户信息队列（以及工作站信息队列和 QSYSOPR）也不必对不同作业来修改中断或通知的方式。在注销或作业非正常结束时，如果此作业的方式为中断或通知，则用户信息队列的方式改为保持。

在转换为替换作业时做这种改变。可用 TFRSECJOB 命令或系统请求功能来切换替换作业。在切换到替换作业后，用你的用户配置文件注册，信息队列的方式为在用户配置文件中规定的方式，这就允许用户信息队列转换到替换作业，然后可在两个作业之间做切换，且有自己的信息队列。

假如切换到替换作业后，用另外的用户配置文件注册，原作业的用户信息队列变为保持方式，现注册的方式为用户配置文件中规定的方式。这样，你的用户信息队列可由另外的用户放在中断或通知方式。如果另外的用户在你切换到第一个作业时仍旧有你的用户信息队列交付方式，那么你的用户信息队列的方式不能改回原有的交付方式。

QSYSOPR 是系统操作员信息队列，上面讲述的情况也能在此信息队列上发生。

7.4.1.1 中断处理程序

在一个信息到达信息队列时且它的严重码大于或等于队列的严重码，而队列是在中断方式时，调用中断处理程序。要请求一个中断处理程序，必须在同一个 CHGMSGD 命令中规定程序名和中断方式。信息管理程序必须用 RCVMSG 命令接收信息，这样才能标识处理的信息，以便不再调用程序，详细内容请看第八章。

注：如果中断程序等待从显示设备上的输入数据，则此程序不能打开显示文件。

可用系统回答列表来规定系统对预告定义的查询给出回答。这样，就不用显示工作站用户回答，详细内容请看 8.6。

7.4.1.2 修改交付方式的例子

在启动系统时，当控制子系统启动时，会把 QSYSOPR 队列设为中断方式，但系统操作员注销，方式会改为保持，再注册系统操作员时，方式为在 QSYSOPR 用户配置文件中规定的方式。

下面是在 CL 初始化程序中的过程，用来把 QSYSOPR 信息队列设为中断方式。初始程序能用类似的过程来监控不是在用户配置文件规定的那个信息队列。

```
PGM /* 把信息队列设为中断方式的过程            */
CHGMSGQ  QSYSOPR DLVRY(*BREAK) SEV(50)
MONMSG   MSGID(CPF0000) EXEC(SNDPGMMSG MSG('Unable to put QSYSOPR +
         message queue in *BREAK mode') TOPGMQ(*EXT))
ENDPGM
```

它把 QSYSOPR 信息队列设为中断方式且严重级别为 50。如果不成功，有信息送往外部作业信息队列（*EXT）。当包含这个过程的程序结束时，显示初始菜单。严重级别 50 用来减少中断用户作业的信息数量，失败的一般原因是另外用户已经把 QSYSOPR 设成了中断

方式。

7.4.2 作业信息队列

系统中的每个作业都有作业信息队列来处理作业需要的所有信息。一个作业的信息队列由一个外部信息队列（*EXT）和一组调用信息队列组成。一个调用信息队列分配给作业中被调用的每个 ILE 过程或 OPM 程序。另外，对每个作业都生成一个作业日志，作业日志是一个逻辑队列，它按年月顺序管理作业中发送的所有信息。你可往外部信息队列和调用信息队列发送信息，但不能往作业日志中发送信息，送往外部信息队列或调用信息队列的信息可由系统逻辑地加到作业日志中。

外部信息队列用来与作业外部的请求者（例如工作给用户）通讯，送往外部信息队列的信息（除状态信息外）也放在作业日志中。（详细内容请看 8.7.1）。

如果一个消息查询或通知信息送往交互作业的外部信息队列，信息在显示程序信息屏上显示，且过程等待显示工作站用户回答或通知。如果用户没有回答或用 F3 或执行键，则用缺省回答返给发送者。如果没有缺省回答，则发送*N。如果查询或通知信息送往批作业的外部信息队列，则把缺省回答送给发送者，如果没有缺省信息，则用*N，系统的回答列表可以替换查询显示或对*EXT 的缺省回答。

如果一个状态信息发送给交互作业的外部信息队列，信息显示在工作站的信息行上。可用这样的状态信息来指示用户长时间运行。例如在用 CPYF 把 n 个成员复制成一个文件时，系统会发送状态信息。

注：在应用程序完成长时间运行操作时，必须发送另外的信息来清除显示中的信息行。可用信息 CPI9801 来做这件事。

例如：

```
PGM
.
.
.
SNDPGMMSG  MSGID(CPF9898)  MSGF(QCPFMSG)  MSGDTA('Status 1')  +
           TOPGMQ(*EXT)  MSGTYPE(*STATUS)
.
.
.
SNDPGMMSG  MSGID(CPF9898)  MSGF(QCPFMSG)  MSGDTA('Status 2')  +
           TOPGMQ(*EXT)  MSGTYPE(*STATUS)
.
.
.
SNDPGMMSG  MSGID(CPI9801)  MSGF(QCPFMSG)  TOPGMQ(*EXT)  +
           MSGTYPE(*STATUS)
.
.
.
ENDPGM
```

一个调用信息队列用来在程序或过程之间发送信息，只要一个过程和文件在调用堆栈中（还没有返回），那么它的调用信息队列就是活动的，且信息能发送给过程和文件。一旦过程和文件返回，它的调用信息队列就不再存在，即不能往它那里发送信息。能往调用信息队列发送的信息类型包括消息、查询、完成、诊断、状态、逃逸和通知。

在过程和文件被调用时建立调用信息队列，它仅与过程和文件运行的调用堆栈入口有关。调用信息队列与调用堆栈入口标识一致。一个调用堆栈入口是由在其运行的过程和文件名来标识的。

在 OPM 程序情况下，相关的调用堆栈入口是由最多 10 个字符的程序名标识的。在 ILE 过程情况下，入口是由三部分组成的，即 256 个字符的过程名，10 个字符的模块名和 10 个字符的程序名，模块名是组成编译过程的模块名字，ILE 程序名是连接模块的 ILE 程序名。

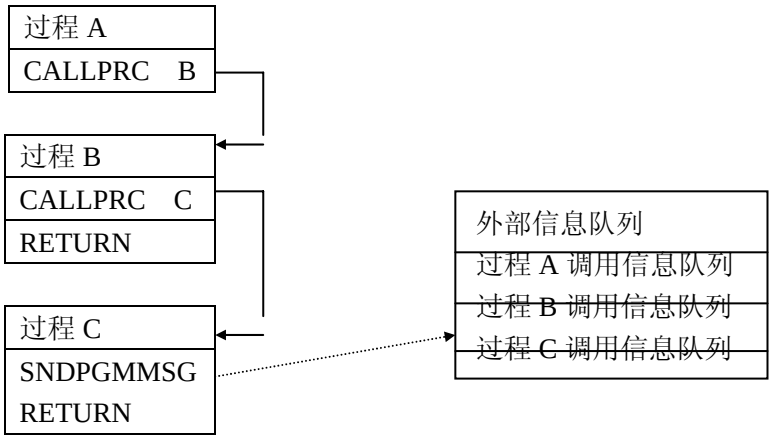
在标识 ILE 过程的调用堆栈入口时，仅规定过程名就足够了。如果过程名不能唯一地标识入口，也可规定模块名或程序名。如果在发送信息时，过程和文件是不只一次的在调用堆栈中，规定的名字将标识最近一次调用的过程和文件。

也有其它方法来标识调用堆栈入口，详细内容请看 8.2.3。

假如编译了的 OPM 或 ILE 在调用堆栈中被代替了，在用程序名来引用一个调用堆栈入口时要特别小心，对所有先在堆栈中的调用堆栈入口而不是指向所做替换操作的那些入口，名字引用将分解为在 QRPLOBJ 库中的被替换目标。这名字引用是合法的，直到被替换目标继续留在 QRPLOBJ 库中。对堆栈中较新的入口，它指向所做的替换操作，名字的引用是对程序的新版本，由于使用的方法是确定的，所以不要把程序直接放在 QRPLOBJ 库中，这个库严格做程序版本的替换操作，对直接放进去的程序做名字引用会失败。

如果程序正在调用堆栈中发生移出程序或改名，那么任何用老名字做的引用都会失败，对 ILE 过程，假如仅用过程名和模块名做引用，对程序改名不会影响名字引用，如果也用 ILE 程序名，则名字引用会失败。

过程和程序的调用堆栈入口信息队列在过程和程序结束时就不能使用了，在相应的调用信息队列中的信息仅能用信息引用键来引用。例如，在下图中，假定过程 A 调用过程 B，过程 B 调用过程 C，过程 C 发送给过程 B 一个信息后结束。信息对过程 B 是可用的，但在 B 结束时，它的调用信息队列不再可用，且不能访问它，即使在作业日志中给出的信息也是如此，A 不能访问送给 B 的信息，除非 A 对此信息有信息引用键。

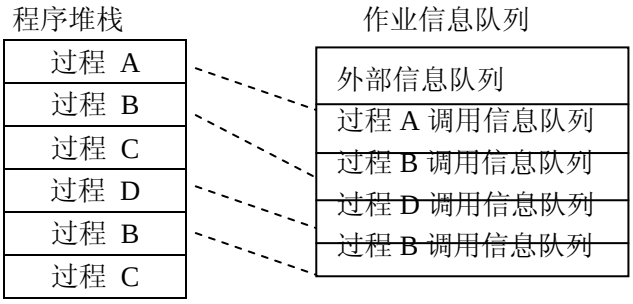


假如 A 要某个信息，必须按下列方法做：

C 向 A 发送某个信息

B 向 A 再发送这个信息

下图给出过程调用作业信息队列和调用堆栈入口队列之间的关系，虚线（---）指出信息队列与之相关的过程调用。



如上图，过程 B 有两个调用堆栈入口队列，过程 C 没有信息队列。因为没有信息发送给 C，当 C 往 B 送信息时，信息到达最后一个对 B 调用的堆栈入口队列。

注：在使用命令入口显示时，可用 F10 键显示送往作业信息队列的所有信息。一旦显示了信息，可用翻页键上、下翻，也可用 DSPJOBLOG 命令显示作业信息。

第八章　处理信息

这章介绍在用户和程序之间用信息做通讯的一些方法，信息可以用如下方法发送：

从一个系统用户到另一个系统用户，信息的接收者当前没使用系统也可。

从一个 OPM 程序或 ILE 过程到另一个 OPM 程序或 ILE 过程。

从过程或程序向系统用户，信息的接收者当前没使用系统也可。

交互系统用户仅可发送立即信息和回答信息。

OPM 程序或 ILE 过程可发送立即信息或预先定义的有用户定义数据的信息，另外，过程或程序也能：

接收信息

从信息文件中取得信息描述且把它放到程序变量中

从信息队列中移出信息

监控信息

8.1　给系统用户发送信息

下列命令可用来向系统用户发送信息：

SNDMSG（发送信息）

SNDBRKMSG（发送中断信息）

SNDPGMMSG（发送程序信息）

SNDUSRMSG（发送用户信息）

SNDUSRMSG 和 SNDPGMMSG 可用在批及交互的 OPM 程序和 ILE 过程中，不能在命令行中输入。SNDMSG 往 QSYSOPR，显示工作站信息队列或用户信息队列中发送信息或

查询。

可以同时向多个信息队列发送查询信息，也可只向一个信息队列发送查询信息，信息是由信息队列规定的交付方式交付的，如果信息不是中断方式，就不能中断用户的当前操作。

下面的 SNDMSG 命令由显示工作站用户发送信息给系统操作员：

```
SNDMSG   MSG('Mount tape on device TAP1') TOUSR(*SYSOPR)
```

SNDBRKMSG 命令从工作站、程序或作业送一个立即信息给一个或多个工作站，它会以中断方式交付而不管接收的信息队列是什么方式。这个命令只能往显示工作站信息队列发送信息，可在需要工作站用户立即干预时用此命令，但不能保证此信息都会引起中断。因为每个作业可用 CHGJOB 命令中的 BRKMSG 参数来控制这件事。

如果发送查询信息，可规定回答返回给哪个信息队列而不是你的显示工作站。

下面的命令是由系统操作员给所有的工作站用户发送信息：

```
SNDBRKMSG MSG('System going down in 15 minutes')
          TOMSGQ(*ALLWS)
```

此命令把信息发送给所有用户，而不是此时正活动的用户。

8.2 从 CL 程序中发送信息

用 SNDPGMMSG 或 SNDUSRMSG 命令从 CL 过程或程序中发送信息。

用 SNDPGMMSG 命令可发送下列类型的信息：

消息、查询、完成、诊断、请求、逃逸、状态、通知。

可从 CL 过程或程序往下列类型的队列发送信息：

作业的外部信息队列（看 7.4.2）

程序的调用信息队列（看 7.4.2）

QSYSQPR 队列

工作站信息队列

用户信息队列

要从过程或程序中发送信息，在 SNDPGMMSG 命令中要规定下列内容：

信息标识或立即信息：信息标识是预先定义信息的信息描述的名字。

信息文件：在发送预先定义信息时，包含信息描述的信息文件的名字。

信息数据字段：如果发送预先定义信息，这些字段包含信息中替换变量的值。每个字段的格式化须在信息描述中说明。如果发送立即信息，就没有信息数据字段。

接收信息的信息队列或用户。

信息类型：下表给出哪类信息可以发送给哪类队列（ √=有效）

信息类型	外部	调用	QSYSOPR	工作站	用户
消息	√	√	√	√	√
查询	√		√	√	√
完成	√	√	√	√	√
诊断	√	√	√	√	√
请求	√	√			

逃逸		√			
状态	√	√			
通知	√	√			

CCSID：规定提供信息或信息数据所用的编码字符集标识。

回答信息队列：接收对查询回答的信息队列的名字，缺省为回答送往发送查询信息的过程或程序的调用信息队列。

键字变量名：包含信息引用键的 CL 变量名。

要往 7.2.9 生成的信息队列发送信息，可用下列命令：

```
SNDPGMMSG   MSGID(USR4310) MSGF(QGPL/USRMSG) +
            MSGDTA(&CUSNO) TOPGMQ(*EXT) +
            MSGTYPE(*INFO)
```

信息的替换变量是客户号，由于客户号可变，不能在信息中规定确切的客户号，那么，要在过程或程序中对客户号说明一个 CL 变量（&CUSNO），然后把它规定为信息数据字段。在发送信息时，变量的当前值传给信息。

```
customer  number  35500  not  found
```

另外，也不总是知道哪个显示工作站使用过程或程序，故在 TOPGMQ 参数中也不能规定确切的显示工作站信息队列。这样，可以规定外部信息队列*EXE。

8.2.1 信息

8.2.1.1 查询和消息

用 SNDUSRMSG 命令，可向显示工作站用户、系统操作员或用户定义的信息队列发送查询或消息。如果发送查询信息，过程或程序要等待用户的回答，信息可以是立即信息或预先定义的信息，对交互作业，信息用缺省值送往显示工作站用户，对批作业，信息用缺省值送往系统操作员。要用 SNDUSRMSG 发送信息，可在命令中规定以下内容：

信息标识或立即信息：信息标识是预先定义信息的信息描述的名字

信息文件：发送预先定义信息时，包括信息描述的信息文件的名字

信息数据字段：如果发送预先定义信息，这些字段包含信息中替换变量的值。每个字段的格式化须在信息描述中说明。如果发送立即信息，就没有信息数据字段

对查询信息有效的回答

信息类型

接收的信息队列

信息回答：一个 CL 变量，它包括对查询信息响应的回答

转换表：用来转换回答值，通常用来做大小写转换

CCSID：规定提供信息或信息数据所用的编码字符集

8.2.1.2 完成和诊断信息

SNDPGMMSG 命令，可发送诊断和完成信息，可发送给任何信息队列。诊断信息告诉调用过程或程序检查到的错误，完成信息告诉所做的工作结果。

通常，逃逸信息是送往过程或程序来告诉调用者有什么问题或发送诊断信息。对完成信

息，通常不发送逃逸信息，因为所要求的功能已经完成。例如，假定系统操作员用菜单来调用程序 SAYPAY 来保存某个目标，用下面的程序来保存然后发布完成信息：

```
PGM
SAVOBJ OBJ(PAY1 PAY2) LIB(PAYROLL) CLEAR(*YES)
SNDPGMMSG MSG('Payroll objects have been saved') MSGTYPE(*COMP)
ENDPGM
```

如果 SAVOBJ 失败，CL 功能检测且系统操作员要显示详细的信息来发布逃逸信息，以解释失败的原因。如果成功地完成 SAVOBJ，要往显示菜单程序的调用信息队列发送完成信息，完成信息要与 IBM 支持的命令一致。多数 IBM 命令会发送完成信息指出成功地完成，查看送往作业日志的信息类型会有助于做问题分析。

8.2.1.3 状态信息

可用 SNDPGMMSG 命令从 CL 过程或程序中往作业的调用信息队列或*EXT 发送状态信息。在送往调用信息队列时，接收的过程或程序能监控状态信息的到达及能处理描述的条件。如果不能监控这些信息，控制要返给发送者来重处理。请看 8.3 的说明。

8.2.1.4 逃逸和通知信息

可用 SNDPGMMSG 命令从 CL 过程或程序中往调用过程或程序的调用信息队列发送逃逸信息，它告诉调用者过程或程序非正常结束以及原因。调用者能监控逃逸信息的到达且处理发生的条件，当调用者处理这个条件时，控制不会返给逃逸信息的发送者。假如调用者是同一程序用的另外过程，程序本身不结束，发出逃逸的过程也可以继续。如果逃逸信息送往程序本身的调用者，所有程序中活动的过程都立即结束，结果是程序不能继续运行。如果调用者没有监控这个逃逸信息，那么采取缺省的系统动作。

可以从 CL 过程或程序往调用程序或过程的信息队列或外部信息队列发送通知信息，它告诉调用者继续运行的条件，调用过程或程序能监控这些信息的到达并处理它产生的条件。如果调用者是 ILE 过程，过程能处理这个条件返回一个回答然后让过程或程序继续进行。如果没有监控，则返回给发送者一个缺省值，然后发送者重新操作，请看 8.3

不能用立即信息做逃逸和通知信息，系统定义了信息 CPF9898，它用做立即的逃逸和通知信息。例如：

```
SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) MSGDTA('Error condition') +
          MSGTYPE(*ESCAPE)
```

8.2.2 发送信息的例子

例 1：此例允许显示工作站用户调用有这个过程的 CL 程序来提交作业而不用 SBMJOB 命令，在提交作业后会发送一个完成信息。

```
PGM
SBMJOB JOB(WKLYPAY) JOBD(USERA) RQSDTA('CALL WKLY PARM(PAY1)')
SNDPGMMSG MSG('WKLYPAY job submitted') MSGTYPE(*COMP)
ENDPGM
```

例 2：此例基于从被调用程序接收的参数来修改信息，然后由过程发送一个完成信息。

```
DCL &RCDCNT TYPE(*CHAR) LEN(3)
CALL PGMA PARM(&RCDCNT)
SNDPGMMSG MSG('PGMA completed' *BCAT &RCDCNT *BCAT +
          'records processed') MSGTYPE(*COMP)
ENDPGM
```

例 3：此过程发送信息请示系统操作员装入一个特别格式，RCVMSG 命令等待回答。系统操作员要给出至少一个字符做回答，但过程不使用这个回答的值。

```
PGM
DCL &MSGKEY TYPE(*CHAR) LEN(4)
SNDPGMMSG   MSG('Load special form') TOUSR(*SYSOPR) +
            KEYVAR(&MSGKEY) MSGTYPE(*INQ)
RCVMSG MSGTYPE(*RPY) MSGKEY(&MSGKEY) WAIT(120)
.
.
.
ENDPGM
```

在 RCVMSG 命令中必须规定 WAIT 参数，这样过程能等待回答。如果没规定 WAIT，过程继续执行 RCVMSG 后的指令，而不接收回答，在 RCVMSG 中使用的 MSGKEY 参数允许过程接收回答,在 SNDPGMMSG 中的变量&MSGKEY 返给过程用在 RCVMSG 命令中。

例 4：下面过程在批作业时把信息发送给系统操作员，作业在工作站运行时发送给工作站用户，过程接收大写或小写的 Y 或 N（小写由转换表（TRNTBL 参数）转成大写）。如果回答的不是这四种，操作员发送信息指出回答无效。

```
PGM
DCL &REPLY *CHAR LEN(1)
.
.
SNDUSRMSG MSG('Update YTD Information Y or N') VALUES(Y N) +
     MSGRPY(&REPLY)
IF (&REPLY *EQ Y)
 DO
 .
 .
 .
 ENDDO
ELSE
 DO
 .
 .
```

```
  ENDDO
  .
  .
  .
 ENDPGM
```

例 5：下面的过程使用 CPF9898 发送逃逸信息，信息内容为‘procedure detected failure’。立刻信息不能做逃逸信息，故 CPF9898 用做信息数据。

```
    PGM
    .
    .
    .
 SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)
       MSGDTA('Procedure detected failure')
    .
    .
    ENDPGM
```

例 6：下面的过程允许系统操作员给几个显示工作站发送信息。在系统操作员调用程序时，这个过程包括在被调用程序中，显示一个提示让系统操作员进入发送信息类型以及信息正文，过程连接时间、　　日期和信息正文。

```
PGM
DCLF WSMSGD
DCL &MSG TYPE(*CHAR) LEN(150)
DCL &HOUR TYPE(*CHAR) LEN(2)
DCL &MINUTE TYPE(*CHAR) LEN(2)
DCL &MONTH TYPE(*CHAR) LEN(2)
DCL &DAY TYPE(*CHAR) LEN(2)
DCL &WORKHR TYPE(*DEC) LEN(2 0)
SNDRCVF RCDFMT(PROMPT)
IF &IN91 RETURN /* Request was ended  */
RTVSYSVAL QMONTH RTNVAR(&MONTH)
RTVSYSVAL QDAY RTNVAR(&DAY)
RTVSYSVAL QHOUR RTNVAR(&HOUR)
IF (&HOUR *GT '12') DO
CHGVAR &WORKHR &HOUR
CHGVAR &WORKHR (&WORKHR - 12)
CHGVAR &HOUR &WORKHR /* Change from military time */
ENDDO
RTVSYSVAL QMINUTE RTNVAR(&MINUTE)
```

```
            CHGVAR    &MSG ('From Sys Opr ' *CAT &MONTH *CAT '/' +
                      *CAT &DAY +
                      *BCAT &HOUR *CAT ':' *CAT &MINUTE +
                      *BCAT &TEXT)
            IF (&TYPE *EQ 'B') GOTO BREAK
NORMAL:     SNDPGMMSG MSG(&MSG) TOMSGQ(WS1 WS2 WS3)
            GOTO ENDMSG
BREAK:      SNDBRKMSG MSG(&MSG) TOMSGQ(WS1 WS2 WS3)
ENDMSG:     SNDPGMMSG MSG('Message sent to display stations') +
                      MSGTYPE(*COMP)
            ENDPGM
```

用在程序中的显示文件 WSMSGD 的 DDS 如下：

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                      DSPSIZ(24 80)
     A          R PROMPT                    TEXT('Prompt')
     A                                      BLINK
     A                                      CA03(91 'Return')
     A                                   1  2'Send Messages To Work Stations'
                                            DSPATR(HI)
     A                                   3  2'TYPE'
     A            TYPE           1   1     +2VALUES('N' 'B')
     A                                      CHECK(ME)
                                            DSPATR(MDT)
     A                                     +3'(N = No breaks  B = Break)'
     A                                   5  2'Text'
     A            TEXT         100   1     +2LOWER
     A
     A
```

如果系统操作员在提示中输入：

```
B

please  sign  offf  by  3:30  today
```

则发送下面的中断信息：

```
from  sysopr  10/30  02:00  please  sign  off  by  3:30  today
```

8.2.3 SNDPGMMSG 的调用堆栈入口标识

如果 CL 过程往 OPM 程序或另外的 ILE 过程发送一个信息，你必须标识调用堆栈入口来指出信息发送到哪儿，信息会送往标识的调用堆栈入口的调用信息队列。

SNDPGMMSG 的 TOPGMQ 参数用来标识要发送的调用堆栈入口。标识由二部分组成：

规定基本入口：规定 TOPGMQ(*PRV　*)指出基本入口是一个在运 SNDPGMMSG 命令的过程中，偏移是对此基本入口的前一个入口，这个规定标识过的调用者正在使用这个命令。

规定基本入口的偏移：偏移是（TOPGMQ 的一个值）指明你是否给基本入口发送信息（*SAME）或给基本入口的调用者（*PRV）发送信息。

为了了解如何标识基本入口 TOPGMQ 的二个值，也需要了解 ILE 程序运行时的调用堆栈。用两个程序来做解释。程序 CL PGM1 是一个 OPM 程序，程序 CL PGM2 是一个 ILE 程序。既然 CL PGM2 是 ILE，它就能由几个过程组成，比如：CL PROC1、CL PROC2、CL PROC3，在运行时发生下列调用：

先调用 CL PGM1

CL PGM1 调用 CL PGM2

CL PGM2 调用 CL PROC1

CL PROC1 调用 CL PROC2

CL PROC2 调用 CL PROC3 或 CL PROC4

图 8-1 用来解释在 CL PROC2 调用 CL PROC4 时的调用堆栈结构：

在调用堆栈入口和 OPM 程序间是一一对应关系，对每一次的调用 OPM 程序，都有一个新的入口加到调用堆栈中。

一个 ILE 程序，做为一个单元，不在堆栈中出现，在调用 ILE 程序时调用的每个过程要往堆栈中加一个入口，结果是往 ILE 过程中送信息，而不是往 ILE 程序中送。

注：在调用 ILE 程序时，运行的第一个过程是程序的 PEP（程序入口例程）。在 CL 中，这个例程由系统生成（-CL-PEP），且调用你提供的第一个例程。在此例中，PEP 入口是在 CL PGM1 的入口和 CL PROC1 之间。

下面是规定基本调用堆栈入口的不同方法。

用命令做为基值的过程：

如果 TOPGMQ 参数规定了 TOPGMQ(*SAME　*)或 TOPEMQ(*PRV　*)。则使用 SNDPGMMSG 命令的过程入口用做基本入口，如果规定 TOPGMQ（*SAME　*），过程往其自身发送信息，如果规定 TOPGMQ(*PRV　*)，过程往调用者发送信息。

注：在过程用 TOPGMQ(*PRV　*)给调用者发送信息时要注意下列信息：

在 CL PROC4CL 和 PROC2 给调用者送回信息时，信息不留在后续程序中，信息在同程序中的过程之间发送。

如果目的是往程序的调用者发送信息，(此例中的 CL PGM1)，最好不要规定 TOPGMQ(*PRV　*)。

在 CL PROC1 发送信息给调用者时，跳过程序入口例程，即使调用得是 PEP 信息也送往 CL PGM1，在规定 TOPGMQ(*PRV　*)，PEP 入口不是可视的，且不包括在发送操作中，如果用其它方法规定 TOPGMQ，PEP 对发送者来说是可视的。

图 8-2 解释了在 CL PROC1、CL PROC2 和 CL PROC4 分别给过程调用者发回信息时的结果。

过程 -CL-PEP (在 CL PGM2 中)

图 8-1 运行时调用堆栈的例子

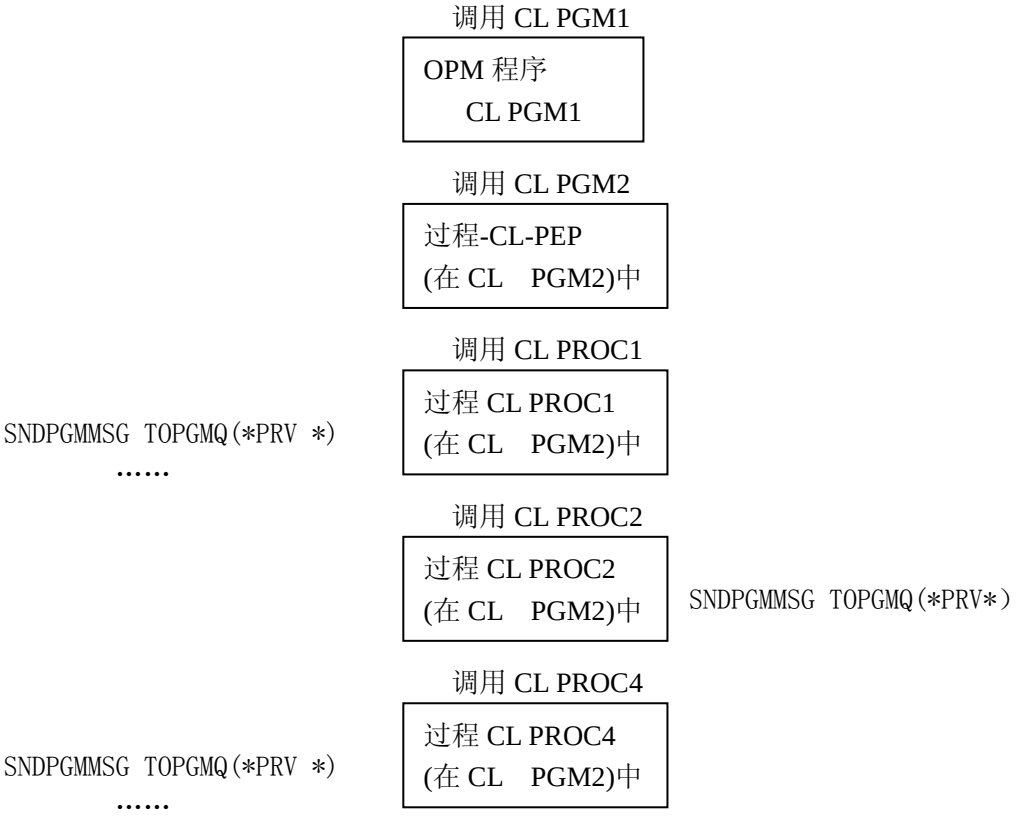


图 8-2 TOPGMQ(*PRV *)的例子

用名字标识基本入口：

能提供一个在此入口运行的 OPM 程序或 ILE 过程的名字 来标识基本入口。名字可以是简单名（一部分）或组合名（二部分或三部分），如下所述：

简单名：用来标识 OPM 程序或 ILE 过程

如果简单名是 10 个字符或少于 10 个字符，那么由系统确定它是 OPM 程序还是 ILE 过程。基本入口是最新用名字调用的 OPM 程序或 ILE 过程来标识，如果名字长度大于 10 个字符，那么系统确定它为 ILE 过程名，基本入口是用最新用名字调用的过程来标识，运行 OPM 程序的入口不做考虑。

图 8-3 是用简单名发送信息的例子，CL PROC4 发送信息给 CL PROC2，CL PROC2 往 CL PGM1 发送信息。

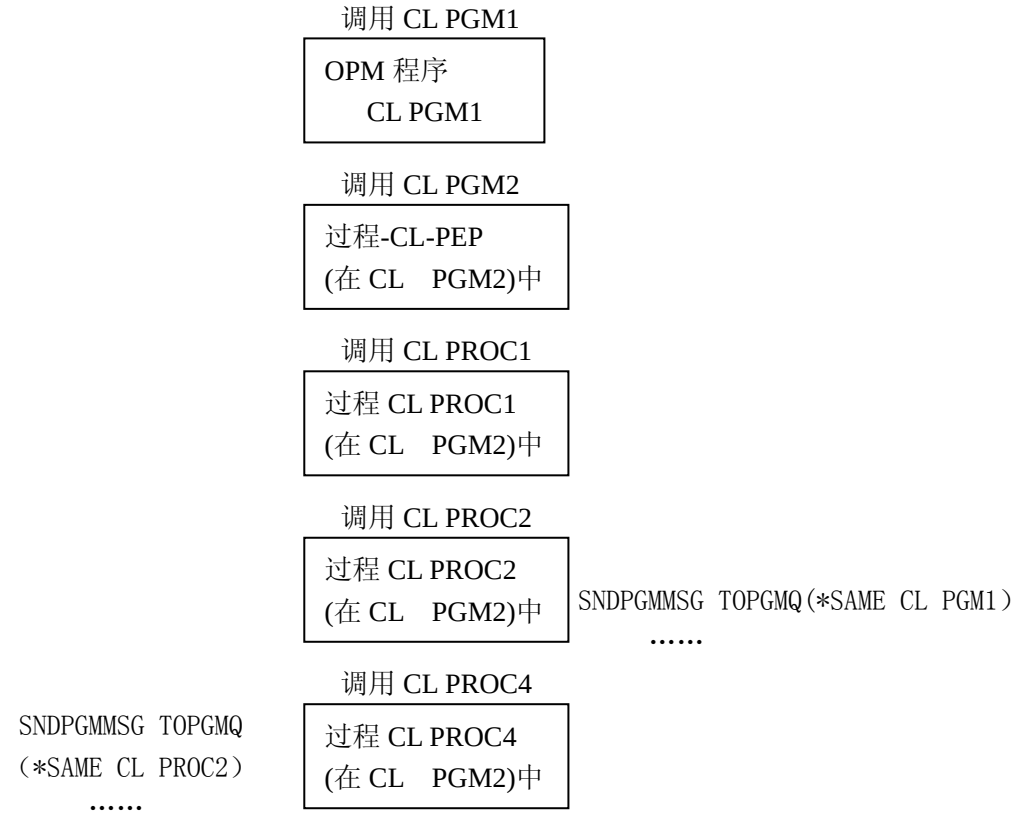


图 8-3　使用简单名的例子

组合名：组合名由二部分或三部分组成，它们是：

— 模块名，编译过程的模块名

— 程序名，连编过程的程序名

— 过程名

在要唯一标识信息送往的过程时，采用由下列部分组成的组合名：

— 过程名、模块名、程序名

— 过程名和模块名

— 过程名和程序名

如果是组合名，标识的基本入口不能由 OPM 程序运行。图 8-4 是用组合名发送息的例子。此例中，CL PROC4 用二部分的组合名往 CL PROC1 发送信息。

不用全 OPM 程序名或全 ILE 过程名，可以使用部分名，详细内容请看 CL 参考一书。

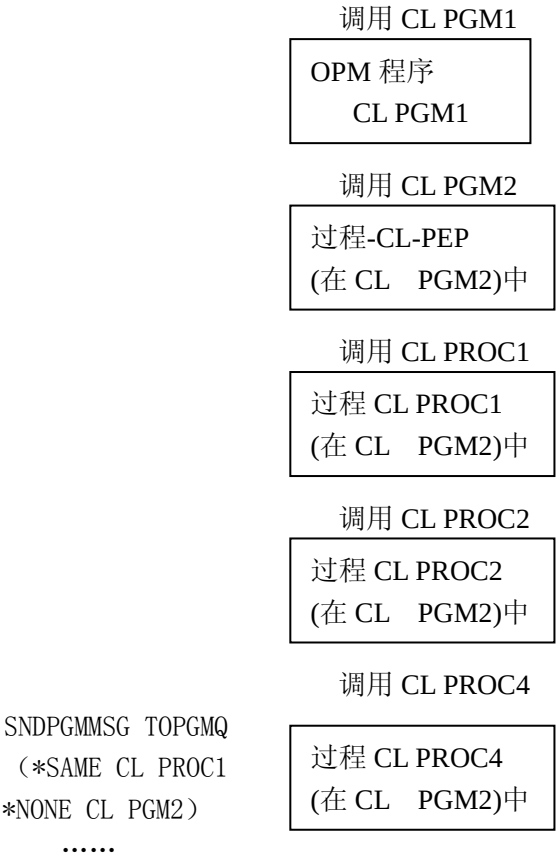


图 8-4　使用组合名的例子

程序边界做基本入口：

可用*PGMBDY 或程序名来标识一个 CL 程序的 PEP,标识的 CL 程序的 PEP 入口做为基本入口。在从程序边界外的过程中发送信息时，这个选项很有用。

图 8-5 是用*PGMBDY 发送信息的例子。此例中，CL PROC4 直接往 CL PGM1 发信息，而 CL PGM1 是调用 CL PGM2 的。CL PROC4 能做这个操作不用知道哪个程序调用 CL PGM2，也不知道发送信息过程与 PEP 的相关位置，用*PGMBDY 而不用一起规定程序名，意思是标识程序的边界是发送信息过程所在的程序。图 8-6 是用*PGMBDY 和一个程序名发送信息的例子，在图 8-6 中，使用下列过程或程序：

CL PGM1 和 CL PGM2，它们是在前面例子中定义过的。

CL PGM3，它是另外一个 ILE 程序。

CL PROCA 在 CL PGM3 中，从 CL PROCA 中往 CL PGM2 的调用者发信息。信息是从 CL PROCA 中用*PGMBDY 和 CL PGM2 往 CL PGM2 的调用者发信息。在此例中，如果规定 TOPGMQ(*PRV -CL-PEP)，信息会送往 CL PGM3 的调用者而不是 CL PGM2 的调用者。这是由于用名字调用的最新调用是 CL PGM3 的 PEP。

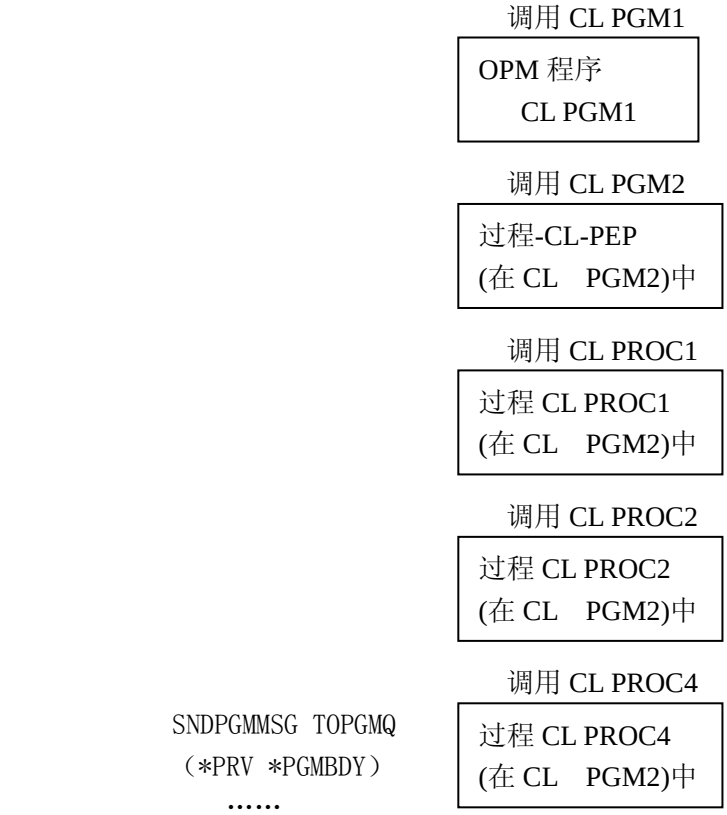
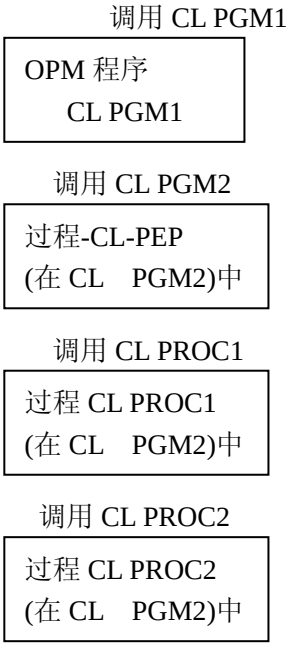


图 8-5　用*PGMBDY 的例子



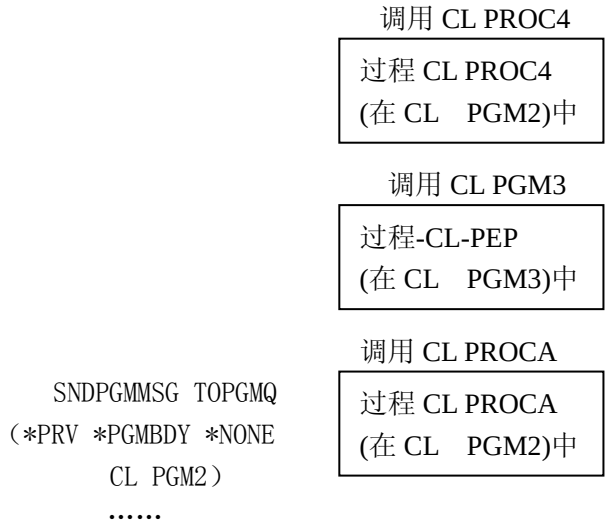
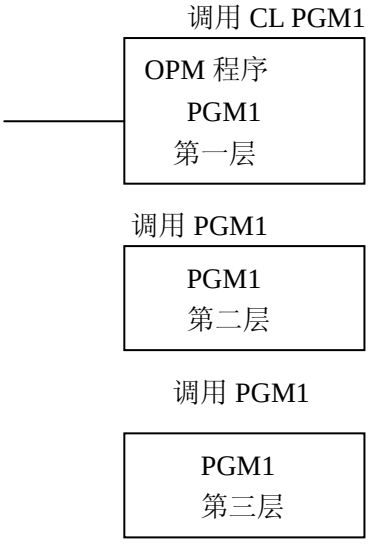


图 8-6　用*PGMBDY 的第二个例子

*PGMBDY 也能用在 OPM 程序中。如果与*PGMBDY 一起规定了一个 OPM 程序名，它的结果与仅用一个程序名是一样的，即 TOPGMQ(*SAME *PGMBDY *NONE　OPM 程序名)发送的信息与 TOPGMQ(*SAME　OPM 程序名)发送到同一个地方。

这种情况的一个例外是再发生往调用者本身的 OPM 程序发送信息。TOPGMQ(*SAME 程序名)是往最后一次再发生层送信息，但 TOPGMQ(*SAME *PGMBDY *NONE　程序名)是往第一次再发生层上送信息。图 8-7 给出怎样调用 PGM1 及如何重复二次以上再调用过程本身，在第三次中，PGM1 调用 PGM2，然后 PGM2 给 PGM1 发信息。如果程序仅用 PGM1 做名字发送信息，信息会到达 PGM1 的第三层。如果用*PGMBDY 和 PGM1 结合起来发信息，信息送到 PGM1 的第一层。



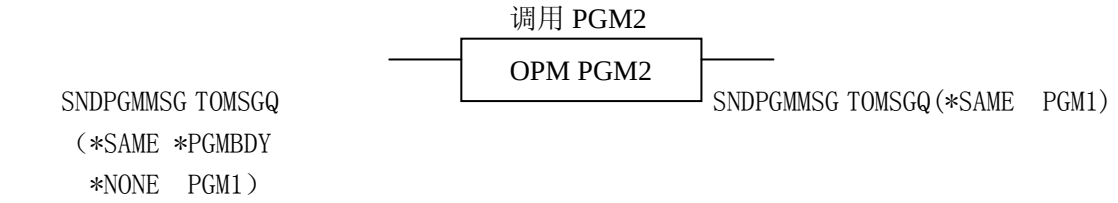


图 8-7　用*PGMBDY 的第三个例子

用最新调用的过程做基本入口：

虽然不知道过程名，但想往最新调用的 ILE 程序中的模块发送信息，可用*PGMNAME 与 ILE 程序名一起用做基本入口名字，做为最新调用程序中过程的名字。在例子中用的程序为：

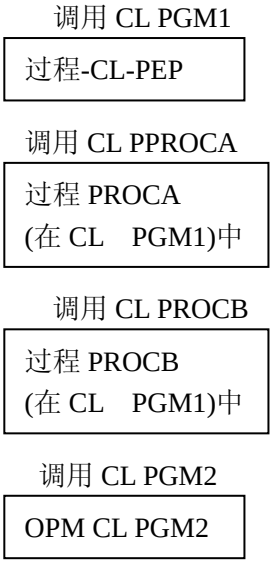
CL PGM1 是一个 ILE 程序，其中有过程 PROCA 和 PROCB。

CL PGM2 和 CL PGM3 都是 OPM 程序。

CL PGM3 给 CL PGM1 发信息但不知道哪个过程是最新调用的。

用*PGMNAME 和 CL PGM1 来完成发送。请看图 8-8。

如果要把某些 CL 程序而不是所有 CL 程序转换为 ILE 程序，*PGMNAME 是很有用的。例如，CL PGM1 是 OPM 程序，CL PGM3 给 CL PGM1 送信息且规定了 TOPGMQ(*SAME CL PGM1)。如果 CL PGM1 转换为 ILE 程序，仅在 CL PGM3 中的 SNDPGMMSG 起作用。由于在CL PGM1的调用堆栈中没有入口，所以CL PGM1不工作，如果规定TOPGM1(*SAME *PGMNAME *NONE CL PGM1)，CL PGM3 会成功的往 CL PGM1 发送信息，而不管你是否使用过程名，*PGMNAME 也能与 OPM 程序名一起使用，作用与仅用名字相同。例如，TOPGMQ(*SAME *PGMNAME *NONE OPM 程序)和 TOPGMQ(SAME OPM 程序)往同一地方发送信息，在不能确定信息是发送给 OPM 或 ILE 程序名时可考虑使用*PGMNAME。



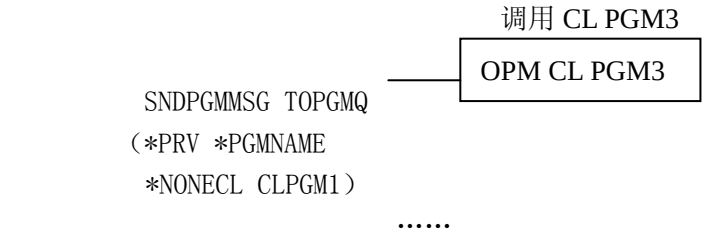
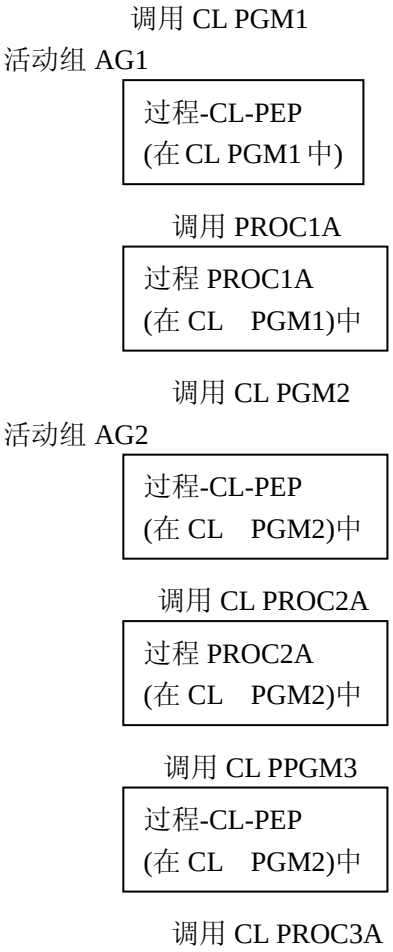


图 8-8　运行时调用堆栈的例子

用控制边界做基本入口：

可用特殊值*CTLBDY 标识基本入口为最近的一个控制边界。如果两个入口在两个不同的活动组中运行，那么在两个调用堆栈之间存在一个控制边界，用*CTLBDY 的一个标志是在同活动组运行做为发送信息的入口。

看图 8-9。例中 CL PGM1、CL PGM2、CL PGM3 是三个 ILE 程序，CL PGM1 在活动组 AG1 中运行，CL PGM2 和 CL PGM3 在活动组 AG2 中运行，PROC3A 往紧换 AG2 边界前的入口发送信息。



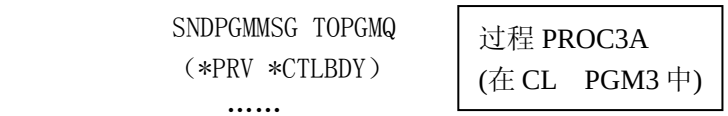


图 8-9 使用*CTLBDY 的例子

服务程序的考虑：

上面介绍的内容对 ILE 程序和 ILE 服务程序都适用，它们之间的最重要的区别是有关的信息管理，服务程序没有 PEP。

用来标识基本入口的任何选项，都不需要 PEP，一个例外是是在明显的使用-CL-PEP 情况。例如，TOPGMQ(*PRV *PGMBDY)总是把信息发送给 ILE 程序或服务程序的调用者。如果是 ILE 程序，PEP 由 PGMBDY 值标识做基本入口，如果是 ILE 服务程序，在服务程序中第一个被调用的入口是由*PGMBDY 值指定的。

8.2.4 在 CL 过程或程序中接收信息

用 RCVMSG 命令从过程或程序的信息队列接收信息，可用下列方式接收信息：

由信息类型：可以规定要接收的所有类型或某些类型（MSGTYPE 参数），新信息用 FIFO 顺序接收，但逃逸信息用 LIFO 顺序接收。

由信息引用键：可做下列之一：

— 用它的信息引用键接收，系统会给在队列中的每个信息分配一个引用键，且把它做为一个变量数据传送。因此必须在过程或程序说明这个变量（DCL 命令），必须在 RCVMSG 命令中规定 CL 变量，通过它传送键（MSGKEY 参数）

— 接收队列中指定信息引用键后的下一个信息，除了规定 MSGKEY 外，还必须规定 MSGTYPE(*NEXT)

— 接收队列中指定信息键前的信息，除了规定 MSGKEY 外，还必须规定 MSGTYPE(*PRV)

由它在信息队列中的位置：对第一个信息规定 MSGTYPE(*FIRST)
对最后一个信息规定 MSGTYPE(*LAST)

由信息类型和引用键一起（MSGTYPE 和 MSGKEY）

要接收一个信息，可以规定：

信息队列：接收用的信息队列

信息类型：可规定某个信息类型或所有类型

是否等待一个信息的到达：等待超时而没有信息，需要返回的 CL 变量要填空格（如果是数字则为零）且控制返给运行 RCVMSG 的过程或程序。

在接收后是否从队列中取消信息：如果不取消，则它成为队列中的老信息，仅能用信息引用键再被接收，但如果用 CHGMSGQ 命令把它变为新信息，就不用信息引用键接收它，但已经回答的查询信息不能置为新信息（详细信息请看 8.2.6）

要转换的 CCSID：规定信息正文返回的 CCSID。

一组 CL 变量可以在其中放下列信息（每一个相应于一个变量）：

— 信息在队列中的引用键（4 个字符长的变量）
— 信息（可变长的字符变量）
— 信息长度，包括替换变量数据的长度（十进变量，5 位小数）
— 联机帮助信息（可变为的字符变量）
— 联机帮助信息的长度，包括替换变量数据的长度（十进变量，5 位小数）
— 信息发送者提供的替换变量的信息数据（可变长字符变量）
— 信息数据的长度（十进变量，5 位小数）
— 信息标识（7 位字符变量）
— 严重码（十进变量，2 位长）
— 信息的发送者（最小为 80 个字符的变量）
— 接收信息的类型（2 位长的字符变量）
— 接收信息的报警选项（9 位长的字符变量）
— 包括预先定义信息的信息文件（10 位长的字符变量）
— 用来接收信息的信息文件所在的库名（10 位长的字符变量）
— 用来发送信息的信息文件所在的库名（10 位长的字符变量）
— 与取代返回数据相关的信息数据的 CCSID（5 位长十进变量）
— 由信息和信息帮助参数返回的与正文有关的正文 CCSID（5 位长十进变量）

```
RCVMSG MSGQ(QGPL/INVN) MSGTYPE(*ANY) MSG(&MSG)
```

接收的信息放在变量&MSG 中，&ANY 是 MSGTYPE 参数的缺省值。

在处理非 CL 语言写的 ILE 过程的调用堆栈入口信息队列时，可能接收到一个没有被处理的例处信息（逃逸或通知信息）。RCVMSG 可用来接收一个信息并告诉系统已经处理了例外情况。这种情况也可用 RMV 键字控制，如果对此键字规定*NO，则处理例外且信息做为老信息放在信息队列中，如果规定*KEEPEXCP，则不处理例外且信息做为新信息留在信息队列中，如规定*YES，则处理例外且信息从信息队列中取消。

也可用 RTNTYPE 键字来确定接收的信息是否为例外信息。如果是，怎么样处理它。

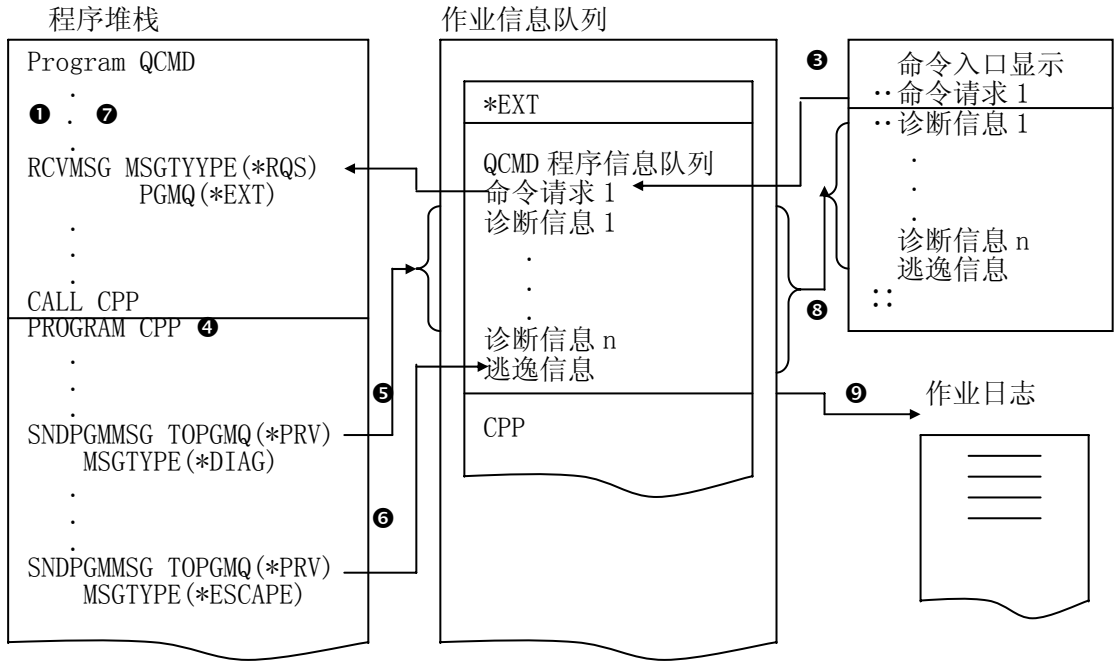
8.2.4.1 请求信息

接收到的请求信息是 CL 过程或程序处理 CL 命令的一种方法。例如，过程或程序从显示工作站得到输入，然后处理程序分析和操作的结果信息。通常，是从作业的外部信息队列（*EXT）接收请求信息的，接收到的请求是从输入流中读到的，对交互作业，接收到的请求是显示工作站用户在命令入口显示中输入的内容。例如，CL 命令是由 IBM 支持的 CL 处理接收到的请求。

你的过程或程序必须在请求信息中定义数据的语法。请求的中断及错误诊断，在分析请求或运行请求时，可检查任何错误，做为结果，会有信息送往过程或程序的调用信息队列中。过程或程序处理这些信息然后接收下一个请求信息，这样，就定义了一个请求处理的循环：接收请求信息，分析请求及运行，显示结果信息，然后接收下一个请求。如果在批作业中没有多个要接收的请求信息，则有逃逸信息送给过程或程序来指出这点。

一个作业的多个 OPM 程序或 ILE 过程能接收请求信息来处理。由较近程序调用接收的请求要嵌套在更高层程序调用之中，每个嵌套层的请求处理循环都互相独立。在 ILE 程序中，多个过程都能接收请求信息，如果多个过程处理请求而不是在同一 ILE 程序中发生嵌套，则嵌套层次保持独立。

下图解释由 QCMD 怎样处理请求信息。
（PIC10）



解释如下：

1、CL 处理程序 QCMD 从*EXT 接收一个请求信息。
2、如果在*EXT 中没有信息，显示命令入口显示，用户在显示中进入命令，它做为一个请求信息放在*EXT 中。
3、命令送到 QCMD 调用信息队列的末尾，且从这儿传送给 QCMD。
4、分析命令，调用命令处理程序（CPP）。
5、CPP 发送诊断信息给 QCMD 的调用信息队列。
6、CPP 发送逃逸信息给 QCMD 的调用信息队列，此信息通知 QCMD 诊断信息在队列中，QCMD 将结束处理 CPP。
7、QCMD 将监控请求检测（CPF9901）或功能检测（CPF9999）信息，然后 QCMD 将接收下一个请求信息。如果接收到 CPF9901 或 CPF999，则运行 RCLRSC 命令，它也监控 CPF1907（结束请求）和 CPF2415（用户在命令入口显示按 F3 或 F12 键）。
8、由于处理了请求信息，QCMD 的调用信息队列的所有信息都写在命令入口显示上，提示用户输入另外的命令。
9、前面的请求信息和相关信息都写到作业日志中，详细情况请看 8.7。

8.2.4.2　写请求处理的过程或程序

在一个程序中写一个 CL 过程管理请求处理有很多优点。它们是：

处理请求信息，如 8.2.4.1 所述。

允许使用结束请求命令（ENDRQS），可以从系统请求菜单中使用，或做为断开作业功能的一部分。

允许过滤产生的信息。

要做为请求处理的过程或程序，必须包括 SNDPGMMSG 和 RCVMSG 命令。下面的命令能使一个过程或程序成为请求处理器。

```
SNDPGMMSG  MSG('Request Message') TOPGMQ(*EXT) MSGTYPE(*RQS)
RCVMSG     PGMQ(*EXT) MSGTYPE(*RQS) RMV(*NO)
```

从 PGMQ *EXT 中接收信息，在接收到请求信息时，它移到规定 RCVMSG 命令程序的调用信息队列中，这样，在信息移出时，必须使用正确的调用信息队列。

如果是由信息引用键（MRK）移动的请求信息，应该从 RCVMSG 命令的 KEYVAR 键字中得到标志，而不是从 SNDPGMMSG 中得到，(在信息到达移出后，信息引用键会改变)。如果信息请求是从调用信息队列移出的，必须在 RCVMSG 命令中规定 RMV（*NO），这是因为过程或程序不是一个请求处理器。

在接收到信息请求时，标识过程或程序是用请求菜单中的选项 2（结束请求）来结束。请求处理器的过程或程序要包括一个对 CPF1907 的监控，由于结束请求功能会对请求处理器发送这个信息，所以监控它是很必要的。

在过程结束（正常或非正常）或运行 RMVMSG 命令从请求处理器的调用信息队列取消所有信息请求之前，过程或程序都要保留请求处理器。下面的命令从信息队列移出所有的信息请求，也结束请求处理：

```
RMVMSG   CLEAR(*ALL)
```

调用 QCAPCMD API 和规定信息收回键来使 AS/400 命令分析器来处理来自 AS/400 命令的信息请求，在接收信息请求时会得到一个信息收回键，QCAPCMD 会更新作业日志中的信息请求且加上提供的新值，QCAPCMD 也隐藏一些参数值，例如口令等。在下面两个条件之一存在时，系统不修改作业日志中的信息请求：

用 QCMDEXC 或 QCAEXEC。

对 QCAPCMD 提供信息回收键动作失败。

8.2.4.3 确定是否有请求处理器

要确定一个作业是否有请求处理器，可显示作业调用堆栈，在 DSPJOB 中选 11 或用 WRKJOB 命令，或在 WRKACTJOB 中选 10。如果在显示作业堆栈的 request level 列有数字，与此数相关的过程或程序就是一个请求处理器。在下例中，QCMD 和 QTEVIREF 都是请求处理器。

```
                          Display Call Stack
                                                        System:   S0000000
Job:   WS31          User:   QSECOFR       Number:   000173

Type options, press Enter.
  5=Display details
```

```
      Request   Program or
Opt   Level     Procedure      Library        Statement      Instruction
                QCMD           QSYS                             01DC
          1     QCMD           QSYS                             016B
                QTECADTR       QSYS                             0001
          2     QTEVIREF       QSYS                             02BA




                                                                      Bottom

F3=Exit   F10=Update stack   F11=Display activation group   F12=Cancel
F17=Top   F18=Bottom
```

下面是请求处理过程的例子：

```
 PGM
      SNDPGMMSG  MSG('Request Message') TOPGMQ(*EXT) MSGTYPE(*RQS)
      RCVMSG     PGMQ(*EXT) MSGTYPE(*RQS) RMV(*NO)
        .
        .
        .
      CALL    PGM(PGMONE)
      MONMSG  MSGID(CPF1907)
        .
        .
        .
      RMVMSG  CLEAR(*ALL)
      CALL    PGM(PGMTWO)
        .
        .
        .
 ENDPGM
```

过程中的头两个命令使它成为请求处理器，在运行 RMVMSG 命令后才取消请求处理器，在调用程序 PGMONE 后放一个 MONMSG 命令，这是因为可能从 PGMONE 往请求处理器送一个结束请求。如果没有监控，那么对结束请求会产生一个功能检测。在调用 PGMTWO 后没有信息监控，这是因为 RMVMSG 结束了请求处理。如果在没有请求处理过程或没有调用程序时试图做结束请求，会有错误信息且不做结束操作。

注：在样板程序中，RCVMSG 命令使用必须的最少的参数成为一个请求处理器，你需

要说明只想接收一个信息请求而不想取消它，也要标识一个特别的调用队列，从它取出信息请求。如果必要，可以增加其它参数。

8.2.5 在 CL 过程中回收信息

可用 RTVMSG 命令从信息文件中回收信息正文放在一个变量中，RTVMSG 对预先定义的信息描述操作，可以规定信息标识和信息文件名，以及下列内容：

要转换的 CCSID：规定信息正文和数据返回用的 CCSID

信息数据字段：替换变量的信息数据

信息数据 CCSID：提供的信息数据要考虑的 CCSID

一组 CL 变量，可以放置下列信息：

- 信息（变长的字符变量）
- 信息长度，包括替换变量数据的长度（5 位小数的十进变量）
- 信息联机帮助（变长的字符变量）
- 信息联机帮助的长度，包括替换变量数据的长度(5 位小数的十进变量)
- 严重码（2 位小数的十进变量）
- 报警选项（9 位的字符变量）
- 服务活动日志中的问题记录（一位的字符变量）
- 信息数据 CCSID，它与返回数据位置有关（5 位小数的十进变量）
- 正文数据 CCSID，定与由信息和信息帮助参数返回的正文有关

例如，下列命令把信息 USR1001 的信息描述加到信息文件 USRMSG 中：

```
ADDMSGD   MSGID(USR1001) MSGF(QGPL/USRMSG) +
          MSG('File &1 not found in library &2') +
          SECLVL('Change file name or library name') +
          SEV(40) FMT((*CHAR 10) (*CHAR 10))
```

下列命令把回收信息 USR1001 的文件名 INVENT 放在 10 个字符的变量&FILE 中，把库名放在 10 个字符的变量&LIB 中：

```
DCL &FILE TYPE(*CHAR) LEN(10) VALUE(INVENT)
DCL &LIB TYPE(*CHAR) LEN(10) VALUE(QGPL)
DCL &A TYPE(*CHAR) LEN(20)
DCL &MSG TYPE(*CHAR) LEN(50)
CHGVAR VAR(&A) VALUE(&FILE||&LIB)
RTVMSG   MSGID(USR1001) MSGF(QGPL/USRMSG) +
         MSGDTA(&A) MSG(&MSG)
```

&1 和&2 的数据放在过程变量&A 中，在&FILE 和&LIB 中的值已连接起来。在&MSG 中放下列信息：

```
File INVENT not found in library QGPL
```

如果在 RTVMSG 中没用 MSGDCA 参数，则&MSG 中的信息为：

```
File not found in library
```

在把信息放到&MSG 中后，可以做以下事情：

用 SNDPGMMSG 发送信息

用变量做 DDS 中的信息行正文（38 列为 M）

使用信息子文件

打印或显示信息

注：不能用包括在正文中的变量名来回收信息正文 。RTVMSGD 用来返回能够发送的信息，要访问用 ADDMSGD 进入的信息，看 QUSRTOOL 中的 CVTMSGF 工具。

8.2.6 从信息队列取消信息

可用下列命令及参数把信息从信息队列取消：

RMVMSG，CLRMSGQ，RCVMSG 中的 RMV 参数，SNDRPY 中的 RMV 参数。

也可在显示信息屏中用取消功能键，或在处理信息队列显示中用清理信息队列选项，可以取消的信息是：

一个信息

所有信息

没回答的例外信息

所有旧信息

所有新信息

从所有非活动程序中来的所有信息

在用 RMVMSG 取消一个信息或用 RCVMSG 取消一个旧信息时，要规定要取消信息的信息引用键。

注：信息引用键也可用来接收信息或回答信息。

如果取消的是没有回答的查询信息，要给信息发送者送出缺省回答，然后取消它们。如果取消的是已回答的查询信息，则查询信息和回答信息都取消。

如果从用户的作业信息队列中取消所有非活动过程或程序的所有信息，要在 RMVMSG 命令中规定 PGMQ(*ALLINACT)和 CLEAR(*ALL)，如果要在取消所有非活动信息前打印作业日志，要用 DSPJBOLOG 命令，规定 QUTPUT(*PRINT)。

在处理 ILE 过程的所有调用信息队列时，在运行 RMVMSG 命令时队列中可能有没处理的例外信息，可用命令中的 RMVEXCP 键字来控制对这类信息采取动作。如果规定 RMVEXCP(*YES)，则 RMVMSG 处理例外情况且取消信息。如果规定*NO，就不取消信息，结果是不处理例外。下列命令从用户信息队列 JONES 中取消一个信息，信息引用键是在变量 MRKEY 中：

```
DCL &MRKEY TYPE(*CHAR) LEN(4)
RCVMSG MSGQ(JONES) RMV(*NO) KEYVAR(&MRKEY)
RMVMSG MSGQ(JONES) MSGKEY(&MRKEY)
```

下列命令从信息队列取消所有信息：

```
RMVMSG  CLEAR(*ALL)
```

注：在用户信息队列或工作站信息队列中每种类型发送的信息的最大数为 65535，即

在队列中可有 65535 个诊断信息，65535 个完成信息等等，对所有调用信息队列或*EXT，每类的最大数没有限制。

8.3　在 CL 过程或程序中监控信息

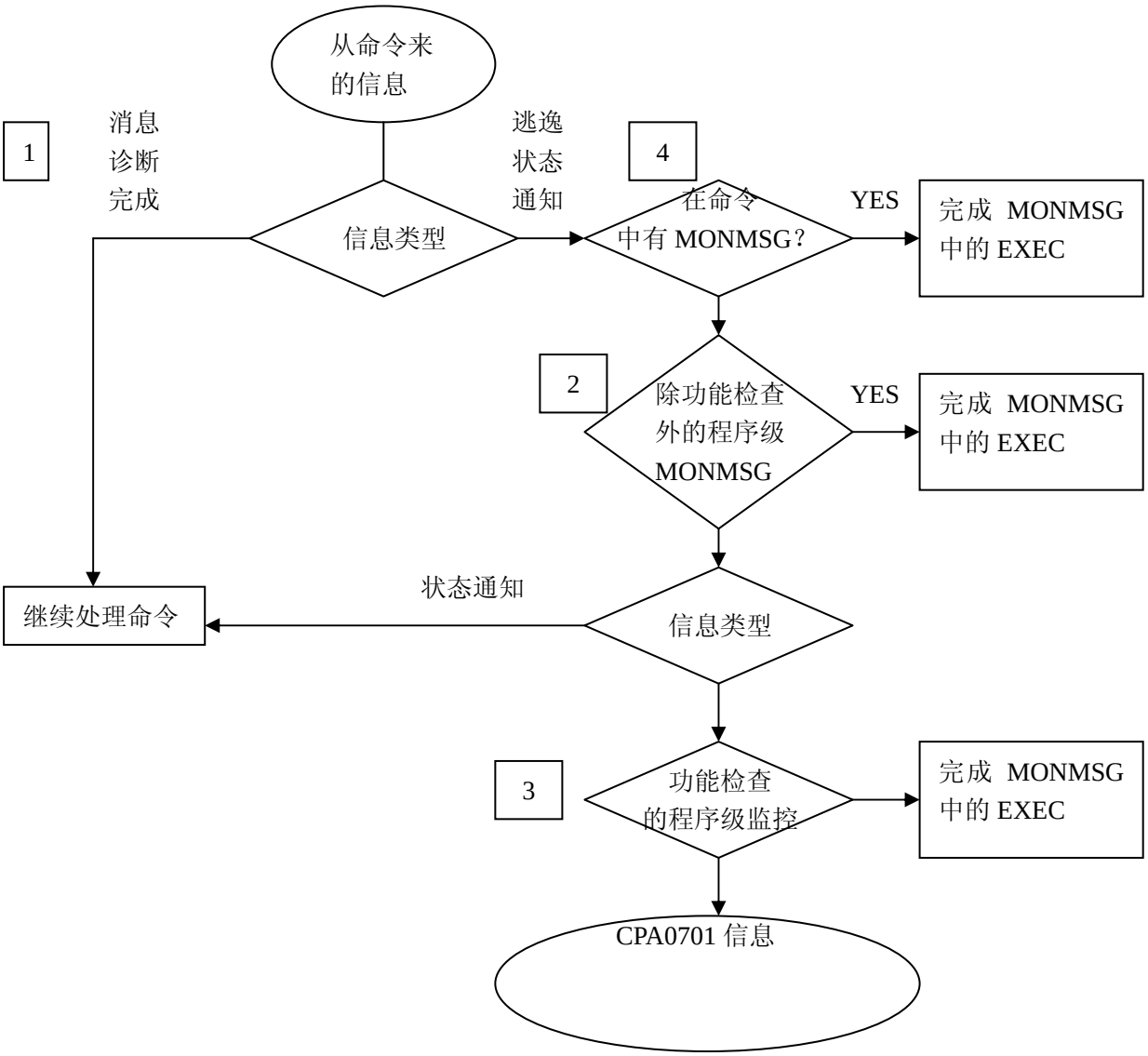
可在过程或程序中用命令，或在另外的过程或程序中的命令来监控送到过程或程序中的逃逸、通知和状态信息。MONMSG 命令监控在命令中规定条件的送往调用信息队列中的信息，如果条件存在，则运行 MONMSG 中规定的命令。MONMSG 的功能如下：

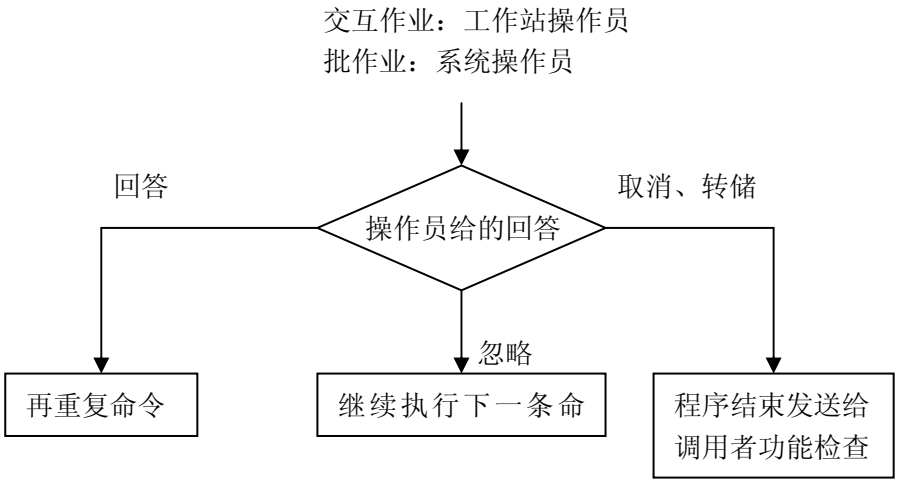
逃逸信息：它告诉过程或程序有错误条件发生，强行结束发送者。要监控这类信息，你要采取适当的动作或清理及结束过程或程序。

状态或通知信息：它告诉过程或程序是一个正常结束的条件，它不至于结束发送者。要监控这类信息，过程或程序可以检查这些条件，也能使功能不继续进行。

可用二级 MONMSG 命令监控信息：

过程级：在 CL 过程或程序的 DCL 命令后立即用 MONMSG 命令来监控逃逸、通知或状态信息，这叫做过程级的 MONMSG 命令，在一个过程或 OPM 程序中可以使用至多 100 个这样的 MONMSG 命令，这就让你对所有命令用同一种方法来处理相同的逃逸信息，EXEC 参数是可选的，且在 EXEC 中只能规定 GOTO 命令。





解释：

1.MONMSG、检查消息、诊断和完成信息。

2.CPF0000 的程序级 MONMSG 由逃逸、状态、通知信息激活。

3.CPF9999（功能检查）的程序级 MONMSG 由不是其它命令或程序级监控的逃逸信息激活。在 CPF9999 送往过程前，原逃逸信息过滤给同一 CL 程序中的活动过程。如原逃逸信息没被活动的过程处理，则发送 CPF9999。

4.如果状态信息送往*EXT，就不能监控它。

命令级：在 CL 过程或程序中的命令后立即用 MONMSG 命令来监控逃逸、通知或状态信息，这叫命令级的 MONMSG 命令，可对一个命令使用多至 100 个命令级 MONMSG，这让你可用不同的方法处理不同的逃逸信息。

要监控这些信息，要在 MONMSG 命令中用如下格式之一规定信息的一般标识：

pppmmnn：监控某个信息，例如 MCH1211 是零做除数的逃逸信息。

pppmm00：监控用某个特许程序码（ppp）开头的由 mm 规定的数字信息标识。例如，CPF5100 指出所有以 CPF51 开头的通知、状态和逃逸信息。

ppp0000：监控信息标识以 ppp 开头的信息。例如 CPF0000 指出要监控以 CPF 开头的所有通知、状态和逃逸信息。

注：在做系统功能时不要用 MONMSG CPF0000，安装、备份、恢复系统时也不要用它，因为可能会去掉一些重要信息。

CPF9999：监控所有信息标识的功能检测信息，如果对一个错误信息没有监控，则它成为 CPF9999（功能检测）。

注：一般来说，在送出通知和诊断信息时，控制也转给监控程序。

除了用信息标识监控逃逸信息外，也能用 MONMSG 中规定的字符串与信息中的数据进行比较。下面的命令监控 MYFILE 的逃逸信息 CPF5101，文件名做为信息数据发送。

```
MONMSG MSGID(CPF5101) CMPDTA(MYFILE) EXEC(GOTO EOJ)
```

比较数据可以是 28 个字符长，比较从信息数据的第一个字段的第一个字符开始，如果它们匹配，则执行 EXEC 参数中规定的动作。

在 EXEC 参数中规定如何处理逃逸信息，除 PGM、ENDPGM、IF、ELSE，DCL，DCLF，

ENDDO 和 MONMSG 外，其余命令都能在 EXEC 中规定。在 EXEC 中可以规定 DO 命令，此时运行在 DO 组内的命令。在运行时，控制返回给发送逃逸之后的命令，但如果规定 GOTO 或 RETURN 命令，则控制不返回，如果没规定 EXEC，则忽略逃逸，过程继续执行。下面是 CHGVAR 命令的例子，它监控要做除数的逃逸信息，信息标识为 MCH1211：

```
CHGVAR VAR(&A) VALUE(&A / &B)

MONMSG MSGID(MCH1211) EXEC(CHGVAR VAR(&A) VALUE(1))
```

变量&A 的值修改为&A 除以&B 的值。如果&B=0，不能做除法，且送给过程一个逃逸信息，这时，&A 的值变为 1，（在 EXEC 参数中规定）。你也可以测试&B 的值，仅在它不是零时才做除法。在下例中，程序监控逃逸信息 CPF9801（目标没找到），它由 CHKOBJ 命令产生：

```
            PGM
            CHKOBJ LIB1/PGMA *PGM
            MONMSG MSGID(CPF9801) EXEC(GOTO NOTFOUND)
            CALL LIB1/PGMA
            RETURN
NOTFOUND:   CALL FIX001 /* PGMA Not Found Routine */
            ENDPGM
```

下列 CL 过程包括二个 CALL 命令和一个过程级的对 CPF0001 的监控命令。（如果不能成功地完成 CALL 命令，则产生逃逸信息）。如果有一个 CALL 失败，过程送出完成信息且结束。

```
          PGM
          MONMSG MSGID(CPF0001) EXEC(GOTO ERROR)
          CALL PROGA
          CALL PROGB
          RETURN
ERROR:    SNDPGMMSG MSG('A CALL command failed') MSGTYPE(*COMP)
          ENDPGM
```

如果在 MONMSG 命令中没有规定 EXEC，则忽略任何逃逸信息，如果在除 IF 以外的命令上发生逃逸信息，过程或程序继续处理下一个不产生逃逸信息的命令，如果在 IF 命令中发生逃逸信息，过程或程序继续处理 IF 命令中为假的条件。下例解释不同类型发生逃逸信息时所做的处理：

```
PGM
DCL &A TYPE(*DEC) LEN(5 0)
DCL &B TYPE(*DEC) LEN(5 0)
MONMSG MSGID(CPF0001 MCH1211)
CALL PGMA PARM(&A &B)
```

```
IF (&A/&B *EQ 5) THEN(CALL PGMB)
ELSE CALL PGMC
CALL PGMD
ENDPGM
```

根据发生逃逸信息的地方不同，能发生下列情况：

如果在调用 PGMA 时产生 CPF0001，过程重新处理 IF 命令

如果在 IF 命令中发生 MCH1211，IF 条件失败，过程处理调用 PGMC

如果在调用 PGMB 和 PGMC 时发生 CPF0001，过程调用 PGMD

如果在调用 PGMD 时，发生 CPF0001，过程执行 ENDPGM，它返回到调用过程

也可由过程或程序中的命令或另外的命令来监控发送的相同逃逸信息，这就需要三个 MONMSG 命令，一个命令写在需要处理逃逸信息的命令下面，它用在发生逃逸信息的时候，另一个 MONMSG 跟在最后一个 DCL 命令后，它能够监控所有其它的命令。

仅在过程或程序需要监控信息时才用 MONMSG 命令，一个过程中的 MONMSG 不能被同一程序中的另外过程使用。程序设计参考一书列出 CL 命令发出的逃逸、通知、状态信息，你也可以列出自己定义的所有信息的表。

8.3.1 缺省处理

多数逃逸信息会送往调用命令、程序或过程的过程中，你可能不想监控所有的信息，只想监控和处理对过程功能有影响的那些信息，系统对你没有监控的任何信息提供了缺省监控及处理。

缺省处理假定在过程中检测到了错误。如果用调试过程，会有信息送到工作站，然后可以改它，如果没有用调试，系统会做一个信息过滤功能，此功能分为两步：

移动调用堆栈中逃逸信息开始的一步

检查看有没有 MONMSG 监控逃逸

如果有 MONMSG，则停止信息过滤，采取命令中规定的动作，继续进行信息过滤直到遇到另一个 MONMSG 或遇到一个嵌套组的边界，即逃逸信息不能跨越活动组边界进行过滤。

如果在有 MONMSG 命令且有信息提供的过程前遇到活动级边界，则开始功能检测处理，原逃逸例外的动作被认为是完成了，然后发出 CPF9999 送给原逃逸的目的过程。如果过程有 MONMSG 做功能检测，则采取命令规定的动作，如果没有 MONMSG，则把查询信息送往交互作业的工作站操作员，操作员能用下列之一回答：

R	重试失败的命令
I	忽略信息，继续处理下一条命令
C	取消过程，对堆栈中下一个过程做过滤功能检测
D	对失败过程转储者调用堆栈入口，取消过程，对堆栈的下一个过程过滤功能检测。如果给出回答或作业是批作业，这是缺省的动作。

功能检测不能跨越活动组边界。如果所做回答引起功能检测跨越一个活动组边界，那么会停止做进一步的功能检测，所有到达活动组边界的过程都被取消，逃逸信息 CEE9901 送往前一个堆栈入口。

你能监控功能检测的逃逸信息，因此可以做：

清理及结束过程

继续处理过程的其它问题

注：如果没监控的逃逸的信息描述规定了一个缺省动作，在发送功能检测信息之前调用

缺省处理程序，当缺省处理程序返回时，开始功能检测处理。

8.3.2 通知信息

除了监控逃逸信息，可以监控送到过程或程序的调用信息队列中的通知信息。通知信息告诉你的过程或程序非典型错误的条件。通过监控它，可以规定不同于没有产生信息条件的动作。很少 IBM 支持的命令提供通知信息，它的监控与对逃逸信息的监控类似，不同的是如果没有监控发生的情况，通知信息也在活动组的边界中从过程或程序中过滤。如果已到达组边界区没有遇到 MONMSG 命令，则自动返回一个缺省回答且允许发送者继续处理。它不象逃逸信息，没监控的通知信息不考虑为过程或程序的错误。

8.3.3 状态信息

可以监控过程或程序中的命令或调用的过程或程序送出的状态信息，它告诉过程完成工作的状态，由监控状态，可以防止发送的过程或程序做更多处理，对状态信息没有信息存在信息队列中，即状态信息是不能接收的。

如果没有监控状态，它也象逃逸信息和通知信息一样会被过滤。如果到达了活动组边界还没有发现 MONMSG 命令，就认为它是完成信息，控制会返回给信息的发送者来继续处理。经常会发送状态信息做通讯正常的条件，操作继续进行。送往外部信息队列的状态在交互显示中出现，通知用户一个功能正常执行。例如 CPYF 命令会给用户发送信息告诉它复制工作正在进行中。

仅预先定义的信息能发送状态信息，立即信息不能发送，如果没有自己的信息说明，可以使用系统支持的信息标识 CPF9898 来发送状态信息。

在功能完成后，过程或程序会从交互显示中取消状态，它不能用命令取消，但可把空的信息送到*EXT 中来取消此信息，这可用 CPI9801 来做这件事。当控制返回到 AS/400 程序时，*STATUS 信息可从第 24 上行取消而不必发送 CPI9801。

下面例子给出 CPF9898 和 CPI9801 的典型使用方法：

```
SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) +
          MSGDTA('Function xxx being performed') +
          TOPGMQ(*EXT) MSGTYPE(*STATUS)

  _
  _   /*  Your processing function  */
  _
SNDPGMMSG MSGID(CPI9801) MSGF(QCPFMSG) +
          TOPGMQ(*EXT) MSGTYPE(*STATUS)
```

8.3.4 防止显示状态

你不能防止发送状态信息的命令，但可以不让状态在屏幕的底部显示，做这件事可有两种方法：

用 CHGUSRPRF 命令：可以修改你的用户配置文件，在用它注册时，可不显示状态，做法是规定 USROPT(*NOSTSMSG)。

用 CHGJOB 命令：可修改正运行的作业，让它不显示状态，做法是规定 STSMSG（*NONE），可在 CHGJOB 中规定 STSMSG(*NORMAL)来看状态。

第三个方法，但不建议用它，是用 OVRMSGF 把状态标识改为一个空白信息。

8.4 中断处理程序

中断处理程序是一个有信息到达方式为*BREAK 的信息队列时自动调用的一个程序。在同一个CHGMSGQ命令必须规定程序名和中断交付名。虽然在CHGMSGQ中规定了程序，但处理信息的程序中可以有多个过程，过程必须运行 RCVMSG 命令接收信息，要接收和处理这些信息，调用的用户定义的程序要传递参数，(程序中的第一个过程传递这些参数)。信息队列的参数标识和信息引用键引起中断。有关这些参数，可以看系统 API 参考一书。如果调用中断处理程序，它中断产生信息的作业，在中断处理程序结束时，原程序重新运行。

下面的程序是中断处理程序，它仅由一个过程组成：

```
PGM PARM(&MSGQ &MSGLIB &MRK)
DCL VAR(&MSGQ) TYPE(*CHAR) LEN(10)
DCL VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
DCL VAR(&MRK) TYPE(*CHAR) LEN(4)
DCL VAR(&MSG) TYPE(*CHAR) LEN(75)
RCVMSG  MSGQ(&MSGLIB/&MSGQ) MSGKEY(&MRK) +
        MSG(&MSG)
.
.
.
ENDPGM
```

在生成中断处理程序后，运行下面的命令让它与 QSYSOR 信息队列相连：

```
CHGMSGQ    MSGQ(QSYS/QSYSMSG)  DLVAR(*BREAK)  PGM(PGMA)
```

注：1、在处理完信息后，它会从信息队列中取消，如信息队列在中断方式，在队列中的任何信息都会调用中断处理程序。

2、接收信息的过程或程序中不必写等待时间不为零的程序段来接收信息，可在 RCVMSG 中对等待参数规定不为零。当作业在中断处理事件中运行时不能由系统处理信息到达。

中断处理程序的一个例子是让程序发送一个信息，它正常的是送往 QSYSOPR 队列或其它队列。

下面是用户定义的处理中断信息的例子（也仅一个过程）。用户不用响应 CPA5243（在设备&1 上接 Rendy，start 或 start-stop）和 CPA5316（校准设备&3）。

```
BRKPGM:     PGM (&MSGQ &MSGQLIB &MSGMRK)
            DCL &MSGQ TYPE(*CHAR) LEN(10)
            DCL &MSGQLIB TYPE(*CHAR) LEN(10)
            DCL &MSGMRK TYPE(*CHAR) LEN(4)
            DCL &MSGID TYPE(*CHAR) LEN(7)
            RCVMSG MSGQ(&MSGQLIB/&MSGQ) MSGKEY(&MSGMRK) +
                   MSGID(&MSGID) RMV(*NO)
            /* 忽略信息 CPA5243 */
```

```
            IF (&MSGID *EQ 'CPA5243') GOTO ENDBRKPGM
            /* 回答校准格式信息 */
            IF (&MSGID *EQ 'CPA5316') +
                  DO
                  SNDRPY MSGKEY(&MSGMRK) MSGQ(&MSGQLIB/&MSGQ) RPY(I)
                  ENDDO
            /* 要用户干预的其他信息 */
            ELSE CMD(DSPMSG MSGQ(&MSGQLIB/&MSGQ))
ENDBRKPGM:  ENDPGM
```

在上面的程序中，在运行 DSPMSG 命令时，如果 CPA5316 到达队列中，DSPMSG 显示会给出产生中断的原始信息和 CPA5316 信息。DSPMSG 将等待操作员回答 CPA5316，然后再进行处理。

注：如果中断程序要等待从显示中输入数据，此程序不能打开显示文件。

可用系统回答列表来指出系统将给预先定义的信息给出回答。这样，显示工作站用户就不需要回答了，详细内容请看 8.6。

用户写的中断处理程序可能需要一个暂停或重存过程，以保证在执行信息处理功能时暂停或重存显示，仅在下列条件下，暂停和重存过程是必须的：

中断程序的过程显示其它的菜单或屏幕。

中断程序调用显示其它菜单或屏幕的程序。

下面的例子解释用户过程和需要暂停或重存显示的显示文件。

注：在生成显示文件时要规定 RSTDSP(*YES)。

```
A           R SAVFMT                    OVERLAY  KEEP
A*
A           R DUMMY                     OVERLAY
A                                       KEEP
A                                       ASSUME
A             DUMMYR          1A     1  2DSPATR(ND)
```

```
PGM PARM(&MSGQ &MSGLIB &MRK)
    DCL VAR(&MSGQ) TYPE(*CHAR) LEN(10)
    DCL VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
    DCL VAR(&MRK) TYPE(*DEC) LEN(4)
    DCLF FILE(UDDS/BRKPGMFM)
    SNDF RCDFMT(SAVFMT)
    CALL PGM(User's Break Program)
    SNDF RCDFMT(SAVFMT)
ENDPGM
```

如果不想让用户规定中断处理程序来中断交互作业，程序可提交给批处理运行。可以规定一个中断处理程序接收信息然后再用 SBMJOB。SBMJOB 用需要的参数完成当前的中断处理程序，控制然后返给交 互作业，它继续正常执行。

中断处理程序的另外一个例子是在 QUSRTOOL 中的 STSMSG 工具中给出，它允许你修改某些或全部在工作站接收的信息成为状态信息。

8.5 QSYSMSG 信息队列

QSYSMSG 是一个可在 QSYS 库中生成的可选信息队列。如果它存在且没有损坏，某些信息会直接送给它，而不送往 QSYSOPR 队列。这就允许用户写的程序在发送某些信息时获得控制权。除非要让 QSYSMSG 接收特别的信息，你不要生成这个队列。

用下面的命令生成 QSYSMSG：

```
CRTMSGQ  QSYS/QSYSMSG +
         TEXT('Optional MSGQ to receive specific system messages')
```

它一旦生成，所有规定的信息（请看 8.5.1）都直接送到这里。你可写一个程序来接收这些信息，用它们来完成特别动作，也可往 QSYSOPR 发送其它信息。这个程序可以写成中断处理程序。

8.5.1 送往 QSYSMSG 信息队列的信息

下面说明送到 QSYSMSG 信息队列中的信息：

CPF0907　存在严重的存储方面问题，请按帮助键。

如果在系统 ASP 中可用的辅存总量达到限值，会发送此信息，可用系统服务工具（SST）功能来显示或修改限值。详细信息请看备份和恢复一书。

CPF111C　系统调度主机下电

系统检查到 OS/400 特许程序的损坏，如果在两小时内不做重存相应的产品特许信息，则系统自动下电。

用下列命令重存产品特许信息：

用 WRKOBJ 命令删除产品信息：

```
WRKOBJ      QSYS/QSZ0050    *PRDDFN
```

用 RSTOBJ 命令重存产品信息：

```
RSTOBJ OBJ(QSZ0050)     DEV(设备名)
       SAVLIB(QSYS)     OPTION(*NEW)
```

CPF111D　系统正在下电

系统检查到 OS/400 特许信息损坏，系统正在下电，系统将完成下电。

用下面的命令来避免系统再次下电：

```
WRKOBJ      QSYS/QSZ0050    *PRDDFN
RSTOBJ      OBJ(QSZ0050)    DEV(设备名)
            SAVLIB(QSYS)    OPTION(*NEW)
```

CPF1269　在通讯设备上接收的程序启动请求，由于某种原因被拒绝

在启动请求被拒绝且包括一个为什么发生拒绝的原因码时，发送此信息。对每个原因码的详细解释，请看 ICF 程序设计一书中 CPF1269 的内容。

如果在用 APPC 时口令不对或没有权限，它意味着作业在错误状态或某人试图中断保护，可以选择避免进一步使用 APPC 设备描述，直到做下面工作后认可这些条件：

往 QSYSOPR 队列送信息

记录下这些内容给安全管理人员看

执行 ENDMOD 命令把允许的作业设为零，它允许当前使用同等设备描述的作业保持活动，而避免其它作业在认可条件前启动。

计算在一个给定时间周期的试图要做的数目，可以根据它在程序中建立一个限值，表示在采取某些行动之前这个试图数目是无效的，可用工作标识单元（它可以是空格）；用 APPC 设备描述或整体的 APPC 环境来分配这个限值。

CPF1393 由于多次不正确的注册请求，使用户配置文件不可用

在用户试图注册多次，引起用户配置文件不可用时发送此信息。

CPF1397 子系统关闭工作站

如果达到分配给 QMAXSIGN 的限值且设备不可用，则发送 CL 信息，它指出用户没有输入一个有效的口令。

它的信息数据包括发送信息的设备名，可用这个信息或设计一个程序来采取适当动作，可以考虑做下列之一：

给 QSYSOPR 发送同样信息

记录下来给安全管理人员看

在延迟一段时间后，设备自动变为可用

CPI0948 在&1 磁盘上的镜像保护暂停

系统不能分配存储设备，数据没丢失

CPI0949 在&1 磁盘上的镜像保护暂停

磁盘镜象保护挂起

CPI0950 存储单元现在可用

从配置中丢失的存储设备，现在可用，数据没丢失

CPI0953 到达 ASP 存储阈限

在规定的 ASP 中可用存储达到限值会发送此信息，它的信息数据中包括辅存特性、使用的辅存限值的百分比及可用辅存的百分比，可用这些信息决定采取的动作。

CPI0954 超出 ASP 存储限值

在规定的 ASP 中所有可用的存储都用完了发送此信息。

CPI0955 超出系统 ASP 非保护存储限值

在系统 ASP 中所有可用的存储都用完了发送此信息。

CPI0964 存在弱电池问题

如果外部 UPS 或内部电池指出一个弱电池条件，发送此信息。

CPI0965 系统单元上的电池电源功能故障

如果电池有故障或系统部件的电池电源部件的充电有故障则发送此信息。

CPI0966 扩展单元上的电池电源功能故障

如果电池有故障或扩充部件的电池电源部件的充电有故障则发送此信息。

CPI0988 恢复磁盘&1 上的镜像保护

磁盘设备的镜象同步已经开始且重设磁盘镜象保护则发送此信息。在重设磁盘镜象保护之前，系统完成的步骤之一是把数据从一个磁盘复制到另外一个磁盘上去，这样两个磁盘上的数据是相同的。在复制时系统性能会很慢，在复制完数据后，会给此信息队列发送 CPI0989 信息，且重

开始磁盘镜象保护。

CPI0989 恢复磁盘&1 上的镜像保护

如果磁盘的镜象同步成功地完成了，会发送此信息，系统完成了数据复制，重开始磁盘镜象保护。

CPI0998 在磁盘单元&1 上发生错误

在磁盘设备&1 上发现错误会发送此信息，它不包括运行错误分析得到的故障信息。

CPI1117 在库&2 中的损坏的目标&1 已删除

库中的作业调度因为损坏了被删除掉会发送此信息。

CPI1136 镜像保护仍暂停

如果在一个或多个磁盘上的镜象保护每小时都挂起会发送此信息。

CPI1138 存储溢出恢复

在 ASP　&1 中不再有任何由于原因&2 溢出到系统 ASP 中的目标时发送此信息。

CPI1139 存储溢出恢复失败

在试图从存储溢出故障中恢复时会发送此信息。

CPI1153 系统口令旁路周期结束

在系统用口令旁路周期有效来操作时发送此信息。旁路周期结束，除非提供正确的系统口令，下一个 IPL 也不能成功地完成。

CPI1154 系统口令旁路周期在&5 天内结束

没输入系统口令或输入的不正确会发送此信息，此时会选择系统旁路周期。

CPI1159 系统 ID 因为多于&1 次安装而超期

在旁路 ID 接近超期时会发送此信息，请与 IBM 服务人员联系。

CPI1160 系统 ID 超期

在系统 ID 超期时会发送此信息，请与 IBM 服务人员联系。

CPI1161 &1 单元的设备奇偶保护可完全操作

&1 单元是有设备奇偶保护的磁盘设备子系统的一部分。&1 需要服务，数据已备份。如果不改正此条件，可能会发生机器检测，性能会下降，也可能丢失数据。

CPI1162 &1 单元的设备奇偶保护不可完全操作

&1 单元是有设备奇偶保护的磁盘设备子系统的一部分。由于下列原因，&1 不可完全操作：服务人员更换此单元；单元不可操作，也没有足够的信息运行问题分析。

CPI1165 一个或多个设备奇偶保护仍然不可完全操作

在此错误期间仍有一个或多个设备奇偶保护的磁盘单元子系统中的单元不可操作。

CPI1166 设备奇偶保护可完全操作

提供设备奇偶保护的所有 IOP 子系统单元全部可操作。

CPI1167 发生临时 I/O 处理错误

磁盘设备的 I/O 处理器发生错误条件。

CPI1168 磁盘单元&1 发生错误

号码为&1 的磁盘设备发现错误，有损坏的目标。如果问题严重，会发生机器检测，接着会标识这些磁盘单元。

CPI1169 磁盘单元&1 不可操作
磁盘单元&1 停止操作，无数据丢失。

CPI1393 子系统&1 使设备&3 上的用户&2 不可操作
用户注册的最大次数已达到 QMAXSIGN 规定的数会发送此信息。此时采取的动作用 QMAXSGNACN 决定。

CPI2209 用户配置文件&1 损坏，被删除
用户配置文件由于损坏被删除时会发送此信息，属于此用户配置文件自己的目标先已被删除，这些目标没有主人，可用 RCLSTG 命令把这些目标的主人改为 QDFTOWN。

CPI2283 QAUPCTL 系统值改为*NONE
由于审查失败而关闭审查后每小时可发送此信息。要打开审查或确定审查失败的原因，可把 QAUDCTL 改为除*NONE 以外的其它值。

CPI2284 QAUPCTL 系统值改为*NONE
在 IPL 期间由于审查失败而由系统关闭了审查时会发送此信息。要打开审查或确定审查失败的原因，可把 QAUDCTL 改为除*NONE 以外的其它值。

CPI8A13 QDOC 库接近保存历史限值
在库 QDOC 中的目标数接近系统支持的一个库中的最多目标数时会发送此信息。

CPI8A14 QDOC 库已超出保存历史限值
在库 QDOC 中的目标数超出系统支持的一个库中的最多目标数时会发送此信息。

CPI8898 在光盘总线上检查出光盘信号丢失
在检查到光盘总线故障时会发送此信息，总线在减少方式下运行，此信息记录在服务活动日志中且可用 PAR 选项。

CPI9014 从设备上接收的口令无效
从不正确的文本交换对话中接收到口令时会发送此信息，可能会指出无权限的访问系统。

CPI9490 在设备&25 上有磁盘错误
在检查到磁盘错误时会发送此信息。

CPI90A0 在设备&25 上有磁盘错误
在检查到磁盘错误时会发送此信息。

CPI94CE 在总线扩允适配器、系统处理器或电缆上检测到错误
系统在主存中检查到故障时会发送此信息，系统性能会降低，运行问题分析来确定有故障的卡。

CPI94CF 检测到主存卡故障
系统在主存中检查到故障时会发送此信息，系统性能会降低，运行问题分析来确定有故障的卡。

CPI94FC 在设备&25 上发生磁盘错误
在 9336 磁盘上检查到它的一个部件已超出错误限值，且开始有错误。

CPP0DD9 检测到系统处理器故障
在系统处理器或系统处理器缓冲器检查到故障时会发送此信息，系统性能会降低。

CPP0DDA 在槽 9 上检测到系统处理器故障

在系统处理器或系统处理器缓冲器检查到故障时会发送此信息，系统性能会降低。

CPP0DDB 在槽 10 上检测到系统处理器故障

在系统处理器或系统处理器缓冲器检查到故障时会发送此信息，系统性能会降低。

CPP0DDC 检测到系统处理器故障

在系统处理器上检查到故障会发送此信息，系统性能降低。

CPP0DDD 检测到系统处理器诊断码错

在 IPL 期间，由系统处理器诊断检查出故障，但系统仍可完成功能。

CPP0DDE 检测到系统处理器错误

在系统处理器上检查到故障，硬件 ECC 正处理故障，但如果做 IPL，不会初始控制，系统会做没有处理器的自身重配置。

CPP0DDF 丢失系统处理器

在多处理器系统中如果丢失一个处理器会发送此信息。

CPP29B0 在设备&25 上的恢复阈限超出

在 9337 磁盘单元中的一个部件开始故障时会发送此信息。

CPP29B8 在设备&25 上的设备保护暂停

在 9337 磁盘单元中的一个部件开始故障时会发送此信息，RAID 5 的技术设备保护在磁盘臂上挂起。

CPP29B9 在设备&25 上的电源保护暂停

在 9337 磁盘臂上的一个电源模块故障时会发送此信息。在此磁盘臂上电源保护被挂起。

CPP29BA 在设备&25 上的硬件故障

在 9337 磁盘臂上一个部件故障时会发送此信息，需要采取服务性动作。

CPP951B 电池电源设备故障

在电池电源故障时会发送此信息。

CPP9522 电池电源设备故障

在 5042 扩充单元或 5040 扩展单元的电池电源单元故障时会发送此信息。

CPP955E 电池电源设备没安装

在 9406 系统单元电源支持的电池电源单元没安装时会发送此信息。

CPP9575 在 9406 上的电池电源设备要更换

在 9406 系统单元的电池电源单元故障且要更换时发送此信息。它可能仍可工作，但可发生比充电循环的建议数多的情况。

CPP9576 在 9406 上的电池电源设备要更换

在 9406 系统单元的电池电源单元故障且要更换时发送此信息。它可能仍可工作，在比建议长的时间内必须安装。

CPP9589 电池电源设备检查完成

对电池电源单元的测试完成后发送此信息，结果要记录下来。

CPP9616 电池电源设备没安装

在 5042 扩充单元或 5040 扩展单元上没安装电池电源单元时会发送此信息。

CPP9617 电池电源设备需要更换

在 5042 扩充单元或 5040 扩展单元上没更换电池电源单元时会发送此信

息，它仍可工作，但可发生比充电循环的建议数多的情况。

CPP9718　电池电源设备需要更换
在 5042 扩充单元或 5040 扩展单元上没更换电池电源单元时会发送此信息，它仍可工作，但在比建议长的时间内必须安装。

CPP961F　大容量 DC-3 故障
在 9406 的 DC 大容量模块 3 故障时会发送此信息。

CPP9620　大容量 DC-2 故障
在 9406 的 DC 大容量模块 2 故障时会发送此信息。

CPP9621　大容量 DC-1 故障
在 9406 的 DC 大容量模块 1 故障时会发送此信息。

CPP9622　大容量 DC-1 故障
在 5042 扩充单元或 5040 扩展单元上的 DC 大容量模块 1 上有故障时会发送此信息，其它的 DC 大容量模块也能引起这种故障。

CPP9623　大容量 DC-2 故障
在 5042 扩充单元或 5040 扩展单元上的 DC 大容量模块 2 上有故障时会发送此信息，其它的 DC 大容量模块也能引起这种故障。

CPP962B　大容量 DC-3 故障
在 5042 扩充单元或 5040 扩展单元上的 DC 大容量模块 3 上有故障时会发送此信息，其它的 DC 大容量模块也能引起这种故障。

8.5.2　从 QSYSMSG 接收信息的例子

下面是从 QSYSMSG 接收信息的例子，程序由一个过程组成，它接收和处理 CPF1269。在 CPF1269 中的原因码是二进制格式，它必须要转换成十进制来与 704 和 705 比较，过程发出 ENDMOD 命令来防止启动新作业，然后发送同样的信息给用户定义的信息队列，来给安全管理员查看，也发信息给系统操作员告诉它发生了什么情况。如果收到不同的信息，它是发送给系统操作员。

可以启动几个作业来调用这个样板程序。作业可以保持活动，等待信息到达，也可用 ENDJOB 命令来结束。

```
/*********************************************************************/
/*                                                                   */
/*  从 QSYSMSG 接收信息的样板程序                                    */
/*                                                                   */
/*********************************************************************/
/*                                                                   */
/*  程序用原因码704或705来查找信息 CPF1269 ，如找到通知QSECOFR有    */
/*  安全故障。其他情况给QSYSOPR重发信息。                           */
/*  下面是 CPF1269 的信息描述：                                      */
/*      CPF1269:  启动程序需要接收由于原因码&6，&7拒绝的通讯设备&1，*/
/*      信息数据来自 DSPMSGD CPF1269                                 */
/*                                                                   */
/*        数据  类型     偏移     长度 说明                          */
/*                                                                   */
/*         &1   *CHAR       1      10  设备                         */
```

```
/*          &2   *CHAR     11      8  方式                          */
/*          &3   *CHAR     19     10  作业 - 号                      */
/*          &4   *CHAR     29     10  作业 - 用户                    */
/*          &5   *CHAR     39      6  作业 - 名字                    */
/*          &6   *BIN      45      2  原因码 – 较大的                */
/*          &7   *BIN      47      2  原因码 – 较小的                */
/*          &8   *CHAR     49      8  远程分配名                     */
/*          &9   *CHAR     57  *VARY  工作单元标识                   */
/*                                                                   */
/*******************************************************************/

          PGM

          DCL         &MSGID   *CHAR LEN(  7)
          DCL         &MSGDTA  *CHAR LEN(100)
          DCL         &MSG     *CHAR LEN(132)

          DCL         &DEVICE  *CHAR LEN( 10)
          DCL         &MODE    *CHAR LEN(  8)
          DCL         &RMTLOC  *CHAR LEN(  8)

          MONMSG      CPF0000  EXEC(GOTO PROBLEM)
          /*********************************************************/
          /* 从 QSYSMSG 信息队列给出的信息                         */
          /*********************************************************/

LOOP:     RCVMSG      MSGQ(QSYS/QSYSMSG) WAIT(*MAX) MSGID(&MSGID) +
                        MSG(&MSG) MSGDTA(&MSGDTA)

          IF          ((&MSGID *EQ 'CPF1269') /* Start failed msg */ +
            *AND       ((%BIN(&MSGDTA 45 2) *EQ 704)                +
              *OR       (%BIN(&MSGDTA 45 2) *EQ 705)) )             +
          THEN(DO)
            /*******************************************************/
            /* 通知 QSECOFR 安全故障                               */
            /*******************************************************/

            CHGVAR      &DEVICE %SST(&MSGDTA 1 10) /* Extract device */
            CHGVAR      &MODE   %SST(&MSGDTA 11 8) /* Extract mode   */
            CHGVAR      &RMTLOC %SST(&MSGDTA 49 8) /* Get loc name   */

            ENDMOD      RMTLOCNAME(&RMTLOC) MODE(&MODE)

            SNDPGMMSG   MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) +
```

```
                    TOMSGQ(QSECOFR)
  SNDPGMMSG  MSG('Device ' *CAT &DEVICE *TCAT ' Mode '  +
                    *CAT &MODE *TCAT ' had security failure, +
                    session max changed to zero')          +
                    TOMSGQ(QSYSOPR)
ENDDO
ELSE DO
  /*********************************************************/
  /* 其他信息 – 送给 QSYSOPR                                */
  /*********************************************************/

  SNDPGMMSG  MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) +
                TOMSGQ(QSYSOPR)

  /* SNDPGMMSG would fail if the message does            */
  /*   not have a MSGID or is not in QCPFMSG             */

  MONMSG     MSGID(CPF0000)  +
             EXEC(SNDPGMMSG MSG(&MSG) TOMSGQ(QSYSOPR))
ENDDO

GOTO       LOOP    /* Go fetch next message             */


/*********************************************************/
/* 通知 QSYSOPR 非正常结束                                */
/*********************************************************/

PROBLEM:  SNDPGMMSG  MSG('QSYSMSG job has abnormally ended') +
                     TOMSGQ(QSYSOPR)
          MONMSG     CPF0000

          SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
                        MSGDTA('Unexpected error occurred')
          MONMSG     CPF0000

          ENDPGM
```

8.6 使用系统回答列表

使用系统回答列表可以规定系统对预先定义的查询信息给出回答，这样就不须用户给出回答，它仅可对查询信息自动给出回答。系统回答列表包括信息标识，可选的比较数据，对每个信息回答的内容和一个转储属性。它仅对使用系统回答列表的作业发送的预先定义的查询信息给出回答。要在下列命令中规定 INQMSGRPY(*SYSRPYL)参数，才能使作业可用系

统回答列表：

BCHJOB
SBMJOB
CHGJOB
CRTJOBD
CHGJOBD

在使用系统回答列表的作业发送预先定义的查询信息时，系统用升序检索与信息标识相匹配的项，也可再比较回答信息的数据。如果有匹配项找到，给出指定的回答，用户就不用再回答，如果没有找到匹配项，信息送给交互作业的工作站用户或批作业的系统操作员。

下面是随系统一起安装的初始项定义：

顺序号	信息标识	比较值	回答	转储
10	CPA0700	*NONE	D	*YES
20	RPG0000	*NONE	D	*YES
30	CBE0000	*NONE	D	*YES
40	PLI0000	*NONE	D	*YES

此表指出，使用系统回答列表的作业发送的 CPA0700—CPA0799、RPG0000—RPG9999 CBE0000—CPE9999 和 PL10000—PL19999 的信息回答都为 D，且都做作业转储，系统要使用这些项，必须规定这些作业要使用系统回答列表。

要往系统回答列表增加查询信息，用 ADDRPYLE 命令，在此命令中要规定顺序号，信息标识，比较数据，比较数据的 CCSID，回答内容及转储属性，也可用 WRKRPYLE 命令来访问 ADDRPYLE 功能。

可在系统回答列表中给出下列回答（参数值放在括号中）：

对查询信息发送缺省回答（*DFT），这时，信息不显示，也不调用缺省处理程序。

要工作站用户或系统操作员来响应（*RQD），如果发信息的信息队列是中断方式，则显示信息，工作站用户必须回答，在没用系统回答列表时也这样做。

发送在系统回答列表中规定的回答信息。（信息回答，最多 32 个字符），这时发送规定的回答，不显示信息，不调用缺省处理程序。

下列命令往系统回答列表中加对 RPG1241、RPG1200、CPA4002、CPA5316 和其它查询信息的回答项。

```
ADDRPYLE SEQNBR(15) MSGID(RPG1241) RPY(C)

ADDRPYLE SEQNBR(18) MSGID(RPG1200) RPY(*DFT)  DUMP(*YES)

ADDRPYLE SEQNBR(22) MSGID(CPA4002) RPY(*RQD) +  CMPDTA('QSYSPRT')

ADDRPYLE SEQNBR(25) MSGID(CPA4002) RPY(G)

ADDRPYLE SEQNBR(27) MSGID(CPA5316) RPY(I) DUMP(*NO) + CMPDTA('QSYSPRT'
21)

ADDRPYLE SEQNBR(9999) MSGID(*ANY) RPY(*DFT)
```

现在的系统回答列表如下：

顺序号	信息标识	比较数据（b 代表空格）	比较的起始位置	回答	转储
10	CPA0700		1	D	*YES
15	RPG1241		1	C	*NO
18	RPG1200		1	*DFT	*YES
20	RPG0000		1	D	*YES
22	CPA4002	'QSYSPRT'	1	*RQD	*NO
25	CPA4002		1	G	*NO
27	CPA5316	'QSYSPRT'	21	I	*NO
30	CBE0000		1	D	*YES
40	PLI0000		1	D	*YES
9999	*ANY		1	*DFT	*NO

对使用这个系统回答列表的作业，在由作业发送加到回答列表中的信息时发生下列情况：

对顺序号 15，在作业发送 RPG1241 时，回答为 C，不做转储。

对顺序号 18，使用一般的信息标识。因此，在送出 RPG1200 时，给出缺省回答。缺省回答可以是在信息描述中规定的，也可是系统缺省回答，在发送回答前，转储作业，以前加的项替代了 RPG1241 的项。

对顺序号 22，如果发送比较数据 QSYSPRT 的 CPA4002，信息送往工作站用户，这时用户必须给出回答。在比较数据没有规定起始位置时，从信息中替代数据的第一个位置开始。顺序号 22 测试名为 QSYSPRT 的打印机。用不同起始位测试一个替代变量的例子，请看顺序号 27。

对顺序号 25，如果发送比较数据不等于 QSYSPRT 的 CPA4002，则回答为 G，不做作业转储。如果打印机为 QSYSPRT，则要求操作员做格式校正，顺序号 25 定义了对任何其它设备，如果发生格式校正的请求信息，用缺省值 G=GO 响应。

对顺序号 27，如果有比较数据为 TESTEDFILETSTLIBRARYQSYSPRT 的查询信息 CPA5316，则回答为 I。如规定了比较数据的超始位置，则从信息数据的起始位置比较。此时，位置 21 是第三个替代变量的起始位置，对 CPA5316 头四个替代变量为：

&1	ODP 文件名	*CHAR	10
&2	ODP 库名	*CHAR	10
&3	ODP 设备名	*CHAR	10
&4	第一行的行号	*BIN	2

这样，在送出回答前，顺序号 27 要测试名为 QSYSPRT 的 ODP 设备。

对顺序号 9999，对任何比它低顺序号的没有匹配项的预先定义查询信息，提供*ANY 的信息标识，并发送缺省回答。如果在系统回答列表中没有这项，那么就要求工作站用户回答。

在比较值中有*CCHAR 数据时，从发送功能来的信息数据在比较前要转换成存在系统

回答列表中的信息数据 CCSID。仅转换类型为*CCHAR 的数据，有关*CCHAR 的详细内容，请看 CL 参考手册。

要取消系统回答列表中的项，用 RMVRPYLE 命令，可用 CHGRPYLE 命令修改项的属性，也可用 WRK RPYLE 命令来显示当前在列表中的各项。

作业日志接收一个完成信息，表示成功的使用 ADDRPYLE、CHGRPYLE 和 RMVRPYLE 修改了系统回答列表，历史日志 QHST 也会接收一个完成信息来记录所做的修改。

8.7 信息日志

有两类信息日志：作业日志和历史日志。

作业日志包括与作业请求有关的信息，历史日志（QHST）包括系统数据，象作业起动和结束的历史信息。

8.7.1 作业日志

每个作业都有与之相关的作业日志，它包括下列内容：

作业中的命令

用 LOG(*YES)选项或 LOG(*JOB)选项生成的 CL 程序中的命令或用 LOGCLPGM（*YES）运行的 CHGJOB 命令。

所有发送给请求者的或没从调用信息队列取消的信息和信息帮助。

在作业结束后，作业日志可以写在输出文件 QPJOBLOG 中或一个数据库文件中，从 QPJOBLOG 中可以打印作业日志。从数据库文件中，可用数据库属性来查询作业日志的信息。你也可以规定对成功运行的作业不写到作业日志中。

往数据库文件中写作业日志要用 QMHCTLJL API。在作业日志直接写往数据库文件时，要生成一个或二个文件。主文件包括诸如信息标识，严重级，类型和数据这些特别的信息。次文件包括信息正文。次文件是可选的，它由 QMHCTLJL API 中的参数控制。这二个文件都能用外部描述说明，也都可用数据库文件和系统查询功能来处理。详细内容请看原文的附录 C.0。

可以控制哪 些信息写到作业日志中，在 CRTJOBD 命令中规定 LOG 参数，也可以用 CHGJOB 或 CHGJOBD 命令修改这些级别。LOG 参数有三个值：信息级，严重级和文本级。这些命令的详细内容，请看 CL 参考手册。

有下面这些信息级级别：

级别	说明
0	没有日志数据
1	只有严重级别大于或等于规定级别的信息，送往作业外部信息队列的所有信息要日志，这类信息指出作业何时开始，何时结束及完成时的状态
2	要在日志中记录下列信息： 日志级 1 的信息，任何从命令行输入的请求或从 CL 程序中发出的严重级别大于或等于规定级别的 CL 命令都要写进日志。日志了的命令或请求所有与之相关的信息也都记录
3	下列信息要记录： 日志级 1 的信息 所有从 CL 程序中记录的请求和命令 如果任何与请求或命令有关的高级信息，它的严重级别大于或等于规定级别，所有与之相关的信息都要记录
4	所有从 CL 程序中记录的请求或命令

仅记录严重码大于或等于规定级别的信息

注：一个高级信息是指发往程序调用信息队列的，从 CL 程序中记录的接收的请求或命令。

第二个值（严重级别）规定记录到日志中的引起错误信息的最小严重级别，可用值为 0—99。只有严重级别大于或等于这个值的信息才记录到日志中。

第三个值（文本级别）规定写在作业日志中信息正文的级别，它们可以是：

*MSG：在作业日志中仅写正文（不包括信息帮助）

*SECLVL：在作业日志中写正文和帮助

*NOLIST：如果作业正常结束，不写作业日志。如果作业以 20 或更高结束，产生包括信息和帮助的作业日志。

在由请求处理程序接收到每个新的请求之前，要做信息过滤。它根据信息日志级别，在作业日志中记录规定的信息，在程序中的每个 CL 命令之后不做过滤，这样，如果交互或批方式运行 CL 程序，那么在程序结束后才做过滤。

注：*NOLIST 规定在作业正常结束时不做日志，那么从规定日志级为零的批作业取消信息是对系统资源的浪费。

下例给出日志级别对存在作业日志中信息的影响情况，也给出在交互运行命令时，在每个命令之后过滤的发生。

注：在此例中，包括高级和详细的信息日志级，高级信息标识为信息，详细信息标识为详细信息。

1、CHGJOB 命令规定日志级为 2，信息严重级为 50，仅往作业日志中写信息（*MSG）。

```
                              Command Entry                        SYSTEM1
                                                    Request level:    1
Previous commands and messages:
     > CHGJOB LOG(2 50 *MSG)
```

2、PGMA 往自己的调用信息队列和规定调用（*PRV）的被调用程序的调用信息队列发送严重码为 20，50，和 60 的三个报告信息，PGMA 给自己发送的是详细信息，详细信息是送往低级程序调用的调用信息队列的信息。

PGMB 往自己的调用信息队列发送严重码为 40，50 的报告信息。它们也是详细信息，PGMB 也往*PRV 发送严重码为 10 的一个报告信息。

注意在这个显示中不再有 CHGJOB 命令，根据日志级别 2，仅严重码等于或大于规定的请求信息才写到作业日志中且此请求无信息发出。如果有这样的信息发出，那么发出的任何详细信息都写在作业日志中且在按 F10 键时会显示它们。

```
                              Command Entry                        SYSTEM1
                                                    Request level:    1
Previous commands and messages:
```

```
  > CALL PGMA
    Message sev 20 - PGMA
    Message sev 50 - PGMA
    Message sev 60 - PGMA
  > CALL PGMB
    Message sev 10 - PGMB




                                                                    Bottom
Type command, press Enter.
===> ____________________________________________________________________
____________________________________________________________________
____________________________________________________________________
____________________________________________________________________
 F3=Exit   F4=Prompt   F9=Retrieve   F10=Include detailed messages
 F11=Display full      F12=Cancel     F13=Information Assistant   F24=More keys
```

3、在命令入口显示中按 F10 键时，与此请求有关的所有信息都显示出来，再按 F10 键不会显示详细信息

```
                              Command Entry                        SYSTEM1
                                                      Request level:   1
All previous commands and messages:
   > CALL PGMA
     Detailed message sev  20 - PGMA
     Detailed message sev  50 - PGMA
     Detailed message sev  60 - PGMA
     Message sev 20 - PGMA
     Message sev 50 - PGMA
     Message sev 60 - PGMA
   > CALL PGMB
     Detailed message sev  40 - PGMB
     Detailed message sev  50 - PGMB
     Message sev 10 - PGMB

                                                                    Bottom
```

```
Type command, press Enter.
===> _____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
 F3=Exit    F4=Prompt    F9=Retrieve    F10=Exclude detailed messages
 F11=Display full      F12=Cancel    F13=Information Assistant    F24=More Keys
```

4、在输入另一命令时（CHGJOB），CALL PGMB 命令或其它信息都移出。这是因为与这个请求有关的高级信息严重码小于在 CHGJOB 命令中规定的严重码，而仍保留 CALL PGMA 命令和其有关的信息，这是由于这个请求发出的高级信息中至少有一个的严重码等于或大于规定的码。

在下个显示中，CHGJOB 命令规定了日志级为 3，严重码为 40。这样信息的第一和第二级正文都写在作业日志中。在进入另外的命令时，由于日志级 3 记录了所有请求，所以 CHGJOB 命令仍留在显示中。

PGMC 向规定了调用（*PRV）程序的调用信息队列发送严重码 30 和 40 的信息。PGMD 问*PRV 发送严重码为 40 的信息。

```
                                Command Entry                    SYSTEM1
                                                      Request level:   1
Previous commands and messages:
     > CALL PGMA
       Message sev 20 - PGMA
       Message sev 50 - PGMA
       Message sev 60 - PGMA
     > CHGJOB LOG(3 40 *SECLVL)
     > CALL PGMC
       Message sev 30 - PGMC
       Message sev 40 - PGMC
     > CALL PGMD
       Message sev 10 - PGMD

                                                                   Bottom
Type command, press Enter.
===> _____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
 F3=Exit    F4=Prompt    F9=Retrieve    F10=Include detailed messages
```

```
F11=Display full      F12=Cancel    F13=Information Assistant    F24=More Keys
```

5、在 CALL PGMD 后输入另外命令时，CALL PGMD 的保留在显示上，但与之有关的信息被删掉了，这是由于它的严重码小于在 CHGJOB 命令中 LOG 参数里规定的严重码。

输入的 SIGNOFF　*LIST 命令打印到作业日志中

```
                                  Command Entry                        SYSTEM1
                                                          Request level:   1
Previous commands and messages:
     > CHGJOB LOG(3 40 *SECLVL)
     > CALL PGMC
       Message sev 30 - PGMC
       Message sev 40 - PGMC
   > CALL PGMD
     > CALL PGME




                                                                        Bottom
Type command, press Enter.
===> SIGNOFF *LIST_____________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
 F3=Exit   F4=Prompt   F9=Retrieve   F10=Include detailed messages
 F11=Display full      F12=Cancel     F13=Information assistant    F24=More Keys
```

作业日志中包括留在显示上的所有请求和信息，另外，也包括与之有关的帮助信息，它是根据 CHGJOB 命令中规定来执行的。但它包括的帮助信息是在作业期间发出的，而不是进入第二个 CHGJOB 命令后发出的信息。

```
5763SS1 V2R3M0 930925                    Job Log                    SYSAS727 12/12/92 07:58:53        Page    1

 Job name . . . . . :   QPADEV0007     User . . . . . :   JOHNDOE     Number . . . . :   004201

 Job description . . . . . :   QDFTJOBD     Library . . . . . :   QGPL

MSGID     TYPE                SEV   DATE       TIME       FROM PGM     LIBRARY     INST    TO PGM      LIBRARY     INST

CPF1124   Information         00    12/12/93   07:57:16   QWTPIIPP     QSYS        04FC    *EXT                    0000
```

```
                                Message . . . . :   Job 004201/JOHNDOE/QPADEV0007 started on 12/12/92 at
                                  07:57:16 in subsystem QINTER in QSYS. Job entered system on 12/12/92 at
                                  07:57:16.
*NONE      Request                    12/12/93   07:57:50   QMHGSD          QSYS        0322     QCMD          QSYS        00B6
                                Message . . . . :  -call pgma
CPF1001    Information          20    12/12/93   07:57:50   PGMA            JOHNDOE     000C     PGMA          JOHNDOE     000C
                                Message . . . . :   Detailed message sev 20 - PGMA
                                CPF1001 second level text - PGMA
CPF1002    Information          50    12/12/93   07:57:50   PGMA            JOHNDOE     0010     PGMA          JOHNDOE     0010
                                Message . . . . :   Detailed message sev 50 - PGMA
                                CPF1002 second level text - PGMA
CPF1003    Information          60    12/12/93   07:57:50   PGMA            JOHNDOE     0014     PGMA          JOHNDOE     0014
                                Message . . . . :   Detailed message sev 60 - PGMA
                                CPF1003 second level text - PGMA
CPF1004    Information          20    12/12/93   07:57:50   PGMA            JOHNDOE     0018     QCMD          QSYS        00DE
                                Message . . . . :   Message sev 20 - PGMA
                                CPF1004 second level text - PGMA
CPF1005    Information          50    12/12/93   07:57:50   PGMA            JOHNDOE     001C     QCMD          QSYS        00DE
                                Message . . . . :   Message sev 50 - PGMA
                                CPF1005 second level text - PGMA
CPF1006    Information          60    12/12/93   07:57:50   PGMA            JOHNDOE     0020     QCMD          QSYS        00DE
                                Message . . . . :   Message sev 60 - PGMA
                                CPF1006 second level text - PGMA
*NONE      Request                    12/12/93   07:58:31   QMHGSD          QSYS        0322     QCMD          QSYS        00B6
                                Message . . . . :  -chgjob log(3 40 *seclvl)
*NONE      Request                    12/12/93   07:58:34   QMHGSD          QSYS        0322     QCMD          QSYS        00B6
                                Message . . . . :  -call pgmc
CPF100F    Information          30    12/12/93   07:58:34   PGMC            JOHNDOE     000C     QCMD          QSYS        00DE
                                Message . . . . :   Message sev 30 - PGMC
                                CPF100F second level text - PGMC
CPF1010    Information          40    12/12/93   07:58:34   PGMC            JOHNDOE     0010     QCMD          QSYS        00DE
                                Message . . . . :   Message sev 40 - PGMC
                                CPF1010 second level text - PGMC
*NONE      Request                    12/12/93   07:58:38   QMHGSD          QSYS        0322     QCMD          QSYS        00B6
                                Message . . . . :  -call pgmd
*NONE      Request                    12/12/93   07:58:45   QMHGSD          QSYS        0322     QCMD          QSYS        00B6
                                Message . . . . :  -call pgme
*NONE      Request                    12/12/93   07:58:52   QMHGSD          QSYS        0322     QCMD          QSYS        00B6
                                Message . . . . :  -signoff *list
CPF1164    Completion           00    12/12/93   07:58:52   QWTMCEOJ        QSYS        01EE     *EXT                      0000
                                Message . . . . :   Job 004201/JOHNDOE/QPADEV0007 ended on 12/12/92 at
                                  07:58:52; 3 seconds used; end code 0 .
                                Cause . . . . . :   Job 004201/JOHNDOE/QPADEV0007 completed on 12/12/92 at
                                  07:58:52 after it used 3 seconds processing unit time.  The job had ending
                                  code 0. The job ended after 1 routing steps with a secondary ending code of
```

0. The job ending codes and their meanings are as follows: 0 - The job completed normally. 10 - The job completed normally during controlled ending or controlled subsystem ending. 20 - The job exceeded end severity (ENDSEV job attribute). 30 - The job ended abnormally. 40 - The job ended before becoming active. 50 - The job ended while the job was active. 60 - The subsystem ended abnormally while the job was active. 70 - The system ended abnormally while the job was active. 80 - The job ended (ENDJOBABN command). 90 - The job was forced to end after the time limit ended (ENDJOBABN command). Recovery . . . : For more information, see the Work Management, SC41-8078.

在打印作业日志的每页顶部的标题给出是哪个作业的日志，以及每项的特性：

作业的完整限定名（作业名，用户名和作业号）
启动作业所用的作业描述名
作业启动的日期和时间
信息标识
信息类型
信息严重级别
每个信息发送的日期和时间，不包括请求信息
信息。如果规定要包括二级信息文本，它出现在信息的后读行中
发送信息或请求的过程或程序
发送信息或请求的过程或程序的机器接口指令号或高级语句号
信息发送给哪个过程或程序
信息发送给它的过程或程序的机器接口指令号或高级语句号

8.7.1.1 发送或接收的过程或程序

当发送或接收的是一个 ILE 过程时，信息项包括过程的全名（过程名，模块名和 ILE 程序名）。当发送或接收的是一个 OPM 程序，仅给出程序名，这时，相应的指令号表示一个指令号，且仅有这一个号码。如果是 ILE 程序，指令号表示的是一个高级语言语句号而不是 MI 指令号。如果 ILE 过程已优化，可能有三个号码，这也不总能用它来确定优化过程的语句号。假如给出多个号码，每个号码表示发送信息过程的一个可能点，也可能不能确定哪个号码，这时，信息中出现*N。

日志级别影响批作业的情况如前例所示。如果作业用 APPC，标题行包括 APPC 标识的工作单元。

8.7.1.2 附加的信息过滤

如果作业日志是通过 QMHCTLJL API 直接写到一个数据库存文件中，可以规定附加的信息过滤，用这个 API 规定的信息过程是在作业结束，往文件中写记录时完成的，这时，出现过滤的信息，这样在作业运行时可以看到它们，在写作业日志时，被过滤的信息就不写到文件时，这样，虽然在作业运行时它们可以出现但不出现在最终产生的文件中。

8.7.1.3 显示作业日志

显示作业日志的方法依赖于作业的状态。

如果作业已经结束，作业日志还没有打印，用下面的命令：

```
DSPSPLF FILE(QPJOBLOG) JOB(001293/FRED/WS3)
```

如果作业仍活动或者是在作业队列中还没有启动，用下列令：

```
DSPJOBLOG JOB(nnnnnn/JSMITH/WS1)
```

它显示一个用户为 JSMITH 的交互作业的日志，它是在工作站 WS1 上做的。nnnnnn 是作业号。

要显示你自己交互作业的作业日志，做下列之一：

用 DSPJOBLOG 命令

用 WRKJOB 命令，然后选择选项 10

如果终端上的‘禁止输出’灯亮且保持不灭，则做下面操作：

1、用系统请求键，然后用执行键

2、在系统请求屏上，选 3

3、在显示作业屏上，选 10

4、在作业日志屏上，出现 DSPJOB 做为处理的请求，用 F10 键

5、在此显示中，用上、下翻页键看从按系统请求功能键后接收到的信息

注销工作站，在 SIGNOFF 命令中规定 LOG（*LIST）

在使用 DSPJOBLOG 命令时，可以看到作业日志的显示，这个显示给出带有特殊符号的程序名，如下所示：

》 运行的命令或下一个要运行的命令。例如，要调用一个程序，显示调用的这个程序

〉 命令已经完成了处理

·· 还没有处理的命令

？ 回答信息。它标识需要回答的信息和已经回答的信息

在作业显示中，可做下列操作：

用 F10 键显示详细信息，这个显示给出在 HLL 程序中或在 LOG 活动的 CL 过程或程序中运行的命令或操作。

用光标移动键到达作业日志的末尾，可用 F18 键直接到末尾。在按 F18 键之后，可用翻页键来看到正运行的命令。

用光标移动键走到作业日志的顶部，也可用 F17 键直接到达。

可用 DSPJOBLOG 命令把作业日志直接写到数据库文件中，而不是打印或显示。这里有两个可用选项，用第一个选项，可规定文件和成员名，这时，把作业日志信息写到规定的数据库文件中。用第二个选项可以使用与前面运行的 QMHCTLJL API 提供信息有关的命令。这时，作业日志写到在 API 调用所规定的文件中，同时产生主、辅二个文件，且在信息写往文件时也完成过滤操作。用这二个选项，在 DSPJOBLOG 完成时，不显示输出，也没有打印用的假脱机文件。

8.7.1.4 避免产生作业日志

要在批作业结束时不产生作业日志，可在 BCHJOB、CHGJOB、SBMJOB、CRTJOBD、CHGJOBD 命令中的 LOG 参数里规定信息文本级的值为*NOLIST。这样，作业结束时不产生作业日志，除非结束码是 20 或更大，此时需产生作业日志。

对交互作业，在 SIGNOFF 命令中 LOG 参数规定的值要优于作业中规定的 LOG 值。

8.7.1.5　作业日志的有关事宜

在使用作业日志时，有以下建议：

要修改系统中所有作业的输出队列，用 CHGPRTF 命令修改 QSYS/QPJOBLOG 文件中的 OUTQ 或 DEV 参数。下面是命令的例子：

```
CHGPRTF FILE(QSYS/QPJOBLOG)
        DEV (USRPRT)
```

或

```
CHGPRTF FILE(QSYS/QPJOBLOG)
        OUTQ(USROUTQ)
```

要修改 QPJOBLOG 文件来使用输出队列 QEZJOBLOG，用操作助手的清理功能。在使用作业日志的自动清理功能时，打印文件必须在这个输出队列中。有关操作助手功能，请看系统操作。

要规定作业日志写到哪个输出队列，保证文件 QPJOBLOG 有 QUTQ(*JOB)，也可用 BCHJOB、CRTJOBD、CHGJOBD 或 CHGJOB 命令的 QUTQ 参数。下面是例子：

```
CHGJOB  QUTQ(*JOB)
```

如果在作业开头修改了缺省的 QUT ，对所有的假脱机文件都有影响。如果在作业完成这前修改，仅影响这个作业日志，不能用 OVRPRTF 命令来影响作业日志。

如果找不到作业的输出队列，则不能产生作业日志。

要保留所有的作业日志，用 CHGPRTF 命令规定 QSYS/QPJOBLOG 文件中的 HOLD（*YES）。在 RLSSPLF 命令运行时，释放作业日志给写出器。下面是命令的例子：

```
CHGPRTF FILE(QSYS/QPJOBLOG)
        HOLD(*YES)
```

如果系统异常结束，启动的提示允许系统操作员规定是否打印在异常结束时正活动的作业日志。要删除作业日志，用 DLTSPLF 命令或输出队列显示中的删除选项。

如果用 CHGPRTF 命令中的 USRDTA 参数修改 QSYS/QPJOBLOG 文件的用户数据值，规定的值不在‘处理输出队列’显示或‘处理所有假脱机文件’显示中出现，出现在用户数据列中的值是已打印作业日志的作业名。

如果要用程序设计技术分析作业日志，用 QMHCTLJL API 把作业日志写到数据库文件中。数据库文件中的记录格式是有保证的而打印格式没有。如果要在作业日志记录中增加新字段，它们是加在记录的末尾，这样已有的程序会继续工作，由系统提供的查询功能可直接用在这些文件上。

8.7.1.6　交互作业日志的考虑

IBM 提供的作业描述 QCTL，QINTER，QPGMR 都有一个 LOG（4　0　*NOLST）的日志级别。这样，所有信息和信息的第一、第二级文本都写到作业日志中，但如果在 SIGNOFF

时没规定*LIST 则不打印作业日志。要修改交互作业的日志级，可使用 CHGJOB 或 CHGJOBD 命令。

如果显示工作站用户使用 IBM 提供的菜单或命令入口显示，那么所有的错误信息都被显示，如果用户使用自己写的初始程序，任何非监控的信息会引起初始程序结束，产生作业日志。但如初始程序监控了信息，在接收信息同时也接收了控制权。这时，要保证产生作业日志，这样才能确定错误原因。例如，初始程序显示一个包括注销选项的菜单，它的缺省值为*NOLIST。初始程序监控所有信息，也包括在发生异常时修改注销选项为*LIST 的 CHGVAR 命令：

```
            PGM
            DCLF MENU
            DCL &SIGNOFFOPT TYPE(*CHAR) LEN(7) VALUE(*NOLIST)
            .
            .
            .
            MONMSG MSG(CPF0000) EXEC(GOTO ERROR)
PROMPT:     SNDRCVF RCDFMT(PROMPT)
            CHGVAR &IN41 '0'
            .
            .
            .
            IF (&OPTION *EQ '90') SIGNOFF LOG(&SIGNOFFOPT)
            .
            .
            .
            GOTO PROMPT
ERROR:      CHGVAR &SIGNOFFOPT '*LIST'
            CHGVAR &IN41 '1'
            GOTO PROMPT
            ENDPGM
```

在上例中，如果发生异常，CHGVAR 命令修改 SIGNOFF 命令使其为*LIST，并把指示器设为 ON，这个指示器用来给出一个常量条件，它显示一个信息告诉用户产生异常的情况及要做什么。如果交 互作业是运行一个 CL 过程或程序，好么在下列条件之一为真且日志级别为 3 或 4 时才记录 CL 命令：

在 CRTCLPGM 中规定 LOG(*YES)

在 CRTCLPGM 中规定 LOG(*JOB)，且（*YES）是当前 LOGCLPGM 作业属性。

设置了 LOGCLPGM 属性且用 SBMJOB、CHGJOBD 和 CRTJOBD 命令令的 LOGCLPGM 参数修改过。

8.7.1.7　批作业日志的考虑

对批处理的应用程序，可能要修改要记录的信息总量。在 IBM 提供的子系统中给出的作业描述规定 LOG(4　0 *NOLIST)，即在作业异常结束时，有一个完整的日志。如果作业正常结束，则不产生日志，如果想在所有情况下都打印作业日志，用 CHGJOBD 来修改作

业描述或在 BCHJOB，SBMJOB 命令中规定不同的 LOG 值。

如果用批作业运行 CL 过程或程序，在用下列命令生成模块或程序时规定 LOG（*YES），则总是日志 CL 命令：

CRTCLPGM、CRTCLMOD、CRTBNDCL

在用 CHGJOB 或 SBMJOB 命令时规定 LOGCLPGM(*YES)也记录 CL 命令。

8.7.2 QHST 历史日志

历史日志（QHST）由一个信息队列和叫做日志版本的物理文件组成，送往历史日志信息队列的信息是由系统写到当前日志版本物理文件中。

QHST 包括系统功能的高级跟踪，象系统、子系统以及作业信息，设备状态和系统操作员信息，它的信息队列是 QHST。当日志版本满时，自动生成新的版本，每个版本是下列格式的物理文件：

Qxxxyydddn

其中：

xxx　　是 3 个字符的 HST

yyddd　是日志中版本中第一个信息的儒略日期

n　　是儒略日期中的顺序号（A—Z 和 0—9）

注：每个版本中的记录数是由系统值 QHSTLOGSIZ 规定的。

日志版本文件的说明由其中第一个信息和最后一个信息的日期和时间组成。第一个信息的日期为说明中的 1—13 位，最后一个信息的日期为 14—26 位，格式为

cyymmddhhmmss

其中：c 为世纪数字

yymmdd　为发送信息的日期

hhmmss　为发送信息的时间

你可以生成一个同样忽略日期的最多 36 位长的日志版本。如果在同一天生成的长于 36 的版本，那么顺序用下一个可用的日期做下个版本名字。如果删除了一些老版本，则它的名字可重用。这时，如果用名字给日志版本排序，则打乱了顺序。也可写一个程序来处理历史日志记录。由于可用的日志版本有几个，所以必须选择要处理哪一个，要确定可用的版本，用 DSPQBJD 命令。

下面的命令显示可利用的历史日志版本：

```
DSPOBJD OBJ(QSYS/QHST*)    OBJTYPE(*FILE)
```

可用 WRKOBJ 命令显示中的删除选项来删掉系统中的日志，要用自动删除方法,请看 QUSRTOOL 中的 DLTLOG 命令。

可用 DSPLOG 来显示或打印日志中的信息，可规定下列内容来显示或打印选择的信息：

时间周期

发送信息的作业名

信息标识

下列的 DSPLOG 命令显示当前日期作业 ODEAILY 的所有可用信息：

DSPLOG　　JOB(ODEAILY)

显示结果为：

Display History Log Contents

Job OEDAILY started
Database file OEMSTR in library OELIB expired
Job OEDAILY abnormally ended
Job OEDAILY started
Job OEDAILY ended

Bottom

Press Enter to continue.

F3=Exit　F10=Display all　F12=Cancel

如果把系统日期和时间设为以前日期，或往前设 48 小时，那么会开始一个新的日志版本，这就保证在某一个日志版本中的所有信息是按年日顺序的。

在 V3.6 版本前生成的日志版本,如果在系统日期和时间重设为前面某个时刻时可能有不按年日顺序的项。这样，在显示日志时，可能会丢失某些项不显示。例如，日志中包括 1988 后面有 1987 的项，你要显示 1987 的项，那就要在 DSPLOG 中的 PERIOD 参数里给出 1987，而有些想要的项就不能显示出来。这时要使用 QTATE 和 QTIME 或象下面那样规定 PERIOD 参数：

PERIOD((起始时间　起始日期) (*AVAIL *END))

在信息队列已满或用 DSPLOG 命令时，系统会把信息写到当前版本的物理文件中。如果想确保当前版本是当前日期，要在 DSPLOG 中规定一个虚构的信息标识。比如###0000，虽然不显示信息，但使日志版本的物理文件是当前的。

如果用 DSPLOG　QUTPUT(*PRINT)来打印日志信息，则每个信息只打印一行，即打印每个信息的前 105 个字符。

如果用 DSPLOG　QUTPUT(*PRTWRAP)来打印日志信息，则多于 105 字符的信息会卷回来打其余的行，但限制在 2000 个字符之内。

如果用 DSPLOG 显示日志信息，则只显示 105 个字符，在 105 个之后的字符从右边截断。

8.7.3　历史日志的格式

用一个数据库文件来存储系统中发送给日志的信息，由于物理文件中的所有记录都有相同的长度而送往日志中的信息有不同的长度，所以信息要跨过几个记录，每个记录有三个字段：

系统日期和时间（8 位长字符字段）。是个内部字段，转换的日期和时间也放到信息中

记录号（2 字节字段）。例如，第一个记录号为十六进制 0001，第二个记录号为 0002，依次类推

数据（132 位长字符字段）

第一个记录的第三个字段有下列格式：

内容	类型	长度	记录中的位置
作业名	字符	26	11-36
转换的日期和时间(1)	字符	13	37-49
信息 ID	字符	7	50-56
信息文件名	字符	10	57-66
库名	字符	10	67-76
信息类型(2)	字符	2	77-78
严重码	字符	2	79-80
发送程序名(3)	字符	12	81-92
发送程序指令号(4)	字符	4	93-96
接收程序名(3)	字符	10	97-106
接收程序指令号(4)	字符	4	107-110
信息正文长度	二进制	2	111-112
信息数据长度	二进制	2	113-114
数据或说明的 CCSID(5)	二进制	4	115-118
保留	字符	20	119-142

⑴格式为 cyymmddhhmmss。

其中：c 为世纪数字（如果 yy>40，则 C=0，如 yy<40，则 C=1）

yymmdd：信息发送的年月日

hhmmss：信息发送的时分秒

⑵与 RCVMSG 中的 RTNTYPE 有相同的值。

⑶如果发送或接收者是 ILE 过程，在历史日志中只包括 ILE 程序名，不包括模块名和过程名。

⑷如果发送和接收者是过程，则发送和接收指令号为 0。

⑸如果此信息是被存储的信息，则 CCSID 仅适应于定义做*CCHAR 数据的信息数据。其余的信息数据可以考虑为 65535，否则，这是立即信息的 CCSID。

其余记录的第三个字段有下列格式：

内容	类型	长度
信息	字符	可变（1）

信息数据	字符	可变（2）

⑴这个长度在第一个记录中规定（位置 111-112），且不能超过 132

⑵这个长度在第一个记录中规定（位置 113-114）

在启动一个新的日志版本时，不能把一个信息分开，信息的第一个记录和最后一个记录要在同一个版本中。

对信息数据的详细说明，请看 7.2.4。

8.7.4 处理 QHST 文件

如果用 HLL 程序来处理 QHST 文件，要注意每个信息的长度都可以不相同。由于信息包括可替代的变量，信息长度是可变的，这样，信息数据的起始位置也是可变的。

8.7.5 QHST 作业起动和完成信息

系统对作业起动和完成信息给出特别的格式。对 CPF1124（起动）和 CPF1164（结束），信息数据总是从第三个记录的位置 11 开始。作业计帐提供比 CPF1124 和 CPF1164 更多的信息。一个简单的计帐功能用 CPF1164 信息。

如果处理 QHST 中的信息，使用 QUSRTOOL 中的 CUTQHST 命令，它生成一个处理 QHST 信息的定长的外部描述文件。

在 CPF1164 中并不显示性能方面的信息。由于是在 QHST 中的信息，用户可以写一个应用程序接收这些数据，性能信息的格式如下所述。

性能信息做为一个变长替代文本值传送，即数据是从数据长度的第一位开始有效，长度字段的尺寸不包括在此长度内，结构中的第一个数据字段是作业进入系统和作业第一个例程启动的日期和时间。时间格式为 hh:mm:ss。日期格式由系统值 QDATFMT 和 QDATSEP 确定。在结构中，作业进入系统的日期和时间放在作业启动日期和时间的前面。

作业进入系统的日期和时间是系统初始化作业的时候。对一个交互作业，作业进入的时间是系统认识它的口令的时候。对批作业是处理 BCHJOB 或 SBMJOB 的时间，对于监控作业，是处理相应的启动命令的时间，对自动启动的作业是启动子系统的时间。

在时间和日期之后是总的响应时间和交易数。响应时间以秒计，包括在工作站按执行键和显示下一个屏幕之间所有处理交互作业的总计时间，这些信息很类似于 WRKACTJOB 显示的内容，这个字段仅对交互作业有意义。

在系统故障或作业异常结束时最后一个交易可能不包括在总计中。这时的作业结果码应该是 40 或更大。交易计数也仅仅对交互作业有意义。响应时间数是由系统作业期间间隔计数的。

在交易数后边的是同步辅存 I/O 操作的数目。这与在 WRKACTJOB 显示中的 AUXIO 字段出现的值是相同的。（除是作业总值之外）。如果作业以 70 结束，这个值也不包括最终作业步的计数。另外，如果一个作业跨越 IPL 存在（用 TFRBCHJOB 命令），那么在 IPL 后、成为活动前结束，这个值为零。

在性能状态的最后字段是作业类型。这个字段的值是：

A 自动起动的作业
B 批作业
I 交互作业
M 子系统监控
R 假脱机阅读器
S 系统作业

W　假脱机写入器

X　起动作业

对从一个可变位置开始信息数据的信息，可做下列操作来访问信息数据：

确定信息中变量的长度。例如，一个信息使用下列五个变量：

作业名　*CHAR　10

用户名　*CHAR　10

作业号　*CHAR　6

时间　　*CHAR　8

日期　　*CHAR　8

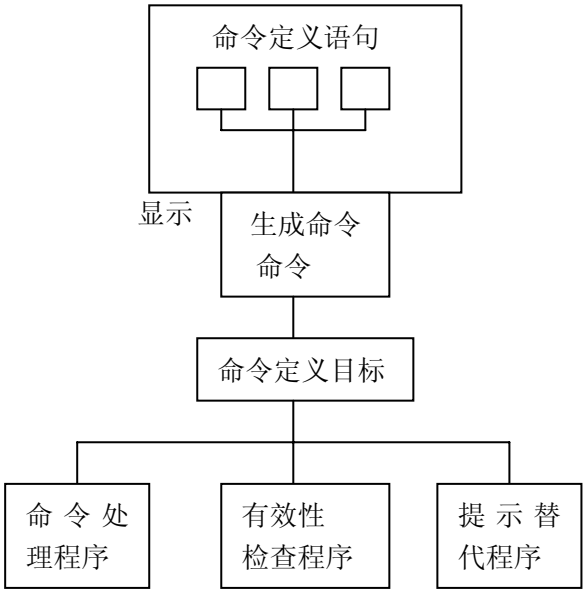
QUSRTOOL 库中的 DLTQHST 工具给出清理老 QHST 文件的批方式。请看 QUSRTOOL 库中的 QATTINFO 里的成员 DLTQHST。

第九章　定义命令

一个 CL 命令是请求系统完成一定功能的语句，这个功能是在进入命令时由运行的程序完成的。CL 命令允许你请求一个大范围的功能。使用这些命令就象它们是 IBM 提供的一样，也能修改由 IBM 提供的缺省值。这章介绍如何定义和生成自己的命令。

9.1　定义命令简介

下面解释生成命令的步聚，接下来有文字介绍每步的作用。



有效性检查程序和提示替代程序是可选的。

9.1.1　各步说明

9.1.1.1　命令定义语句

命令定义语句包括对工作站用户输入的提示，检查输入的有效性，及在命令运行时传给被调用程序的值的定义。

命令定义语句可以放在作为 CRTCMD 命令输入的任何文件中。例如，SEU 源文件，软盘，及其它可以包括命令定义语句的设备文件中，它们通常是由 SEU 输入到源文件中。表 9-1 给出定义命令所用的语句。

表 9-1 定义 CL 命令的语句

语句类型	语句名	解释
命令	CMD	对命令名规定提示文本
参数	PARM	定义参数或键字参数
元素	ELEM	定义用作参数值的列表中的元素
限定	QUAL	定义用作参数的限定名
关系	DEP	定义参数之间的关系
提示控制	PMTCTL	定义提示参数的条件

9.1.1.2 CRTCMD 命令

它处理命令定义语句，生成命令定义目标，CRTCMD 可在交互作业或批作业中运行。

9.1.1.3 命令定义目标

命令定义目标是由系统程序检验的确保命令的有效性及输入适当的参数。

9.1.1.4 有效性检查

系统对命令做有效性检查，你也可以自己写检查程序。

由系统完成的有效性检查要保证：

要输入必须的参数值

每个参数值要附合规定的类型和长度

每个参数值要附合在命令定义中规定的可选要求：

—有效值列表

—值的范围

—值的关系比较

不能输入冲突的参数

在下列时候，系统做有效性检查：

从工作站交互地进入命令

用假脱机从批输入流进入命令

用 SEU 把命令输入到数据库文件中

由一个 HLL 调用把命令传给 QCMDEXC、QCMDCHK、QCAPCMD 时

生成 CL 模块或 OPM 程序

由 CL 过程或程序或 REXX 过程运行命令

用 C 语言系统功能运行命

如果要做比系统做的多的检查，可以自己写程序（看 9.11），或在命令处理程序中加上检验功能。在 CRTCMD 命令中规定命令处理程序的名字和检查程序的名字。

如果自己写检查程序，那么命令的参数值要先传给有效性检查程序然后再给命令处理程序，在下列条件下要在对假脱机输入流做语法检查时运行写的程序：

对命令所有参数用常量代替变量

用 SEU 把命令输入到 CL 数据库文件中

在程序发现错误时，用户会接收到信息并允许立即改正错误。命令处理程序假定传给它的数据是正确的。要发送系统信息，有效性检查程序必须从信息队列接收信息，把它放在CPD0006 的替换变量&2 中，操作系统使用信息变量&1 的头四个字符。

9.1.1.5 提示替代程序

写一个提示替代程序在提示命令时提供参数缺省的当前值，详细内容请看 9.6，这个程序是可选的。

9.1.1.6 对命令提供帮助信息

可以使用帮助面板组为命令提供帮助信息。面板组是一个类型为*PNLGRP 的目标。详细内容请看应用显示程序设计一书。

9.1.1.7 命令处理程序

命令处理程序 CPP 是命令调用完成功能请求的程序。CPP 可以是 CL、REXX 或 HLL 程序，它可以是命令调用的应用程序，也可以是 CL 程序或 REXX 过程。它包括系统命令和一系列命令。

9.1.2 定义命令需要的权限

对要使用你生成的命令的用户，他们必须对命令有操作权，对命令处理程序和有效检查程序有数据数，对命令所在的库有读的权限，对命令处理程序和有效性检查程序有读的权限。如果命令处理程序或有效性检查程序引用任何服务程序，用户对服务程序及其所在的库要有执行权，用户对下面列出的程序要有执行权：

命令处理程序（CPP）
有效性检查程序（VCP）
由 CPP 或 VCP 使用的任何服务程序
CPP、VCP 和服务程序所在的库

如果其它命令在命令处理程序中运行或打开文件，用户也必须对这些命令处理程序或文件有适当的权限。

9.1.3 生成命令的例子

如果要生成一个命令，让系统操作员调用一个程序来启动系统。可以按下列步骤做：（此例假定使用 IBM 提供的源文件）

1、往源文件 QCMDSRC 中的成员 STARTUP 输入源语句：

```
CMD  PROMPT('S Command for STARTUP')
```

2、用下列命令生成命令

```
CRTCMD  CMD(S)  PGM(STARTUP)  SRCMBR(STARTUP)
```

3、输入源语句命令处理程序

```
PGM
STRSBS QINTER
```

```
STRSBS QBATCH
STRSBS QSPL
STRPRTWTR DEV(QSYSPRT) OUTQ(QPRINT) WTR(WTR)
STRPRTWTR DEV(WSPR2) OUTQ(WSPRINT) WTR(WTR2)
SNDPGMMSG MSG('STARTUP procedure completed') MSGTYPE(*COMP)
ENDPGM
```

4、用下列命令生成 CL 程序

```
CRTBNDCL  STARTUP
```

在上面例子中，S 是新命令的名字（由 CMD 参数规定）。STARTUP 是命令处理程序的名字（由 PGM 参数规定），也是包括命令定义语句的源成员名（由 SRCMBR 参数规定）。现在，系统操作员可以进入 S 来调用命令或用 CALL STARTUP 来调用命令处理程序。

9.2 定义命令

要定义命令，首先要用命令定义语句定义命令。这些命令在 CL 参考一书中有详细介绍，以下介绍它的一般格式和编码原则。

语句	编码原则
CMD	仅能用一个 CMD 语句，它可以放在源文件的任何地方
PARM	最多允许 75 个 PARM，在源文件中 PARM 的顺序决定了在处理时的顺序。每个参数都要有一个 PARM，它要传给命令处理程序，要把一个参数规定为键字参数，必须在 PARM 中规定 KEYPARM(*YES)。有键字定义的 PARM 个数由定义要修改的目标的唯一需要来限制。要使用键字参数，在生成命令时必须规定提示替代程序，它不允许与提示控制同用。
ELEM	在一个列表中最多可有 300 个 ELEM，它在源文件中的顺序决定了列表中元素的顺序。第一个 ELEM 语句必须有一个语句标号，它与在 PARM 中的 TYPE 语句或列表中的 ELEM 语句的标号匹配。
QUAL	对一个限定名最多有 300 个限定，它在源文件中的顺序决定了规定限定的顺序和传送给命令处理程序和有效性检查程序的顺序。
DEP	DEP 语句必须放在它引用的所有 PARM 语句后边。这样，DEP 语句通常都放在源文件接近结束的地方。
PMTCTL	必须放在它引用的所有 PARM 语句后边。这样，PMTCTL 通常放在源文件要结束的地方。

在源文件中，至少要有一个 PARM 参数，它放在 ELEM 或 QUAL 语句前。输入命令定义语句的源文件用做 CRTCMD 命令的源语句来生成命令。

9.2.1 使用 CMD 语句

在定义一个命令时，在命令定义语句中必须要有一个且仅能有一个 CMD 语句。

在定义一个命令时，可对用户提供命令提示说明。如果用户选择要提示代替整个命令的输入，那么可输入命令名然后按 F4 键，显示命令名，且在显示的第一行给出标题提示说明。可在 CMD 中用 PROMPT 语句来规定提示的标题，然后规定参数的提示，列表元素和限定，它们分别在 PROMPT 语句中用 PARM，ELEM 和 QUAL 语句。

在 PROMPT 语句中，可规定最多 30 个字符的串做为提示标题说明，或者在信息描述中

规定信息标识。在下例中，对命令 ORDENTRY 规定一个字符串：

```
CMD PROMPT('Order Entry')
```

在用户输入命令名且按 F4 键后，提示的第一行象下面那样：

```
Order Entry (ORDENTRY)
```

如果对定义的命令没规定提示说明，就可仅用 CMD 语句，也可在文本中用 PROMPT 键字来规定提示。

9.2.2 定义参数

每个命令可最多定义 75 个参数，要定义参数，必须用 PARM 语句。

在此语句中，可以规定以下内容：

参数的键字名
参数是否为键字参数
要传送的参数值的类型
值的长度
如果需要，参数值的缺省值

另外，在定义参数时，要考虑以下方面的事情：（括号中是 PARM 语句的参数）：

是否由命令处理程序返回值：如果规定 RTNVAL(*YES)，那么在调用命令时必须要规定返回变量。如果没有规定返回变量，则给命令处理程序传送空指针
参数是否不出现在用户提示中而做为常量传给命令处理程序（CONSTANT）
是否限制参数为某些值(VALUES，SPCVAL，SNGVAL)。或可以是与参数类型、长度、值范围或特定关系相匹配的值
规定有效的参数值（VALUES，SPCVAL，SNGVAL）
确定参数是否要检查，要做什么样的检查（REL 和 RANGE）
参数是必须的还是可选的（MIN）
在一个列表中，一个参数可以规定多少个值（MIN 和 MAX）
是否有不能打印的数据（ALWUNPRT）
参数值是否可为变量名（ALWVAR）
值是否为数据区名（DTAARA）
值是否为文件名（FILE）
值是否必须是确定长度的（FULL）
值的长度是否为给定值（VARY）
值是否可为表达式（EXPR）
传送给参数的值是否包括给出的属性信息（PASSATR）
如果没规定参数定义，是否向命令处理程序和有效性检查程序传送值（PASSVAL）
大小写值是否被保护或是否转换成大写（CASE）
在列表位移中的列表为二字节或四字节二进制数（LISTDSPL）
信息标识或提示说明（PROMPT）
在提示显示中可选字段的有效值（CHOICE）
选择值是否由程序提供（CHOICEPGM）
参数提示是否由另外的参数控制（PMTCTL）

PMTCTL 的值是否由程序提供（PMTCTLPGM）
值是否在作业日志中或在提示命令中隐藏（DSPINPUT）

9.2.2.1　命名参数的键字

为参数选择的键字名要给出参数值所必须的信息。例如用户名用 USER，比较值用 CMPVAL，顺序项类型为 OETYPE 等。这些键字最多 10 个字母数字字符，第一个字符必须是字母。

9.2.2.2　参数类型

基本的参数类型为：

十进制（*DEC）。它做为压缩十进制数，用 LEN 中规定的长度传送给命令处理程序。当给出值比定义值长时，发生截断。

逻辑值（*LGL）。“1”或“0”，做为长度为 1 的字符串（F1 或 F0）传给命令处理程序。

字符（*CHAR）。要放在引号中，用 LEN 中规定的长度传给命令处理程序。传送时去掉引号，左对齐，用空格填充。

名字（*NAME）。字符串表示一个基本名，最大长度为 256 个字符。第一个字符为 A-Z、$、#或@，其余的除此之外可以是数字 0-9，-， •， 名字可用双引号括起，以 LEN 中规定的长度做字符串传送给命令处理程序，左对齐，空格填充，通常用它做目标名。如果参数的值为类似*LIBL 或*NONE，必须用 SPCVAL 参数说明这特殊值。这样如果用户把这些值做为名字参数值，则不管名字验证的原则。

简单名（*SNAME）。字符串，规则与*NAME 相同，但不允许有逗号。

通讯名（*CNAME）。字符串，规则与*NAME 相同，但不允许有逗号和-号。

路径名（*PNMAE）。字符串，用在 LEN 中规定的长度放在引号中以字符串传送给命令处理程序。传送时去掉引号，左对齐，空白填充。使用规则请看集成文件系统介绍一书。

类属名（*GENERIC）。它以*结尾。如果不这样，就假定它为完整的目标名，它标识一组名字以*前所有字符组成的目标。它传给命令处理程序来找到名字以类属名字规定开头的那些目标。

日期（*DATE）。以字符串传给命令处理程序。字符串用 cyymmdd 格式传送。系统根据命令中的日期规定中的年来设置 c，如果年为四位数，且以 19 开头则 c 为 0，如果年为 20 开头则 c 为 1。对二位数的年，如果 yy 为 40-99，则 c 为 0，若 yy=00-39，则 c=1。用户必须用作业属性中的 DATFMT 格式输入命令的日期参数。作业属性的 DATSEP 确定输入日期时的可选分隔符。用 CHGJOB 命令可以修改 DATFMT 和 DATSEP 属性。程序用 1，1940 到 31，2039 的范围读两位数字的年，四位数字年日期的范围要在 8 月 24，1928 到 5 月 9，2071 范围内。

时间（*TIME）。用 nnmmss 格式的字符串传送给命令处理程序。

十六进制（*HEX）。规定的字符为 0-F，做为十六进制的 EBCDIC 字符传给命令处理程序（每字节十个十六进制数字）。右对齐，零填充。如果把值放在引号中，需要数字成对出现。

零元素（*ZEROELEM）。考虑做一系列零元素，即在命令中没有规定值。这个类型用于避免通过命令输入命令处理程序希望的值。例如，两个命令使用同一个命令处理程序，一个命令传送给它了一个参数列表，另一个命令就不用给命令处理程序传送任何值了，那么第二个命令的参数就可以规定TYPE(*ZEROELEM)。

整数（*INT2 或*INT4）。做为二字节或四字节有符号的二进制数传给程序。CL 程序不支持变量中用十进制数。

空（*NULL）。参数值为空指针，它做为一个空位传给命令处理程序，仅 KWD、MIN 和 MAX 可用这个参数类型。

命令串（*CMDSTR）。参数值命令，做为命令串传给命令处理程序，输入的 CL 变量不允许此类型。

语句标号。它指出一系列 QUAL 或 ELEM 语句中的第一个语句，其后要进一步说明限定名或由 PARM 定义的混合列表。

下面的参数类型仅适用于 IBM 提供的命令：

表达式（*X）。参数值为字符串，变量名或数值。如果它仅包括数字，+，-号或小数点，则按数字值传送，否则，按字符串传送。

变量名（*VARNAME）。做为字符串传送命令处理程序。左对齐，空白填充。在处理时它要引用一个实际的数据值，变量名最多长 10 位以&开头，后边第一个字符必须是字母数字。如果变量名不遵循 AS/400 系统的命令规则，必须用引号括起。

命令（*CMD）。参数值是命令。例如，CL 命令 IF 有一个 THEN 的参数，它的值必须是另外的命令。

9.2.2.3　参数值的长度

在下列类型的参数可以用 LEN 规定长度。日期型长度总为 7 而时间总为 6。下表给出每种类型参数的最大长度和它的缺省长度。

数据类型	最大长度	缺省长度
*DEC	24 (9 位小数)	15 (5 位小数)
*LGL	1	1
*CHAR	5000	32
*NAME	256	10
*SNAME	256	10
*CNAME	256	10
*GENERIC	256	10
*HEX	256	1
*X	(256 24 9)	(1 15 5)
*VARNAME	11	11
*CMDSTR	20K	256
*PNAME	5000	32

最大长度是指在命令运行时可给出的这类参数的最大允许长度。但在命令定义语句中允许的字符常量最多为 32 个字符。这个限制对 CONSTANT，DFT，VALUES，REL，RANGE，

SPCVAL 和 SNGVAL 语句都有效。在提示 CL 命令时，输入字段是有特别规定的长度的，它们是 1-2 个字符以及 17，25，32，50，80，132，256 和 512 个字符。如果个别参数有不允许的长度，输入字段显示下一个长一点的字段长度，提示显示的 512 字符长的输入字段能比 512 个字符再长一些。

9.2.2.4 缺省值

如果定义一个可选参数，也可用 DFT 定义一个值，以便在用户不给命令输入参数值时使用，这个值叫做缺省值。缺省值要附合对这个参数值规定的要求（例如类型、长度和特殊值）。如果没有规定可选的参数值，可用下面给出的缺省值：

数据类型	缺省值
*DEC	0
*INT2	0
*INT4	0
*LGL	'0'
*CHAR	空格
*NAME	空格
*SNAME	空格
*CNAME	空格
*GENERIC	空格
*DATE	零('F0')
*TIME	零('F0')
*ZEROELEM	0
*HEX	零('00')
*NULL	空
*CMDSTR	空格
*PNAME	空格

9.2.2.5 定义参数的例子

下例给出为命令定义 OETYPE 参数的例子：

```
PARM    KWD(OETYPE)  TYPE(*CHAR)  RSTD(*YES) +
        VALUES(DAILY WEEKLY MONTHLY)  MIN(1) +
        PROMPT('Type of order entry:')
```

OETYPE 参数是必须的（MIN 是 1）。它的值被限制为 DAILY、WEEKLY 或 MONTHLY（RSTD 为*YES）。PROMPT 给出参数的提示说明。因为没有规定 LEN 但 TYPE 为*CHAR，所用缺省长度 32。

9.3 数据类型和参数的限制

下表给出根据参数类型确定的参数的有效组合。X 指出组合是有效的，给出的数字说明列在表的下部的注中。

	LE	RTNVA	CONSTANT	RST	VALUES	RE	RANG	SPCVA	SNGVA
*DEC	X	2	X	X	X	X	X	3	1
*LGL	X	2	X	X	X			3	1
*CHAR	X	2	X	X	X	X	X	3	1
*NAME	X		X	X	X	X	X	3	1
*SNAME	X		X	X	X	X	X	3	1
*CNAME	X		X	X	X	X	X	3	1
*PNAME	X	2	X	X	X	X	X	3	1
*GENERIC	X		X	X	X	X	X	3	1
*DATE			X	X	X	X	X	3	1
*TIME			X	X	X	X	X	3	1
*HEX	X		X	X	X	X	X	3	1
*ZEROELEM									
*INT2			X	X	X	X	X	3	1
*INT4			X	X	X	X	X	3	1
*CMDSTR	X		X						
*NULL	X								
STMT LABEL			X						X

注：1、仅在 MAX 大于 1 时有效，在命令处理程序是 REXX 过程时忽略 TO 值。传送给 REXX 过程参数的是输入的值或缺省值。

2、在命令处理程序是 REXX 过程时无效。

3、在命令处理程序是 REXX 过程时忽略 TO 值，传送给 REXX 过程参数的值是输入的值或缺省值。

	MI	MA	ALWUNPRT	ALWVAR	PG	DTAARA	FIL	FULL	EXP	VAR
*DEC	X	X		X		X				
*LGL	X	X		X		X	X	1		
*CHAR	X	X	X	X	X	X	X	X	X	1
*NAME	X	X		X	X	X	X	X	X	1
7*SNAME	X	X		X	X	X	X	X	X	1
*CNAME	X	X		X	X	X	X	X	X	1
*PNAME	X	X	X	X	X	X	X	X	X	1
*GENERIC	X	X		X	X	X	X	X	X	1
*DATE	X	X		X		X				
*TIME	X	X		X					X	
*HEX	X	X		X				X	X	
*ZEROELEM	X	X								
*INT2	X	X		X					X	
*INT4	X	X		X					X	
*CMDSTR	2	3		4						1
*NULL	2	3								

STMT LABEL	X	X			X					

注：1、在命令处理程序是 REXX 时忽略参数。

2、TYPE 为*NULL 时 MIN 的值不能大于 1。

3、TYPE 为*NULL 时或*CMDSTR 时，MAX 的值不能大于 1。

4、这个类型忽略 ALWVAR 的值，在参数类型为*CMDSTR 时不允许用 CL 变量。

	PASA	PASSVCAS	LISTDS	DSPIN	CHOI	CHOICE	PMTC	PMTCTL	PRO	T
*DEC	1	X	3	X	X	X	X	X	X	X
*LGL	1	X	3	X	X	X	X	X	X	X
*CHAR	1	X	3	X	X	X	X	X	X	X
*NAME	1	X	3	X	X	X	X	X	X	X
*SNAME	1	X	3	X	X	X	X	X	X	X
*CNAME	1	X	3	X	X	X	X	X	X	X
*PNAME	1	X	3	X	X	X	X	X	X	X
*GENERI	1	X	3	X	X	X	X	X	X	X
*DATE	1	X	3	X	X	X	X	X	X	X
*TIME	1	X	3	X	X	X	X	X	X	X
*HEX	1	X	3	X	X	X	X	X	X	X
*ZEROEL			3							
*INT2	1	X	3	X	X	X	X	X	X	X
*INT4	1	X	3	X	X	X	X	X	X	X
*CMDSTR	1		3	X	X	X	X	X	X	X
*NULL										
STMT LABEL		2	3			X	X	X	X	X

注：1、在命令处理程序是 REXX 时忽略参数。

2、在命令处理程序为 REXX 时，PASSVAL 传送无空格的键字或放在引号中的字符。

3、仅在类型为*CHAR 和*PNAME 时来允许 CASE(*MIXED)。

下表给出对 PARM、ELEM 和 QUAL 语句的有效参数组合或限制。表中行、列交叉为空的则表示没有限制且组合是有效的。交叉中有数字，请看表后的注。

	LE	RTNVA	CONSTANT	RST	DF	VALUE	RE	RANG	SPCVAL	SNGVA
LEN										
RTNVAL			1	1	1	1	1	1	1	1
CONSTANT		1			4					16
RSTD		1				7	9	9	7	7
DFT		1	4							
VALUES		1		7						
REL		1		9				9		
RANGE		1		9			9			

SPCVAL		1		7						
SNGVAL		1	21	7						
MIN					8					
MAX		2	2							10
ALWUNPRT										
ALWVAR		12								
PGM		1								
DTAARA		1								
FILE		1								
FULL		1								
EXPR		1	5							
VARY		3								
PASSATR		3								
PASSVAL		13						11		
CASE										
LISTDSPL										
CHOICE			14							
CHOICEPGM										
PMTCTL			15							
PMTCTLPGM			15							
PROMPT			6							

注：1、RTNVAL不能与下列参数一起用：CONSTANT, RSTD, DFT, VALUES, REL, RANGE, SPCVAL, SNGVAL, PGM, DTAARA，FILE, FULL, EXPR。它也不能用于REXX过程做命令处理程序的任何命令。

2、不允许 MAX 的值大于 1。

3、如果规定 PASSATR(*YES)和 RTNVAL(*YES)，必须也规定 VARY(*YES)。如果规定 RTNVAL(*YES)和 VARY(*YES)，必须规定*INT2 或 INT4。*INT2 和 *INT4 不能一起用。

4、CONSTANT 和 DFT 参数互相排斥。

5、EXPR(*YES)和 CONSTANT 互相排斥。

6、不允许 PROMPT。

7、如果规定 RSTD，要规定 VALUES，SPCVAL 和 SNGVAL 其一。

8、MIN 必须为零。

9、REL，RANGE，RSTD(*YES)互相排斥。

10、或者 MAX 大于 1，或者参数类型为语句标号，或两者都是。

11、参数不能引用 PASSVAL(*NULL)定义的参数，用 PASSVAL(*NULL)定义的 PARM 参数之间的范围无效。

12、如果规定 RTNVAL(*YES)，不能规定 ALWVAR(*NO)。

13、不允许 PASSVAL(*NULL)与 RTNVAL(*YES)一起规定，或规定 MIN 的值大于零。

14、CHOICE 和 CONSTANT 参数互相排斥。

15、CONSTANT 与 PMTCTL 和 PMTCTLPGM 互相排斥。

16、如果在 PARM 中定义了 SNGVAL，则不能在 ELEM/QUAL 中定义 CONSTANT

	MI	MA	ALWUNPRT	ALWVAR	PG	DTAARA	FIL	FULL	EXP	VAR
LEN										
RTNVAL		2		8	1	1	1	1	1	3
CONSTANT		2							4	
RSTD										
DFT	5									
VALUES										
REL										
RANGE										
SPCVAL										
SNGVAL		7								
MIN		6								
MAX	6									
ALWUNPRT										
ALWVAR										
PGM										
DTAARA					9		9			
FILE						9	9			
FULL										
EXPR										
VARY										
PASSATR										3
PASSVAL	10									
CASE										
LISTDSPL										
CHOICE										
CHOICEPGM										
PMTCTL	11									
PMTCTLPGM										
PROMPT										

注：1、RTNVAL不能与下列参数一起用：CONSTANT, RSTD, DFT, VALUES, REL, RANGE, SPCVAL, SNGVAL, PGM, DTAARA，FILE, FULL, EXPR。它也不能用于 REXX 过程做命令处理程序的任何命令

2、不允许 MAX 的值大于 1

3、如果规定 PASSATR(*YES)和 RTNVAL(*YES)，必须也规定 VARY(*YES)。如果规定 RTNVAL(*YES)和 VARY(*YES)，必须规定*INT2 或 INT4。*INT2 和 *INT4 不能一起用

4、EXPR(*YES)与 CONSTANT 互相排斥

5、MIN 必须为零

6、在 MIN 中规定的值不能超过 MAX 的值

7、或者 MAX 大于 1，或者参数类型为语句标号，或两者都是

8、如果规定 RTNVAL(*YES)，不能规定 ALWVAR(*NO)

9、PGM(*YES)，DTARA(*YES)与 FILE 不是*NO 互相排斥

10、 不允许 PASSVAL(*NULL)与 PTNVAL(*YES)一起规定，或规定 MIN 的值大于零 11、PMTCTL 不允许与 MIN 大于零的值一起用

	PASA	PASSV	CA	LISTD	DSPINP	CHO	CHOICE	PMTC	PMTCTLP	PRO
LEN										
RTNVAL	1	4								
CONSTA				9	5			7	7	2
RSTD										
DFT										
VALUES										
REL										
RANGE		3								
SPCVAL										
SNGVAL										
MIN		4						8		
MAX										
ALWUNP										
ALWVAR										
PGM										
DTAARA										
FILE										
FULL										
EXPR										
VARY	1									
PASSAT										
PASSVA										
CASE			10							
LISTDS				11						
CHOICE							6			
CHOICE	M					6				
PMTCTL										
PMTCTL	M									
PROMPT										

注：1、如果规定 PASSATR(*YES)和 RTNVAL(*YES)，必须也规定 VARY(*YES)。如果规定 RTNVAL(*YES)和 VARY(*YES)，必须规定*INT2 或 INT4。*INT2 和*INT4 不能一起用

2、不允许 PROMPT 参数

3、参数不能引用 PASSVAL(*NULL)定义的参数，用 PASSVAL(*NULL)定义的 PARM 参数之间的范围无效。

4、PASSVAL(*NULL)不允许与 RTNVAL(*YES)或 MIN 大于零的值一起使用
5、CHOICE 和 CONSTANT 互相排斥
6、CHOICE(*YES)需要 CHOICEPGM 的名字
7、CONSTANT 与 PMTCTL 和 PMTCTLPGM 互相排斥
8、PMTCTL 不允许与 MIN 大于零的一起用
9、CONSTANT 与 DSPINPUT(*NO)和 DSPINPUT(*PROMPT)互相排斥
10、CASE 参数仅在 PARM 和 ELEM 语句中有效，在 QUAL 语句中无效
11、LISTDSPL 仅在 PARM 语句中有效

9.4 定义参数列表

可以定义一个参数列表来接收一组值，而不用一个一个地接收单个值。可定义下列类型的列表：

简单列表：对一个参数规定同一类型的多个值
混合列表：对一个参数，分别定义一组值
列表中的列表：对一个参数规定多个列表，或对混合列表中的值规定一个列表。

下例命令的源码解释列表的不同类型：

```
            CMD         PROMPT('Example of lists command')

/* 下面的参数是一个简单列表，它接收5个名字。                          */
            PARM        KWD(SIMPLST) TYPE(*NAME) LEN(10) DFT(*ALL) +
                          SPCVAL((*ALL)) MAX(5) PROMPT('Simple list +
                          of up to 5 names')

/* 下面的参数是三个值的混合列表，每个有不同的类型和长度，每个元素    */
/* 不可以重复。                                                      */
            PARM        KWD(MXDLST) TYPE(MLSPEC) PROMPT('This is a +
                  mixed list of 3 val')
MLSPEC:     ELEM        TYPE(*CHAR) LEN(4) PROMPT('Elem 1 of 3')
            ELEM        TYPE(*DEC) LEN(3 0) PROMPT('Second of three')
            ELEM        TYPE(*CHAR) LEN(10) PROMPT('Last of three +
                          elements')


/* 下面的参数是列表中的列表，它包含两个元素的列表，它可以重复3次。   */
            PARM        KWD(LWITHINL1) TYPE(LWLSPECA) MAX(3) +
                          PROMPT('Repeatable list of 2 elements')
LWLSPECA:   ELEM          TYPE(*CHAR) LEN(10) PROMPT('1st part of +
                            repeatable list')
            ELEM          TYPE(*DEC) LEN(5 0) PROMPT('2nd part of +
                            repeatable list')


/* 下面的参数是列表中的列表，它包含两个元素的列表，其中一个可以重    */
```

```
/* 复3次。                                                          */
            PARM        KWD(LWITHINL2) TYPE(LWLSPECB) MAX(1) +
                          PROMPT('Repeated simple within mixed')
LWLSPECB:   ELEM        TYPE(*CHAR) LEN(10) MAX(3) PROMPT('Simple +
                                                  list within a list')
            ELEM        TYPE(*DEC) LEN(5 0) PROMPT('Single parm +
                          within a list')
```

下面的显示给出上例的提示：

```
                        Example of lists command (LSTEXAMPLE)

Type choices, press Enter.

Simple list of up to 5 names . . SIMPLST        *ALL
                         + for more values
This is a mixed list of 3 val    MXDLST
  Elem 1 of 3  . . . . . . . . .
  Second of three  . . . . . . .
  Last of three elements . . . .
Repeatable list of 2 elements    LWITHINL1
  1st part of repeatable list  .
  2nd part of repeatable list  .
                         + for more values
Repeatable simple within mixed   LWITHINL2
  Simple list within a list  . .
                         + for more values
  Single parm within a list  . .


                                                                  Bottom
F3=Exit   F4=List   F5=Refresh   F12=Cancel    F13=Prompter help
F24=More keys
```

9.4.1 定义简单列表

简单列表接收参数规定的一种类型的多个值。例如，参数为用户名，一个简单列表表示在这个参数中可以规定多个用户名。

```
USER(JONES SMITH MILLER)
```

如果参数值是一个简单列表，那么要用 PARM 中的 MAX 规定它能接收的最多元素数，对一个简单列表，只要规定 PARM 语句即可，不要其它的命令定义语句。

下例定义一个参数 USER，显示工作站用户可以规定 5 个用户名。

```
PARM        KWD(USER)  TYPE(*NAME)  LEN(10)  MIN(0)  MAX(5) +

            SPCVAL(*ALL) DFT(*ALL)
```

由 MIN(0)规定，此参数是一个可选参数，由 DFT(*ALL)规定缺省值为*ALL。

当简单列表中的元素传给命令处理程序时，根据使用的 CL、HLL 或 REXX 格式会有不同，下面二节内容介绍前例中的元素如何传送给 CL 和 HLL 及 REXX 的不同。

9.4.1.1 简单列表用于 CL 和 HLL

在用 CL 或 HLL 运行命令时，简单列表中的元素用下列格式传送给命令处理程序。

要传送值的个数	值	值	值	……

要传送值的个数是两字符的二进值，它指出实际传送多少个值，不是可以规定多少个值。由这种类型参数传送的值就是一个参数传送的值。例如，规定 USER 参数有两个用户名（BJONES 和 TBROWN），则传送下面的内容：

0002	BJONES	TBROWN

用户名做为 10 个字符的值传送，左对齐，后补足空格。

当传送简单列表时，仅传送在命令中规定元素数，紧接最后一个元素存放的不是列表的一部分也不能做列表的部分来引用。这样，在命令处理程序一个简单列表中，它用传来的元素个数来确定要处理多少个元素。

图 9-1 给出一个用二进内部函数来处理简单列表的 CL 过程：

```
PGM PARM (...&USER..)
.
.
.
/* 说明5个简单列表，可接收10个字符值。              */
DCL VAR(&USER) TYPE(*CHAR) LEN(52)
.
DCL VAR(&CT)    TYPE(*DEC)  LEN(3 0)
DCL VAR(&USER1) TYPE(*CHAR) LEN(10)
DCL VAR(&USER2) TYPE(*CHAR) LEN(10)
DCL VAR(&USER3) TYPE(*CHAR) LEN(10)

DCL VAR(&USER4) TYPE(*CHAR) LEN(10)
```

```
DCL VAR(&USER5) TYPE(*CHAR) LEN(10)
.
.
.
CHGVAR  VAR(&CT) VALUE(%BINARY(&USER 1 2))
.
IF (&CT > 0) THEN(CHGVAR &USER1 %SST(&USER 3 10))
IF (&CT > 1) THEN(CHGVAR &USER2 %SST(&USER 13 10))
IF (&CT > 2) THEN(CHGVAR &USER3 %SST(&USER 23 10))
IF (&CT > 3) THEN(CHGVAR &USER4 %SST(&USER 33 10))
IF (&CT > 4) THEN(CHGVAR &USER5 %SST(&USER 43 10))
IF (&CT > 5) THEN(DO)
/*  如果 CT 大于5，传送的值大于程序接收的值，则     */
/*  记录错误。                                      */
.
.
.
ENDDO
ELSE DO
/* 传送的值是正确的数，程序继续处理。        */
.
.
.
ENDDO
ENDPGM
```

图 9-1　简单列表的例子

可用同样技术处理 CL 过程或程序中的其它列表。

对一个简单列表，可在命令中进入一个象*ALL 或*NONE 这样的值。它做为单个值传送。同样，如果对一个参数没有规定值，则用缺省值做为列表值传送。例如，USER 参数用缺省值*ALL，则传送下面的内容：

0001	*ALL

*ALL 做为 10 字符值传送，左对齐，后补足空格。

如果对可选的简单列表参数没规定缺省值，则传送下列内容：

0000

9.4.1.2　简单列表用于 REXX

在运行同样命令时，在简单列表中的元素用下列格式的自变量串传给 REXX 过程：

```
...USER(值 1  值 2...值 n)...
```

此处值 n 是简单列表中的最后一个值。

例如，USER 中规定二个用户名 BJONES 和 TROWN，那么传送下列内容：

```
...USER(BJONES  TBROWN)...
```

在传送一个简单列表时，仅传送命令中规定的元素个数。因此，在命令处理程序处理简单列表时，用传送过来的元素个数来确定要处理多少个元素。图 9-2 中给出用 REXX 产生与图 9-1 中用 CL 或 HLL 同样结果的例子。

```
.
.
.
PARSE ARG . 'USER(' user ')' .
.
.
CT = WORDS(user)
IF CT > 0 THEN user1 = WORD(user,1) else user1 = ''
IF CT > 1 THEN user2 = WORD(user,2) else user2 = ''
IF CT > 2 THEN user3 = WORD(user,3) else user3 = ''
IF CT > 3 THEN user4 = WORD(user,4) else user4 = ''
IF CT > 4 THEN user5 = WORD(user,5) else user5 = ''
IF CT > 5 THEN
  DO
    /* 如果CT大于5，传送的值大于程序接收的值，则记录错误。*/
.
.
.
  END
ELSE
  DO
    /* 传送的值是正确的数，程序继续执行。          */
  END
EXIT
```

可在 REXX 程序中用上面的过程处理其它列表。

对一个简单列表，可在命令中进入一个象*ALL 或*NONE 这样的值。它做为单个值传送。同样，如果对一个参数没有规定值，则用缺省值做为列表值传送。例如，USER 参数用缺省值*ALL，则传送下面的内容。

```
...USER(*ALL)...
```

如果对可选的简单列表参数没有定义缺省值，则传送下列内容。

```
...USER( )...
```

有关 REXX 的详细内容，请看 REXX/400 程序员手册和参考。

9.4.2 定义混合列表

一个混合列表接收一组分别定义的值，实际上它们有不同的含义、不同类型且在表中有固定位置。例如 LOG(4 0 *SECLVL)就能规定一个混合列表。第一个值 4 指出要记录的信息级别，第二个值零指出要日志的最低信息级别，第三个值*SECLVL，规定要日志的信息总量。如果参数值是一个混合列表，列表中的元素必须用 ELEM 语句来分别定义。

与 PARM 一起的 TYPE 参数必须有标号，用它来引用列表中的第一个 ELEM 语句：

```
          PARM     KWD(LOG)     TYPE(LOGLST) ...

LOGLST:   ELEM     TYPE(*INT2)       ...
          ELEM     TYPE(*INT2)       ...
          ELEM     TYPE(*CHAR)       LEN(7)
```

仅第一个 ELEM 语句有标号，要用列表中元素的顺序来规定 ELEM 语句的顺序。

在 PARM 中的 MAX 大于 1，且 TYPE 参数引用 ELEM 语句时，被定义的参数是一个在列表中的列表。

可在 ELEM 语句中规定的参数有:

```
TYPE, LEN, CONSTANT, RSTD, DFT, VALUES, REL, RANGE, SPCVAL, SNGVAL,
MIN, MAX, ALWUNPRT, ALWVAR, PGM, DTAARA, FILE, FULL, EXPR, VARY
PASSATR, CHOICE, CHOICEPGM, PROMPT.
```

在下例中，定义了 CMPVAL 参数，这样，工作站用户规定一个比较值和一个比较的起始位置（混合列表）。

```
        PARM    KWD(CMPVAL) TYPE(CMP) SNGVAL(*ANY) DFT(*ANY) +
                  MIN(0)
CMP:    ELEM    TYPE(*CHAR) LEN(80) MIN(1)
        ELEM    TYPE(*DEC) LEN(2 0) RANGE(1 80) DFT(1)
```

在混合列表中的元素传送给命令处理程序时，格式会因命令处理程序是 CL，HLL 和 REXX 有所不同。下面的章节介绍这些内容。

9.4.2.1 混合列表用于 CL 和 HLL

在用 CL 或 HLL 运行一个命令时，在混合列表中的元素同下列格式传送给命令处理程序：

混合列表中值的个数	元素 1 的值	元素 2 的值	…	元素 n 的值

混合列表中值的个数用长度为 2 的十进制值传送。它指出在列表中定义了多少个值，而不是在命令中输入的值的个数。如果输入 SNGVAL 或传送缺省值，则这个值必须为 1，如

果用户没有输入一个元素的值，则传送缺省值，由它们的类型传送元素就象传送单个值一样。例如，在前例中用它输入 CMPVAL 的值为 QCMD1，但没输入串的起始位置值，它的缺省值为 1，则传送下列内容：

0002	QCMD1	1

数据 QCMD1 做为 80 个字符传送，左对齐后补足空格。元素个数是长度为 2 的二进值。

在用户输入一个值或缺省值为单值时，它做为列表中的第一个值传送。例如，用户输入 *ANY，则传送下列内容。

0001	*ANY

*ANY 做为 80 个长的字符传送，左对齐，后补足空格。

CL 程序可以处理混合列表，它不象简单列表，不用检查二进值来确定在表中有多少个值，因为这个值总与列表给的值相同，除非有 SNGVAL 传送给命令处理程序。此时，这个值为 1，如果对参数输入一个值，仅传送这一个值。在 CL 过程中处理混合列表，必须使用子串内部函数（见第二章）。

一种情况是，仅传送二进值 0000 做为列表的值的个数。如果在可选参数的 PARM 语句中没定义缺省值，且需要列表中的第一个值（MIN（1）），则不必要参数本身。而如果规定了任一个元素，则第一个元素是必须的。这时，如果没有规定参数值而输入命令，则传送下列内容：

0000

这种参数的例子如下：

```
        PARM  KWD(KWD1)     TYPE(E1) MIN(0)
E1:     ELEM  TYPE(*CHAR)   LEN(10)  MIN(1)
        ELEM  TYPE(*CHAR)   LEN(2)   MIN(0)
```

如果参数用 CL 过程处理，参数值接收到一个 14 位字符的 CL 变量中，头两个字符要与下列之一做比较：

用%SUB 函数与初值为十六进制 0000 的二位字符变量比较

用%BINARY 内部函数与十进零比较

标准的处理混合列表的方法是用 QUSRTOOL 中的 EXTLST 命令。

9.4.2.2　混合列表用于 REXX

在用 REXX 过程运行一个命令时，列表中的元素用下列格式传送给命令处理程序：

…CMPVAL(值 1　　值 2　　…值 n)…

其中值 n 是列表中最后一个元素。

如果用户没有输入元素值，则传送缺省值。在前面例子中，如果用户输入 CMPVAL 的值为 QCMD1，但没给出起始位置，它的缺省值为 1，则传送下列内容。

```
...CMPVAL(QCMD1       1)...
```

注意不传送结尾的空格。

当工作站用户输入一个单值或缺省为一个单值时，此值做为列表中的第一个元素传送。例如，用户输入*ANY 做为参数值时。传送下列内容：

```
...       CMPVAL(*ANY)...
```

注意，不传送结尾的空格。

如果对可选参数的 PARM 语句定义缺省值且列表的第一个值是必须的（MIN（10）），那么不必要参数本身，但如果规定了任何元素，则第一个元素是必须的，这时，如果没规定参数值而输入命令，则传送下列内容：

```
...       CMPVAL()...
```

9.4.3 定义列表中的列表

列表中的一个列表可以是：

对一个参数可以定义多次（简单或混合列表）

可在混合列表中规定列表做为一个值

下面是例子：

```
STMT((START RESPND) (ADDDSP CONFRM))
```

外边的括号括起对参数规定的列表（外层列表），而里面的括号括起列表中的列表（内层列表）

在下例中，在简单列表中定义混合列表，规定混合列表的 PARM 参数大于 1。这样，可以重复规定由 MAX 中规定的次数。

```
        PARM KWD(PARM1) TYPE(LIST1) MAX(5)
LIST1:  ELEM TYPE(*CHAR) LEN(10)
        ELEM TYPE(*DEC) LEN(3 0)
```

在下例中，二个元素可以规定五次，在输入这个参数的值时，可以出现下面的内容；

```
PARM1((VAL1 1.0) (VAR2 2.0) (VAR3 3.0))
```

在下例中，规定简单列表做混合列表的值，ELEM 语句中的 MAX 大于 1，元素可以重复在 MAX 中规定的次数。

```
        PARM KWD(PARM2) TYPE(LIST2)
LIST2:  ELEM TYPE(*CHAR) LEN(10) MAX(5)
        ELEM TYPE(*DEC) LEN(3 0)
```

此例中，第一个元素可以规定五次，而第二个元素仅可规定一次，在输入这些参数值时，

出现下面的内容：

```
PARM2((NAME1 NAME2 NAME3) 123.0)
```

在列表中的列表传送给处理程序时，格式会因 CL、HLL、REXX 而有所不同，下面各节介绍这些内容。

9.4.3.1　列表中的列表用于 CL 或 HLL。

在用 CL 或 HLL 运行命令时，列表中的列表，用下列格式传给处理程序：

列表数	列表 1 位移	列表 2 位移	…	列表 n 位移	参数数据	…

列表数做长度为 2 的二进制值传送，后边传送列表的位移（不是列表中输入的值）。根据 LISTDSPL 的规定，位移可以是长度为 2 或为 4 的二进值传送。

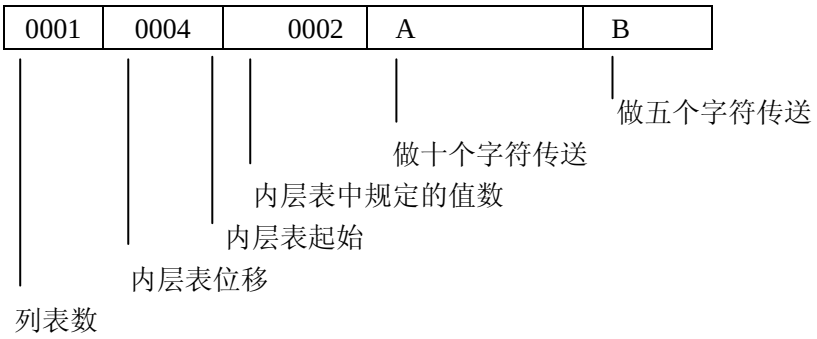
下例给出参数 KWD2 的定义（它是一个简单列表中的混合列表），由用户规定参数和传送什么，参数定义是：

```
        PARM    KWD(KWD2)     TYPE(LIST) MAX(20) MIN(0) +
                DFT(*NONE)   SNGVAL(*NONE)   LISTDSPL(*INT2)
LIST:   ELEM    TYPE(*CHAR)   LEN(10) MIN(1)        /*From value*/
        ELEM    TYPE(*CHAR)   LEN(5) MIN(0)         /*To value*/
```

用户输入的 KWD2 参数为：

```
KWD2((A   B))
```

下列内容传送给处理程序：



列表中的列表顺序从 n（工作站用户输入的最后一项）到 1（用户输入的第一项）传送给处理程序，但位移是从 1-n。

下面是列表中的列表的更复杂的例子，参数定义如下：

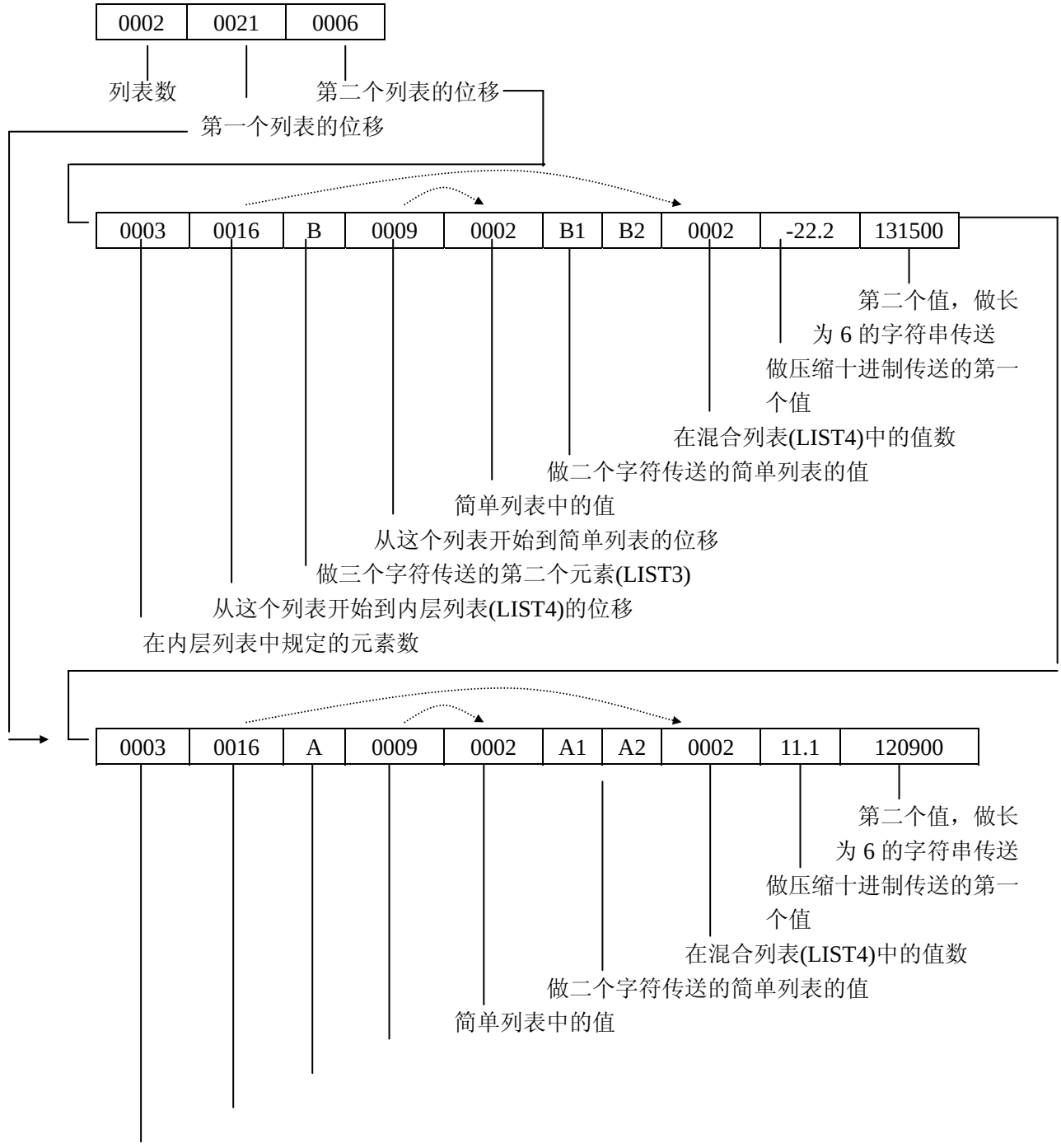
```
         PARM    KWD(PARM1)   TYPE(LIST3)   MAX(25)
LIST3:   ELEM    TYPE(LIST4)
         ELEM    TYPE(*CHAR)   LEN(3)
         ELEM    TYPE(*NAME)   LEN(2)   MAX(5)
LIST4:   ELEM    TYPE(*DEC) LEN(7 2)
```

ELEM TYPE(*TIME)

如果用户给 PARM1 参数输入的内容为：

PARM1(((11.1 120900) A (A1 A2)) ((-22.2 131500) B (B1 B2)))

那么下面内容传送给处理程序：



从这个列表开始到简单列表的位移
做三个字符传送的第二个元素(LIST3)
从这个列表开始到内层列表(LIST4)的位移
在内层列表中规定的元素数

9.4.3.2 列表中的列表用于 REXX

在用 REXX 运行命令时，把列表中的列表用此参数输入值的格式传给命令处理程序，不传送未尾的空格。

下例给出 KWD2 的定义，它是在简单列表中的混合列表。参数及其值由用户规定，参数的定义为：

```
        PARM    KWD(KWD2)     TYPE(LIST) MAX(20) MIN(0) +
                DFT(*NONE)   SNGVAL(*NONE)
LIST:   ELEM    TYPE(*CHAR)   LEN(10) MIN(1)        /*From value*/
        ELEM    TYPE(*CHAR)   LEN(5) MIN(0)         /*To value*/
```

用户输入的 KWD2 参数为：　KWD2((A　B))
传给处理程序的为：　KWD2(A　B)
如果用户输入下列内容：　KWD2((A　B) (C　D))
传给处理程序的是：　KWD2((A　B) (C　D))
下面是列表中列表更复杂的例子，参数定义为：

```
         PARM    KWD(PARM1)   TYPE(LIST3)   MAX(25)
LIST3:   ELEM    TYPE(LIST4)
         ELEM    TYPE(*CHAR)   LEN(3)
         ELEM    TYPE(*NAME)   LEN(2)   MAX(5)
LIST4:   ELEM    TYPE(*DEC) LEN(7 2)
         ELEM    TYPE(*TIME)
```

用户为 PARM1 输入的内容为：

```
PARM1(((11.1  12D900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

下列内容传给处理程序：

```
PARM1(((11.1  12D900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

9.4.4 定义一个限定名

限定名由目标名和它所在的库名组成。如果参数值或列表项是一个限定名，必须用 QUAL 语句来分别定义这些名字，限定名的每部分都要用 QUAL 语句。限定名的各部分必须用它们在限定名出现的顺序来说明。在第一个 QUAL 语句中要规定*NAME 或*GENERIC 与之有关的 PARM 和 ELEM 语句必须规定标号，用它来引用限定名的第一个 QUAL 语句。

下面的命令定义语句定义最常用的限定名，QUAL 语句必须用它们在限定名中出现的顺

序来出现。

```
                PARM      KWD(NAME) TYPE(NAME1) SNGVAL(*NONE). . .
                                    |
  ┌─────────────────────────────────┘
  │
  └NAME1:  QUAL      TYPE(*NAME)
           QUAL      TYPE(*NAME)
```

能对 QUAL 语句定义的参数个数与 PARM 语句说明的数目相同，但在 TYPE 参数中仅可规定下列值：

*NAME
*GENERIC
*CHAR
*INT2
*INT4

在限定名传送给命令处理程序时，根据使用的 CL、HLL、REXX 而各不相同，下面继续介绍这方面的内容。

9.4.4.1 限定名用于 CL 或 HLL

在用 CL 或 HLL 时，限定名用下面的格式传送给命令处理程序：

限定 1 的值	限定 2 的值

例如，用户为以前定义的 QUAL 输入 NAME(USER/A)的值，名字用下列格式传给命令处理程序：

A	USER
10 个字节	10 个字节

限定名由它们的类型和长度连续地传给命令处理程序就象单个的参数值一样传送(请看 9.2.2)，分隔符（/）不传送。它不考虑传送的是单个参数、混合列表的元素或限定名的简单列表。

如果用户输 入限定名的单个值，传送值的长度是限定名各部的总和。例如，定义了每个长为 10 个字符的二个值的限定名，用户输入了一个单值，它被左对齐传送，在右边增加空格使总数为 20 来传送。如果用户用*NONE 传送，则传送下列 20 个字符：

*NONE

下例是在 CL 程序中用子串功能处理限定名，它的功能是把限定名分成二个值：

```
PGM PARM(&QLFDNAM)
DCL &QLFDNAM TYPE(*CHAR) LEN(20)
DCL &OBJ TYPE(*CHAR) LEN(10)
DCL &LIB TYPE(*CHAR) LEN(10)
```

```
CHGVAR &OBJ %SUBSTRING(&QLFDNAM 1 10) /* First 10 */
CHGVAR &LIB %SST(&QLFDNAM 11 10) /* Second 10 */
 .
 .
 .
ENDPGM
```

可用适当的 CL 语法来规定限定名，例如 OBJ(&LIB/&OBJ)。也可用下列方法把限定名分成二个值：

```
PGM PARM(&QLFDNAM)
DCL &QLFDNAM TYPE(*CHAR) LEN(20)
CHKOBJ (%SST(&QLFDNAM 11 10)/%SST(&QLFDNAM 1 10)) *PGM
 .
 .
 .
ENDPGM
```

限定名的简单列表用下列格式传送给命令处理程序：

限定名个数	值 1 限定 1	值 1 限定 2	值 2 限定 1	值 2 限定 2	……

例如，NAMEPARM 参数的中规定 MAX（3）：

```
        PARM    KWD(NAME) TYPE(NAME1) SNGVAL(*NONE) MAX(3)
NAME1:  QUAL    TYPE(*NAME)
        QUAL    TYPE(*NAME)
```

如果用户输入 NAME(QGPL/A USER/B)
那么名字参数用下列格式传给命令处理程序：

0002	A	QGPL	B	USER

如果用户输入 NAME(*NONE),那么名字参数用下列格式传送：

0001	*NONE

9.4.4.2 限定名用于 REXX

在用 REXX 运行一个命令时，限定名用对此参数输入的值传给命令处理程序，不传送结尾的空格。

例如，用户对上节定义的 QUAL 语句输入下列内容：

MANE(USER/A)

限定名用下列格式传给命令处理程序：NAME(USER/A)

限定名用它的类型和长度就象一个单个参数值一样连续的传给命令处理程序。

如果用户输入*NONE，则传送下列 20 个字符：

NAME(*NONE)

下例给出用户输限定名的简单列表

NAME(QGPL/A　USER/B)

9.4.5　定义依赖关系

如果在参数之间有要需求关系且在命令运行时要检查参数的值，要用 DEP 语句定义这些关系。用 DEP 可以做：

规定一些控制条件，它在 PARM 参数需要为真定义的参数关系前要为真（CTL）

如果由 CTL 定义的控制条件为真，规定必须要测试的参数关系（PARM）

如果控制条件为真，在与之相关的 PARM 语句中规定定义的参数关系数

规定在信息文件中错误信息的信息标识，如果参数依赖关系不适合的话，这个错误信息要发送给显示工作站用户。

在下面的例子中，如果用户规定 TYPE(LIST)，那也必须规定 ELEMLIST 参数。

```
DEP CTL(&TYPE *EQ LIST) PARM(ELEMLIST)
```

在下例中，参数&WRITER 必须等于&NEWWTR。如果条件不为真，则给用户发送 USR0001 信息。

```
DEP CTL(*ALWAYS) PARM((&WRITER *NE &NEWWTR)) MSGID(USR0001)
```

在下例中，如果用户规定 FILE 参数，也必须规定 VOL 和 LABEL 参数。

```
DEP CTL(FILE) PARM(VOL LABEL) NBRTRUE(*EQ 2)
```

9.4.6　可能的选择和值

在提示显示中的参数输入字段的右边会显示可选的选择，也能自动生成在命令定义源文件中规定的提示说明或由出口程序动态生成，描述可能选择的说明也能在 PARM、ELEM 或 QUAL 语句中定义。但由于显示格式的限制，显示说明的长度只能是 12 或更少。在组中除第一个限定名个，其余都是长度为 10 或更少。

可选的说明是用 CHOICE 参数定义的，这个参数的缺省为*VALUES。它指出说明是由 TYPE、RANGE、VALUES、SPCVAL 和 SNGVAL 键字规定的值自动生成的。说明限制在 30 个字符。如果有更多的值，那么会在说明末尾加上删节号（……）表示说明是不完整的。也可规定没有可能的选择来显示（*NONE），或规定要显示的说明串或说明信息的标识，它是从 CRTCMD 命令中的 PMTFILE 参数中规定的信息文件中取得的。

也可规定一个在提示期间运行的出口程序来提供可能选择的说明。例如想给用户看当前存在系统中的目标，就可以这样做。也可用这样的出口程序规定在参数值的显示中出现的列表。要规定出口程序，对 CHOICE 参数规定*PGM，在 PARM、ELEM 或 QUAL 中的 CHOICEPGM 中规定出口程序的限定名。

出口程序要接收下面二个参数：

参数 1：21 字节的字段，它由提示传给选择程序，它包括下列内容：

1—10 位：命令名，规定引起程序运行的命令名

11—20 位：键字名，规定可能选择的键字或需要的可能值

21 位：C 或 P，指出提示的数据类型，C 为 30 个字节的字段，存放返回的可能选择的说明。P 为 2000 字节字段，存放返回的可能值

参数 2：30 或 2000 字节的字段，用于返回下列之一：

— 如果第一个参数的 21 字节为 C，它指出要返回可能的选择，另外这是一个 30 字节的字段，程序要把这个说明放在提示显示中的输入字段右边。

— 如果第一个参数的 21 字节为 P，这是一个 2000 字节的字段，程序要在这里放列表，列表的头两个字节必须包括列表中的项数（用二进制）。这个值后放二字节二进制长度，其后为值，它必须是 1—32 字符长。

如果头两个字节返回的是二进制零值，那么不显示许可值。

如果头两个字节返回的是二进制负值，从命令中取许可值。

如果在调用程序时发生异常，那么可选择的说明变成空格，许可值列表从命令中取得。

9.5 使用提示控制

可用提示控制规则来控制在提示期间显示命令的哪个参数，它能够简化命令的提示，仅显示用户要看的那些参数。

也能规定根据其它参数的值来显示的参数。这在一个参数在仅另外参数（叫做控制参数）有特定值时才有意义的时候特别有用。也可规定仅在提示期间按一个功能键来请求附加参数时来选择参数做提示，可用这种方法来做很少用户规定的参数。或者因为通常使用缺省值，或者由于它们控制极少使用功能。要想看一个命令中有提示控制的所有参数，可在提示期间用 F9 键来显示所有参数。

9.5.1 条件提示

在对用户提示命令时，如果有下列条件，一个参数由其它参数做条件控制：

由对控制参数规定的值选择

对被控制参数规定的值有错

对条件参数规定了一个值

在提示请求显示所有参数时，按一个功能键

在用条件参数提示用户且对它的控制参数没有规定任何值时，则显示以前选择的所有参数，在用户按执行键时，检查控制参数来确定是否显示条件参数。

要在命令定义源文件中规定条件提示，对每个由其它参数控制的参数的 PARM 语句里用 PMTCTL 参数规定每个标号名，规定的标号必须在规定控制参数的 PMTCTL 语句中定义，并要检查条件来选择要提示的参数，多个 PARM 语句可以引用同一个标号。

在 PMTCTL 语句中，规定控制参数的名字、要检测的一个或多个条件以及对选择提示的条件参数必须为真的条件数。假如控制参数有特别 映象，在 PMTCTL 语句中所用的值必须是 TO_值。如果控制参数是一个列表或限定名，仅比较第一个列表项或限定名。

在下例中，如 OUTPUT 为*OUTPUT，则选择 OUTPUT 和 OUTMBR 参数。如果 OUTPUT 为*PRINT，则仅选择 OUTQ 参数。

```
PARM OUTPUT TYPE(*CHAR) LEN(1) DFT(*) RSTD(*YES) +
       SPCVAL((*) (*PRINT P) (*OUTFILE F))
PARM OUTFILE TYPE(Q1) PMTCTL(OUTFILE)
PARM OUTMBR TYPE(*NAME) LEN(10) PMTCTL(OUTFILE)
```

```
      PARM OUTLINK TYPE(*CHAR) LEN(10)
      PARM OUTQ TYPE(Q1) PMTCTL(PRINT)
      Q1: QUAL TYPE(*NAME) LEN(10)
          QUAL TYPE(*NAME) LEN(10) SPCVAL(*LIBL) DFT(*LIBL)
OUTFILE: PMTCTL CTL(OUTPUT) COND((*EQ F)) NBRTRUE(*EQ 1)
PRINT:   PMTCTL CTL(OUTPUT) COND((*EQ P)) NBRTRUE(*EQ 1)


PARM P1 TYPE(*CHAR) LEN(5) RSTD(*YES) VALUES(*ALL *SOME *NONE)
PARM P2 TYPE(*NAME) LEN(10) SPCVAL(*ALL)
PARM P3 TYPE(*CHAR) LEN(10) PMTCTL(PMTCTL1)
PMTCTL1:PMTCTL CTL(P1) COND((*EQ *ALL))
        PMTCTL CTL(P1) COND((*EQ *SOME)) LGLREL(*OR)
        PMTCTL CTL(P2) COND((*EQ *ALL)) LGLREL(*AND)
        PMTCTL CTL(P1) COND((*EQ *NONE)) LGLREL(*OR)
        PMTCTL CTL(P2) COND((*NE *ALL)) LGLREL(*AND)
```

在上面例子中，在检测 OUTMBR 参数后，对用户提示 OUTLINK 参数，在某些时候，检测到一个错误，在 OUTMBR 返回一个单个星号值之前，提示用户 OUTLINK 参数，在编码 PMTCTL 参数时要考虑这点。在发生错误时，由那个参数做条件显示所有的参数。

写的出口程序必须接收三个参数：

一个 20 个字符的字段。提示把命令名传给前 10 个字符，把控制参数的名字传给后 10 个字符，这个字段不能修改。

控制参数的值。这个字段用传给命令处理程序的同一格式，不能被修改。如果控制参数的值定义为 VARY(*YES)，则此值前没有长度值，如果控制参数 PASSATR(*YES)，则不包括属性字节。

一个 32 字符字段。出口程序把在 PMTCTL 语句中要检测的值放在这里，这个要检测的值必须用在数据类型说明中的相同格式返回。

在下例中，OBJ 是一个限定名，它可以是命令名，程序名或文件名，出口程序决定目标类型并把类型返回到变量&RTNVAL 中。

```
CMD
PARM OBJ TYPE(Q1) PMTCTLPGM(CNVTYPE)
Q1: QUAL TYPE(*NAME) LEN(10)
    QUAL TYPE(*NAME) LEN(10) SPCVAL(*LIBL) DFT(*LIBL)
PARM CMDPARM TYPE(*CHAR) LEN(10) PMTCTL(CMD)
PARM PGMPARM TYPE(*CHAR) LEN(10) PMTCTL(PGM)
PARM FILEPARM TYPE(*CHAR) LEN(10) PMTCTL(FILE)
CMD: PMTCTL CTL(OBJ) COND((*EQ *CMD) (*EQ *)) NBRTRUE(*EQ 1)
PGM: PMTCTL CTL(OBJ) COND((*EQ *PGM) (*EQ *)) NBRTRUE(*EQ 1)
FILE: PMTCTL CTL(OBJ) COND((*EQ *FILE) (*EQ *)) NBRTRUE(*EQ 1)
```

出口程序的源码为：

```
PGM PARM(&CMD &PARMVAL &RTNVAL)
DCL &CMD *CHAR 20                 /* 命令和参数名                    */
DCL &PARMVAL *CHAR 20             /* 参数值                          */
DCL &RTNVAL *CHAR 32              /* 返回值                          */
DCL &OBJNAM *CHAR 10              /* 目标名                          */
DCL &OBJLIB *CHAR 10              /* 目标类型                        */
CHGVAR &OBJNAM %SST(&PARMVAL 1 10)
CHGVAR &OBJLIB %SST(&PARMVAL 11 10)
CHGVAR &RTNVAL '*'                /* 返回的错误初值                   */
CHKOBJ &OBJLIB/&OBJNAM *CMD       /* 看命令是否存在                   */
MONMSG CPF9801 EXEC(GOTO NOTCMD) /* 如果没命令，跳过                 */
CHGVAR &RTNVAL '*CMD'             /* 指出目标是一个命令               */
RETURN                            /* 结束                            */
NOTCMD:
CHKOBJ &OBJLIB/&OBJNAM *PGM       /* 看程序是否存在                   */
MONMSG CPF9801 EXEC(GOTO NOTPGM) /* 如果不是程序，跳过               */
CHGVAR &RTNVAL '*PGM'             /* 指出目标是一个程序               */
RETURN                            /* 结束                            */
NOTPGM:
CHKOBJ &OBJLIB/&OBJNAM *FILE      /* 看文件是否存在                   */
MONMSG CPF9801 EXEC(RETURN)       /* 如果不是文件，结束               */
CHGVAR &RTNVAL '*FILE'            /* 指出目标是一个文件               */
ENDPGM
```

9.5.2 附加参数

可以规定一个参数，它不是经常使用，就不用提示，除非在提示期间用户按一个功能键来请求附加参数时才做提示。这可用参数 PARM 语句中的 PMTCTL(*PMTRQS)来做。在提示一个命令时，有此规定的参数不做提示，除非为它们规定一个值或用户按 F10 键来请求附加参数。

在提示显示中会在附加参数前有分隔行，以便与其它参数区别开来。一般说，有 PMTCTL(*PMTRQS)规定的参数都要最后提示，也可在用 PROMPT 键字来规定一个关系提示号来替代它们。如果这样做，那么在按 F10 键时就很难看出哪些参数是后加到提示中来的。

9.6 使用键字参数和提示替代程序

提示替代程序允许在提示一个命令时用当前值而不是用缺省值来显示。如果为一个命令定义提示替代程序，可用下面二种方法看到调用提示替代程序的结果：

在命令行写出命令名，不带任何参数，按 F4 键，下一屏给出命令的键字参数。键字参数是唯一标识此目标的参数，填好给出的所有字段，按执行键，下一屏给出命令的所有参数，以及包括当前值而非缺省值的非键字参数字段的参数字段。

例如，在命令行写 CHGLIB 按 F4 键，仅能看到 LIB 参数。如果填上*CURLIB 按执行键，显示当前库的值。

在命令行写命令名和所有键字参数的值按 F4 键，下一屏给出命令的所有参数，及包括当前值而不是缺省值（例如*SAME 和*PRV）的非键字参数字段的参数字段。

例如，在命令行写 CHGLIB　LIB(*CURLIB)，按 F4 键，则显示当前库的值。

在按 F10 键（附加参数）时，则用当前值显示定义为 PMTCTL(*PMTRQS)的任何参数，详细内容请看 9.5.2。

要结束命令提示，用 F3 键。

9.6.1　使用提示替代程序的过程

要用提示替代程序，做下列操作：

1、在命令定义源文件的 PARM 语句中规定做键字参数的参数。对 KEYPARM 参数的信息，请看 9.6.1.1 。

2、写一个提示替代程序，详细信息请看 9.6.1.2。

3、在生成或修改命令时的 PMTOVRPGM 参数中规定提示替代程序的名字，详细内容请看 9.6.1.3。

9.6.1.1　规定键字参数

键字参数的数目受限于需要在目标修改时唯一定义的参数个数。

为了保证在命令定义源文件中正确的规定键字参数，要按以下要求做：

在命令定义源文件中规定 KEYPARM(*YES)

在规定 KEYPARM(*NO)之前定义有 KEYPARM(*YES)的所有参数。

注：如果在规定 KEYPARM(*NO)之后定义有 KEYPARM(*YES)的参数，则此参数不看作一个键字参数并发出警告信息。

在 PARM 语句中不要规定 MAX 大于 1 的值。

在与键字参数有关的 ELEM 语句中不要规定 MAX 值大于 1 的值。

在 PARM 语句中不要规定*PMTRQS 或提示控制语句。

在命令定义源语句中键字参数的位置要与提示它们时出现的位置一致。

9.6.1.2　写一个提示替代程序

在提示命令时，命令替代程序要把传送一定的信息做返回的当前值。在写提示替代程序时必须考虑传送的信息和返回值。例子请看 9.6.2。

传送给提示替代程序的参数：

一个 20 个字符的字段。头十个字符是命令名，后十个字符地是命令所在的库名。

每个键字的一个值。如果定义了多个键字值，要传送的参数值按在命令定义源文件中键字参数定义的顺序传送。

一个 5700 字节的空间。用来放由提示替代程序生成的命令串，头两个字节为返回的命令串的十六进制长度，后跟实际的命令串。

例如，对一个命令定义两个键字参数，有四个参数要传给提示替代程序：

— 一个参数用做命令

— 两个参数用做键字参数

— 一个参数用做命令串空间

从命令替代程序返回的信息：

根据传送的值，提示替代程序取得不是键字参数的参数当前值，这些值放到一个命令串中，在这里确定和返回串的长度。

用下列方法保证正确的定义命令串：

用在命令行中使用命令串的键字格式

在命令串中不包括命令名和键字参数

在每个键字前用一个选择提示符来定义如何显示参数及传给命令处理程序的值，使用选择提示符的内容，请看 6.5.2。

在使用选择提示时，做下列操作：

— 如果在命令定义源语句中定义参数为 MIN(1)（即此参数是必须的），那么来自提示替代程序命令串中键字的选择提示符必须是??。

— 在提示替代程序命令串中的选择提示字符不能用?－，

下面例子给出从提示替代程序返回的命令串：

```
??Number(123456) ?<Qualifier(CLIB/CFILE) ?<LIST(ITEM1 ITEM2 ITEM3) ?<TEXT
                                                        ('Carol's file')
```

要保证程序传送的空间的头二个字节是命令串的实际十六进制长度。

在命令串中只包括在提示命令时要显示当前值的那些参数。没包括在命令串中的参数用缺省值显示。

出现在命令串的数字用其字符格式，不要用十进制和压缩格式。在命令串中不要有十六进制撽字。

在库和限定符之间及限定符和目标之间要没有空格。例如，

```
??KWD1(library /object)    无效

??KWD1(library/ object)    无效

??KWD1(library/object)     有效

??KWD1( library/object)    有效
```

如果用特殊值或一个单值，要保证把它们传送给在命令定义源文件中定义的 FROM_值中。

例如，在命令定义源文件中有一个键字定义为 SPCVAL(*SPECIAL)。*SPECIAL 是一个 FROM_值。*是 TO_值，在取得这个键字的当前值时，*是获得的值，而*SPECIAL 必须出现在从提示替代程序返回的命令串中，正确的 FROM_值必须放在命令串中。这是基于特殊值或单值可能有相同的 TO_值。例如，规定了 KWD1 SPCVAL((*SPC *)(*SPECIAL *))，提示替代程序必须确定*是*SPC 的 TO_值还是*SPECIAL 的 TO_值。

如下定义用于取得说明的长度字段：

（2*（在命令定义源文件中定义的字段长度））+2

这个长度允许说明字段允许的引号的最大数。例如，在命令定义源文件中定义的CHGxxx 中 TEXT 参数的长度为 LEN(50)，那么在提示替代程序中此参数要说明为 CHAR（120）。这方面的详细说明，请看 9.6.2。

如果在提示替代程序中没有正确的定义说明字段，以及由提示替代程序获得的说明串有引号，那么不能正确地做提示。

要保证嵌入的撇号是成对的。例如：

```
?<TEXT('Carol's library')
```

某些命令仅能在某种方式下（例如 DEBUG）或某种作业状态下（例如*BATCH）运行，但可在某它方法或状态下提示。在提示命令时，调用提示替代程序而不管用户环境。如果在此命令无效的环境或方式下调用提示替代程序，则显示命令的缺省值且返回的长度为零，在非调试环境下使用 CHGDBG 和 ADDPGM 命令就是这种情况。

在提示替代程序中允许的错误：如果提示替代程序检测到错误，它做如下事情：

把命令串的长度设为零，这样在提示命令时就显示缺省值而不是当前值。

给先前的调用发送诊断信息。

发送 CPF0011 逃逸信息。

例如，如果需要一个库不存在的信息，要加一个类似 下面的信息描述：

```
ADDMSGD      MSG('Library &2 does not exist') +
             MSGID(USR0012) +
             MSGF(QGPL/ACTMSG) +
             SEV(40) +
             FMT((*CHAR 4) (*CHAR 10))
```

注：替换变量&1 不在信息中而在 FMT 参数中做 4 字符定义，&1 保留由系统使用，必须也是 4 个字符。如果&1 仅是在信息中定义的替换变量，必须在发送信息时保证第四个字节的信息数据且不包括空格，第四个字节由系统在命令处理和提示期间用来管理信息。

在提示替代程序中规定下列内容可把上述信息发送给调用提示替代的程序。

```
SNDPGMMSG    MSGID(USR0012) MSGF(QGPL/ACTMSG) +
             MSGDTA('0000' || &libname) MSGTYPE(*DIAG)
```

在提示替代程序发送所有必要的诊断信息后，然后发送 CPF0011。要发送这个信息。用下列命令：

```
SNDPGMMSG    MSGID(CPF0011) MSGF(QCPFMSG) +
             MSGTYPE(*ESCAPE)
```

在接收到 CPF0011 时，发送 CPD680A 信息给调用程序且在提示屏上指出已经找到错误，所有诊断信息放在用户作业日志中。

9.6.1.3　规定生成或修改命令的提示替代程序

要使用一个要生成命令的提示替代程序，在 CRTCMD 命令中规定此程序名，也可用

CHGCMD 命令在修改命令时来规定程序名，对这两个命令要在 PMTOVRPGM 参数中规定提示替代程序的名字。

如果在命令定义源文件中定义了键字参数,但在命令生成或修改时没有规定提示替代程序，则会产生 CPD029B 警告信息，它用命令定义源文件中规定的缺省来显示。

有时在命令生成时规定了提示替代程序，但在命令定义源文件中没定义参数，这种情况下，在提示命令前调用提示替代程序，在命令生成或修改时会发送 CPD029A 信息。

9.6.2 使用提示替代程序的例子

下例给出命令及其提示替代程序的命令源码，这个命令允许修改一个库的主人和说明，这个命令的提示替代程序接收库名，取得库主人及说明的当前值，然后把这些值放到一个命令串中且返回它，这个提示替代程序用“？”做选择提示符。

9.6.2.1 命令源码的例子

```
CHGLIBATR: CMD   PROMPT('Change Library Attributes')
           PARM  KWD(LIB) +
                   TYPE(*CHAR) MIN(1) MAX(1) LEN(10) +
                   KEYPARM(*YES) +
                   PROMPT('Library to be changed')
           PARM  KWD(OWNER) +
                   TYPE(*CHAR) LEN(10) MIN(0) MAX(1) +
                   KEYPARM(*NO) +
                   PROMPT('Library owner')
           PARM  KWD(TEXT) +
                   TYPE(*CHAR) MIN(0) MAX(1) LEN(50) +
                   KEYPARM(*NO) +
                   PROMPT('Text description')
```

9.6.2.2 提示替代程序的例子

```
 PGM PARM(&cmdname &keyparm1 &rtnstring)
/*************************************************************************/
/*                                                                       */
/*  定义传给提示替换程序的参数                                           */
/*                                                                       */
/*************************************************************************/
 DCL  VAR(&cmdname)    TYPE(*CHAR) LEN(20)
 DCL  VAR(&keyparm1)    TYPE(*CHAR) LEN(10)
 DCL  VAR(&rtnstring) TYPE(*CHAR) LEN(5700)


/*************************************************************************/
/*                                                                       */
/* 返回命令串结构说明                                                    */
```

```
/*                                                                   */
/*******************************************************************/

                     /* Length of command string generated          */
 DCL  VAR(&stringlen) TYPE(*DEC)  LEN(5 0) VALUE(131)
 DCL  VAR(&binlen)    TYPE(*CHAR) LEN(2)
                     /* OWNER keyword                               */
 DCL  VAR(&ownerkwd)  TYPE(*CHAR) LEN(8)  VALUE('?<OWNER(')
 DCL  VAR(&name)      TYPE(*CHAR) LEN(10)
                     /* TEXT keyword                                */
 DCL  VAR(&textkwd)   TYPE(*CHAR) LEN(8)  VALUE(' ?<TEXT(')
 DCL  VAR(&descript)  TYPE(*CHAR) LEN(102)


/*******************************************************************/
/*                                                                   */
/* 与命令串有关的变量说明                                            */
/*                                                                   */
/*******************************************************************/
 DCL  VAR(&quote)      TYPE(*CHAR) LEN(1) VALUE('')
 DCL  VAR(&closparen)  TYPE(*CHAR) LEN(1) VALUE(')')


/*******************************************************************/
/*                                                                   */
/*                    启动可操作码                                   */
/*                                                                   */
/*******************************************************************/
/*******************************************************************/
/*                                                                   */
/* 监控例外                                                          */
/*                                                                   */
/*******************************************************************/
    MONMSG  MSGID(CPF0000) +
            EXEC(GOTO CMDLBL(error))


/*******************************************************************/
/*                                                                   */
/* 取得LIB参数规定的库的主人和说明。注：此程序假定在TEXT说明中无引号*/
/*                                                                   */
/*******************************************************************/
    RTVOBJD OBJ(&keyparm1) OBJTYPE(*LIB) OWNER(&name) TEXT(&descript)
```

```
    CHGVAR VAR(%BIN(&binlen)) VALUE(&stringlen)


/*****************************************************************/
/*                                                               */
/* 见命令串                                                      */
/*                                                               */
/*****************************************************************/
    CHGVAR VAR(&rtnstring) VALUE(&binlen)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &ownerkwd)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &name)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &closparen)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &textkwd)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &quote)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &descript)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &quote)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &closparen)

    GOTO CMDLBL(pgmend)

    CHGVAR VAR(&stringlen) VALUE(0)
    CHGVAR VAR(%BIN(&binlen)) VALUE(&stringlen)
    CHGVAR VAR(&rtnstring) VALUE(&binlen)


/*****************************************************************/
/*                                                               */
/* 发送错误命令                                                  */
/*                                                               */
/* 注：如果要发送诊断信息以及CPF0011，则必须在下面列出的第一个   */
/*     SNDPGMMSG命令中的MSGF参数输入一个有效的错误信息ID。如果   */
/*     不需要发送一个诊断信息，在程序中不要用第一个SNDPGMMSG，   */
/*     但在错误条件下，必须总发送CPF0011，这样，在程序中必须有  */
/*     第二个SNDPGMMSG命令。                                     */
/*                                                               */
/*****************************************************************/
    SNDPGMMSG MSGID(XXXXXXX) MSGF(MSGLIB/MSGFILE) MSGTYPE(*DIAG)
    SNDPGMMSG MSGID(CPF0011) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)

PGMEND:
ENDPGM
```

9.7 生成命令

在用命令定义语句定义了命令之后，要用 CRTCMD 命令来生成它，除了规定命令名、

库名、命令处理程序名（CL 或 HLL），源成员名、源文件名、命令环境、REXX 的出口程序，还要定义下列命令属性：

命令使用的有效性检查

命令运行方式：

- 产品
- 调试
- 服务

命令使用环境：

- 批作业
- 交互作业
- 批作业中的 ILE CL 模块
- 批作业中 CL 程序
- 交互作业中的 ILE CL 模块
- 交互作业中的 CL 程序
- 批作业中的 REXX 过程
- 交互作业中的 REXX 过程
- 由系统调用 QCMDEXC 解释处理的一个命令（有关 QCMDEXC 的内容，请看第六章）

能规定的位置参数的最大数目

包括提示正文的信息文件

用做提示参数帮助的帮助面板组

命令所用的一般帮助模块的帮助标识名

包括在 DEP 语句中标识信息的信息文件

命令处理时活动的当前库

命令处理时活动的产品库

如果规定了 REPLACE(*YES)，是否用相同名、类型和库来结束命令

命令的公共授权和它的说明

简短介绍命令及其功能的说明

对 REXX CPPS 的命令，还要规定下列内容：

在过程启动时处理命令的初始环境

控制过程运行的出口程序

下例定义一个名为 ORDENTRY 的命令，它调用一个定货单应用程序，用 CRTCMD 定义它的属性且用 QCMDSRC 中的成员 ORDENTRY 里的参数定义生成命令，ORDENTRY 中有 9.2.2.5 例中的 PARM 语句。

```
CRTCMD      CMD(DSTPRODLB/ORDENTRY) +
            PGM(*LIBL/ORDENT) +
            TEXT('Calls order entry application')
```

命令结果是：

ORDENTYR OETYPE(Value)

此时 Value 可以是 DAILY，WEEKLY 和 MONTHLY。

一旦生成了一个命令，可以做：

用 DSPCMD 显示命令的属性

用 CHGCMD 修改命令的属性
用 DLTCMD 删除命令

9.7.1 命令定义源码的清单

在生成命令时，会产生源码清单，下面是它的例子，清单中的数字在其后给出说明：

```
5763SS1 V3R1M0  940909 1       Command Definition          DSTPRODLB/ORDENTRY    05/26/94 10:35:32 2 Page 1 3


Command name  . . . . . :   ORDENTRY
  Library . . . . . :     DSTPRODLB
Command processing program  . . . . . :   ORDENT      4
  Library . . . . . :     DSTPRODLB
Source file . . . . . :   QCMDSRC
  Library . . . . . :     QGPL
Source file member  . . . . . :   ORDENTRY     05/26/94  10:33:32
Validity checker program  . . . . . :   *NONE
Mode in which valid . . . . . :   *PROD
                                  *DEBUG
                                  *SERVICE
Environment allowed . . . . . :   *IREXX
                                  *BREXX
                                  *BPGM
                                  *IPGM
                                  *EXEC
                                  *INTERACT
                                  *BATCH
                                  *BMOD
                                  *IMOD
Allow limited user  . . . . . :   *NO
Max positional parameters . . . . . :   *NOMAX
Prompt file . . . . . :   *NONE
Message file  . . . . . :   QCPFMSG
  Library . . . . . :     *LIBL
Authority . . . . . :   *USE
Replace command . . . . . :   *YES
Text  . . . . . :   Calls order
Help book name  . . . . . :   *NONE
Help bookshelf. . . . . :   *NONE
Help panel group  . . . . . :   HLPORDEN
  Library . . . . . :     *LIBL
Help identifier . . . . . :   HIDORDEN
Help search index . . . . . :   *NONE
Current library . . . . . :   *NOCHG
Product library . . . . . :   *NOCHG
```

```
Prompt override program . . . . . :   *NONE
Compiler  . . . . . :   IBM AS/400 Command Definition Compiler  5

                                Command Definition Source
 6
SEQNBR  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+.  DATE   8
   100-              CMD        PROMPT('Order entry command')                                          05/26/94
7
   200-              PARM       KWD(OETYPE) TYPE(*CHAR) RSTD(*YES) +                                   05/26/94
   300                            VALUES(DAILY WEEKLY MONTHLY) MIN(1) +                              05/26/94
   400                            PROMPT('Type of order entry:')                                     05/26/94
                                       * * * * *  E N D  O F  S O U R C E  * * * * *

5763SS1 V3R1M0  940909               Command Definition              DSTPRODLB/ORDENTRY     05/26/94 10:35:32 Page 2
                                      Cross Reference

Defined Keywords  9
Keyword              Number          Defined      References
OETYPE                 001             200
                             * * * * *  E N D  O F  C R O S S  R E F E R E N C E  * * * * *

5763SS1 V3R1M0  940909               Command Definition              DSTPRODLB/ORDENTRY     05/26/94 10:35:32 Page 3
                                      Final Messages

Message              Sequence
ID                    Number          Sev        Text   10

                                      Message Summary
Total         Info        Error
    0            0           0    11
* CPC0202                              00        Command ORDENTRY created in library DSTPRODLB.  12
                                  * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
```

说　　明：

标　　题：　1. 程序名、版本及 OS/400 日期

2. 运行的日期及时间

3. 清单页码

序　　言：　4. CRTCMD 中规定的参数值（如果没规定，给出缺省值），如果源文件不是数据库文件，则省略成员名、日期和时间。

5. 生成命令定义的编译程序名。

源　　码：　6. 源文件中的行号，行号后的—号指出源语句从这个行号开始，没有—号的说明是前个语句的继续。例如，在行号 200 开始 PARM 语句，继续到行号 300、400（注意：在行号 200 和 300 中 PARM 语句的续行符号（+号））。

注释与其它语句一样处理，也有行号。

7. 源语句

8. 最后修改或增加源语句的日期，如果从没修改或增加过，没有此日期。如果源码不在数据库文件中，也省略日期。

如果在处理命令定义源语句时发现错误，则认为是一个特殊的源语句。在紧接此源语句后打印错误信息，并用*号指出有错的这行，这行包括信息标识，错误级别和信息正文。

交叉引用： 9. 键字表是在命令定义中规定的有效键字的交叉引用表，此表给出键字、在命令中的位置及定义键字的语句顺序号，还有引用键字的语句顺序号。

如果在命令定义中定义了有效标号，则提供标号的交叉引用表（标号表）。此表列出的是标号、定义标号的语句号及引用此标号的语句号。

信　　息： 10. 一般的错误信息没在源码段列出，那么对这样的信息，在这里给出信息标识，产生错误的语句号，错误程度及信息正文。

信息总结： 11. 在处理命令定义语句期间发出的信号数目总计，由错误的严重程度分类总计。

12. 打印的完成信息。

9.7.2 在处理命令定义语句时遇到的错误

此类错误包括语法错误，引用的键字和标号没定义，及丢失语句错，由命令定义编译程序测检到的值错误和语法错误不能生成命令，遇到这种情况之后，命令定义编译程序会继续检查其它的错误。

语法错误和固定值错会防止最后检查用户名及值或引用键字或表的错误，检查会继续进行，这就让你在再生成命令前能尽量看到及改正错误。

在命令定义源清单中，直接与某个源语句有关的错误条件是在这个命令后列出，不与某个源语句有直接关系的信息而是较普通的信息列在清单中的信息段，它不在源语句段出现。

9.8 显示命令定义

能用 DSPCMD 命令显示或打印在 CRTCMD 中做为参数定义的值，它显示如下信息：

限定的命令名。命令名及所在库名

命令处理程序的限定名。如果在用 CRTCMD 或 CHGCMD 时规定了库名，那么会显示这个库名。如果没规定则显示*LIBL，如果 CPP 是 REXX 过程，则给出*REXX。

限定的源文件名。如果源文件是数据库文件，则给出源文件的限定名及在用 CRTCMD 命令时源文件所在的库名，如果源文件不是数据库文件，则此字段为空格。

源成员名。如果源文件是数据库文件，则给出成员名。

如果 CPP 是 REXX 过程，则会给出下列信息：

- REXX 过程成员名
- REXX 过程所在的限定的源文件名
- REXX 命令环境
- REXX 出口程序

有效性检查程序的限定名。在 CRTCMD 或 CHGCMD 时如果规定了库名，则显示库名和程序名。如果没规定库名，则显示*LIBL。

操作的有效方式。

命令运行的合法环境。

命令的位置限制。如果无位置限制，则显示*NOMAX。

提示信息文件的限定名。库　　名是 CRTCMD 运行时放信息文件的库名，如果没有信息文件，则显示*NONE。

DEP 语句的信息文件限定名。如果在生成时规定了信息文件的库名，则显示它，如果规定用库列表，则显示*LIBL，如果在命令中没有 DEP 信息文件，则显示*NONE。

帮助面板标识名。

命令的帮助标识名。

提示替代程序的限定名。

与命令有关的说明。如果没有说明，则显示空格

9.9　修改过程或文件中命令定义的影响

在生成 CL 模块或程序时，用其中的命令定义来生成模块或程序，在运行 CL 过程或文件时，也用其命令定义。如果在 CL 过程或文件中对命令规定了库名，在生成和运行时，命令必须与过程放在同一库中，如果在 CL 过程或文件中时命令规定*LIBL，那么在过程或文件和生成和运行时，用*LIBL 来查找命令。

对命令定义语句做下列修改不用对使用它的模块和程序重新编译（重生成），这些修改有些是在命令定义源码中做的，它要求重新生成命令，另一些修改可用 CHGCMD 命令：

在任何位置加一个可选参数。在位置限制前加可选参数可能影响过程或程序，以及用位置格式规定参数的批输入流。

修改 REL 和 RANGE 检查来减少限制

增加新的特殊值。如果这个值在修改前规定，则将修改过程或文件的动作

修改参数的顺序。修改位置限制前参数的顺序会影响过程或文件，以及用位置格式规定的参数的批输入流。

在简单列表中增加可选元素的数目

修改缺省值，但这可能影响过程或文件的操作

减少在简单列表中必须项的数目

把必须参数改为可选参数

把 RSTD 由*YES 改为*NO

在规定了 FULL(*NO)时，增加长度

把 FULL　由*YES 改为*NO

修改 PROMPT 说明

修改 ALLOW 值来减少限制

如果新的 CPP 接收相当数量及类型参数要修改 CPP 的名字

如果新的有效性检查程序，CPP 接收相当数量及类型参数要修改检查程序名

修改命令运行方式而新方式不影响同一命令的老方式

把 TYPE 改成兼容的及减少限制值，比如从*NAME 改为*CHAR

把 MAX 的值改为大于 1

修改 PASSATR 和 VARY 的值

根据使用命令的 CL 过程或文件，可对命令定义语句做如下修改：

取消一个参数

修改 RANGE 和 REL，增加一些限制

取消特别值

减少列表中允许的元素数目

修改 TYPE，增加一些限制或与源 TYPE 值兼容，例如把*CHAR 改为*NAME 或把*PNAME 改为*CHAR。

增加 SAGVAL 参数，它原先是列表项

从值列表中取消一个值

增加必须列表项的数目

把 SNGVAL 参数改为 SPCVAL

把简单列表改为类似元素的混合列表

把可选参数改为常量

把 RTNVAL 从*YES 改为*NO，或从*NO 改为*YES，把 CASE 的值从*MIXED 改为*MONO。

可对命令定义语句做下列修改，但可能导致用此命令的 CL 过程或文件功能有改变：

修改值的意义

修改缺省值

把 SNGVAL 改为 SPCVAL 参数

把一个值改为 SNGVAL 参数

把列表改为列表中的列表

把 CASE 的值从*MIXED 改为*MONO

对命令定义语句做以下修改，要对用此命令的 CL 过程或文件再生成（重编译）：

增加必须的新参数

取消必须的参数

修改必须的参数的名字

把必须的参数改为常量

把 CPP 改为*REXX 或反之

另外，如果在命令生成或修改时规定 CPP 或有效性检查的限定名为*LIBL，可把 CPP 或有效检查移到库列表中的其它库而不用修改命令定义语句。

9.9.1 修改命令的缺省值

可用 CHGCMDDFT 命令修改命令键字的缺省值，键字必须已有缺省值，因此可改为新的缺省值，要修改的命令必须是 IBM 支持的或用户写的命令，在修改 IBM 支持命令的缺省值时要小心。下面是对修改缺省值的建议：

1.如果修改的是 IBM 支持的命令，要用 CRTDUPOBJ 命令在用户库中生成此命令的复本，这就允许系统中的其它用户在必要时用 IBM 提供的缺省值。

用 CHGSYSLIBL 命令把这个用户库放到 QSYS 库前或在库列表中任何 IBM 提供库的前面，这就让用户使用修改过的命令而不用库的限定。

在系统范围基础上对命令的修改要在用户库中做，此用户库的名字要加到 QSYSLIBL 系统值中放在 QSYS 前，修改过的命令在系统范围内用。如果你要用 IBM 提供的缺省值来运行应用程序，也要用 CHGSYSLIBL 来取消特别的库或影响命令的库限定。

2.在安装特许程序的新版本时，机器中所有特许程序的 IBM 提供的命令都被替代，这样，你要用一个 CL 程序来修改命令，要运行 CL 程序来复制新命令来使用新键字及修改缺省值。

如果 IBM 提供的命令的新键字，那么从先前版本复制的命令可能不能正确的运行。

下例用来删除老版本生成新的修改命令：

```
PGM
```

```
DLTCMD USRQSYS/SIGNOFF
CRTDUPOBJ OBJ(SIGNOFF) FROMLIB(QSYS) OBJTYPE(*CMD) +
          TOLIB(USRQSYS) NEWOBJ(*SAME)
CHGCMDDFT CMD(USRQSYS/SIGNOFF) NEWDFT('LOG(*LIST)')
 .
 .
Repeat the DLTCMD, CRTDUPOBJ and CHGCMDDFT for each
command you want changed
 .
 .
ENDPGM
```

用下面的步骤对 CHGCMDDFT 命令建立 NEWDFT 命令串，此例中使用 USRQSYS/CRTCLPGM 命令。

1.用下列命令在用户库中生成要修改命令的复本。

```
CRTDUPOBJ OBJ(CRTCLPGM) FROMLIB(QSYS) OBJTYPE(*CMD) +
          TOLIB(USRQSYS)  NEWOBJ(*SAME)
```

2.用 SEU 在源文件中输入要修改的命令名。

3.用 F4 键调用命令提示。

4.对要修改的键字输入一个新的缺省值，在此例中，输入：

```
  AUT(*EXCLUDE)         TEXT('Isn't  this  nice  text')
```

5. 必须的键字没有缺省值，为了在源文件中得到命令串，对每个必须的键字规定一个有效值，对 PGM 参数规定 PGM1。

6.用执行键把命令串放到源文件中，返回的命令串如下：

```
USRQSYS/CRTCLPGM PGM(PGM1) AUT(*EXCLUDE) +
TEXT('Isn't this nice text')
```

7.从命令串中去掉必须的键字：

```
USRQSYS/CRTCLPGM  AUT(*EXCLUDE) +
TEXT('Isn't this nice text')
```

能修改的仅是已有缺省值的参数、元素或限定名，如不然没有缺省值可供修改。

8.在开头插入 CHGCMDDFT 命令：

```
CHGCMDDFT USRQSYS/CRTCLPGM AUT(*EXCLUDE) +
TEXT('Isn't this nice text')
```

9.输入的 NEWDFT 键字要有引号：

```
CHGCMDDFT USRQSYS/CRTCLPGM ’AUT(*EXCLUDE) +
TEXT(’Isn’t this nice text’)’
```

10.由于在 NEWDFT 中有嵌入的引号，它们必须要双写：

```
CHGCMDDFT USRQSYS/CRTCLPGM ’AUT(*EXCLUDE) +
TEXT(’Isn’t this nice text’’)’
```

11.按 F4 键调用命令提示，然后按 F11 请求键字的提示，将出现下面的显示：

```
Command . . . . . :   CMD          R   CRTCLPGM
  Library . . . . . :                   USRQSYS
New default parameter string:   NEWDFT      R   ’AUT(*EXCLUDE)
TEXT(’Isn’t this nice text’’)’
```

12.如果按执行键，CHGCMDDFT 命令串为：

```
CHGCMDDFT CMD(USRQSYS/CRTCLPGM) NEWDFT(’AUT(*EXCLUDE) +
          TEXT(’Isn’t  this nice text’’)’)
```

13.用 F1 键结束 SEU，生成和运行 CL 过程或文件。

14.USRQSYS/CRTCLPGM 有缺省值 AUT　*EXCLUDE，TEXT 的内容为’Isn’t this nice text’。

9.9.1.1　例 1

要给 CRTPF 的 MAXMBRS 键字提供*NOMAX 的缺省值：

```
CRTPF FILE(FILE1) RCDLEN(96) MAXMBRS(1)
  .
  .
CHGCMDDFT CMD(CRTPF) NEWDFT(’MAXMBRS(*NOMAX)’)
```

9.9.1.2　例 2

要给 CRTPF 的 MAXMBRS 提供缺省值 10，做：

```
CRTPF FILE(FILE1) RCDLEN(96) MAXMBRS(*NOMAX)
  .
  .
CHGCMDDFT CMD(CRTPF) NEWDFT(’MAXMBRS(10)’)
```

9.9.1.3　例 3

下例对 CRTCLPGM 的 SRCFILE 键字提供缺省值 LIB001 做第一个限定名，提供 FILE001 做 SRCFILE 的第二个限定名，AUT 键字有*EXCLUDE 的缺省值。

```
CRTCLPGM PGM(PROGRAM1) SRCFILE(*LIBL/QCMDSRC)
  .
  .
CHGCMDDFT CMD(CRTCLPGM) +
          NEWDFT('SRCFILE(LIB001/FILE001) AUT(*EXCLUDE)')
```

9.9.1.4 例 4

下例对 CHGJOB 的 PRTTXT 键字给出‘Isn’t this print text‘的缺省值：

```
CHGJOB PRTTXT('Isn't this print text')
  .
  .
CHGCMDDFT CMD(CHGJOB) +
          NEWDFT('PRTTXT('Isn't this print text'')')
```

9.9.1.5 例 5

下例对 CRTLF 的 DTAMBRS 键字的第一个列表项的第一个限定（库名）给出缺省值 QGPL，第二个列表项的新缺省值为 MBR1：

```
CRTLF FILE(FILE1) DTAMBRS(*ALL)
  .
  .
CHGCMDDFT CMD(CRTLF) +
          NEWDFT('DTAMBRS((QGPL/*N (MBR1)))')
```

对整个 DTAMBRS 列表来说，SNGVAL 为*ALL，在原命令提示显示中，不给出库名的缺省值*CURRENT 和成员名*NONE，可以把这两个缺省值改为新值，但也不能在原提示中显示，这是由于对整个 DTAMBRS 列表来讲是*ALL 这个单值。

9.9.1.6 例 6

生成一个命令来显示一个作业的假脱机文件：

```
CRTDUPOBJ OBJ(WRKJOB) FROMLIB(QSYS) +
          TOLIB(MYLIB) NEWOBJ(WRKJOBSPLF)
WRKJOBSPLF  OPTION(*SPLF)
  .
  .
CHGCMDDFT CMD(MYLIB/WRKJOBSPLF) +
          NEWDFT('OPTION(*SPLF)')
```

9.10 写一个命令处理程序或过程

命令处理程序（CPP）可以是一个 CL 或 HLL 程序或一个 REXX 过程，用 CL 或 HLL 写的程序也可用 CALL 命令直接调用。可用 STRREXPRC 命令直接调用 REXX 过程。在 CRTCMD 运行时，CPP 不必须存在，如果用*LIBL 做库名，在运行生成命令时，用库列表

来查找 CPP。

做为运行 CPP 结果发出的信息能送往作业信息队列，能自动地显示或打印，也可把此显示发送给需要的显示工作站。

注：

1.在命令中定义的参数是用它们定义的顺序（PARM 语句的顺序）分别传送的。

2.十进制值做为压缩十进制，用在 PARM 语句中规定的长度传送给 HLL 和 CL 程序。

3.字符、名字和逻辑值做为一个字符串，用在 PARM 中规定的长度传送给 CL 或 HLL。

9.10.1 写一个 CL 或 HLL 的 CPP

图 9—3 给出 CRTCMD 与命令定义语句、命令处理程序之间的关系。

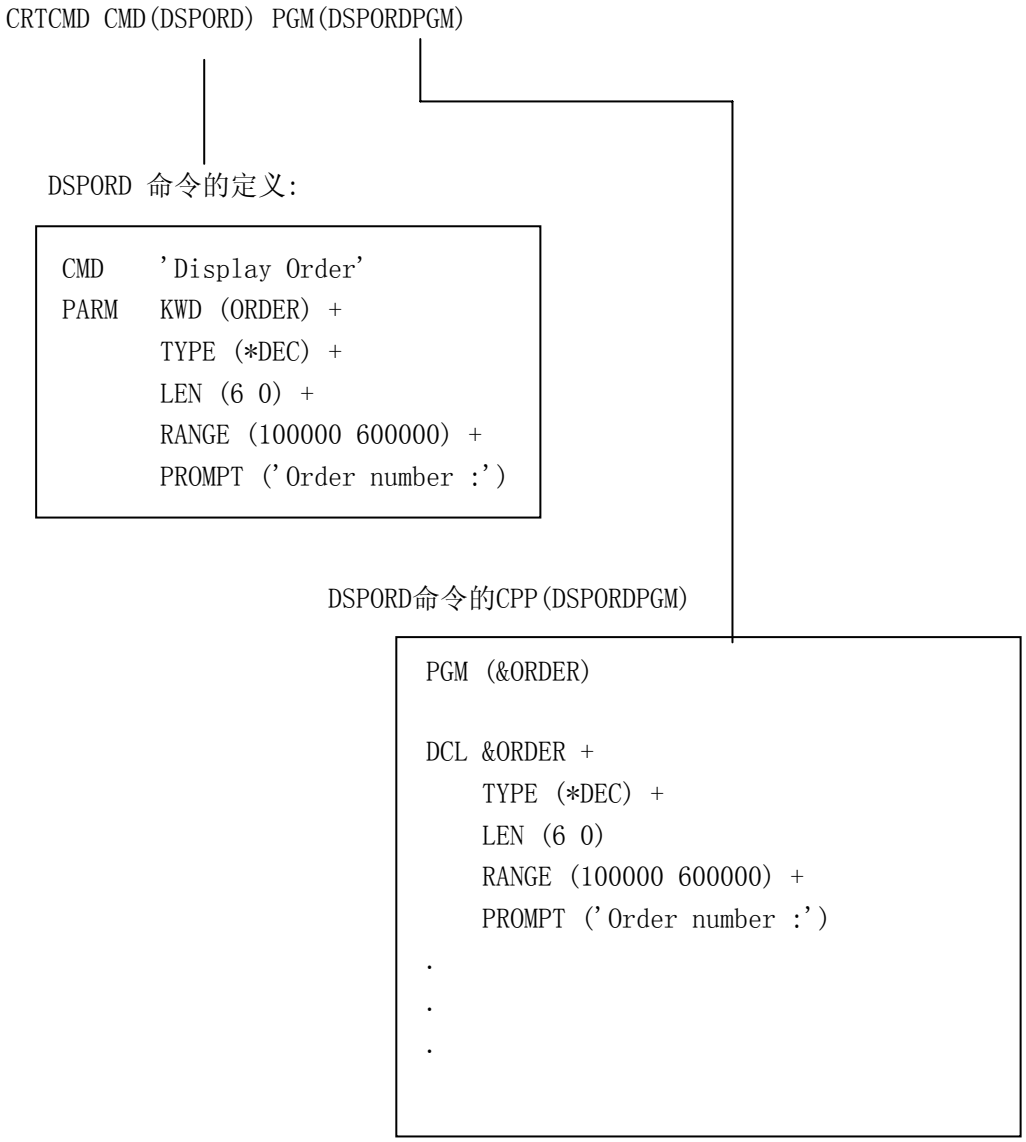


图 9—3 CL、HLL 的命令之间的关系

如果 CPP 是用 CL 写的程序，接收参数值的变量必须说明为与每个 PARM 语句规定的长度和类型相对应，下面给出这些对应关系（对参数 ORDER 的说明在图 9—3）。

PARM 语句类型	PARM 语句长度	说明的变量类型	说明的变量长度
*DEC	x y(1)	*DEC	x y(1)
*LGL	1	*LGL	1
*CHAR	n	*CHAR	看(2)
*NAME	n	*CHAR	看(2)
*CNAME	n	*CHAR	看(2)
*SNAME	n	*CHAR	看(2)
*GENERIC	n	*CHAR	看(2)
*CMDSTR	n	*CHAR	看(2)
*DATE	7	*CHAR	7
*TIME	6	*CHAR	6
*INT2	n	*CHAR	2
*INT4	n	*CHAR	4

(1)X 表示长度，Y 表示小数点位数。

(2)对字符变量，如果传送值的长度大于说明长度，值截断为说明的长度。如果规定了 RTNVAL(*YES)，说明的长度必须等于在 PARM 语句中定义的长度。

用 CL 写的 CPP 在处理二进制值时（如*INT2 和*INT4），程序把它们做为字符值接收，可用%BINARY 把它们转换为十进值，CPP 的例子，请看 9.12。

9.10.2 写个 REXX 的 CPP

图 9—4 给出 CRTCMD、命令定义语句和 REXX CPP 之间的关系：

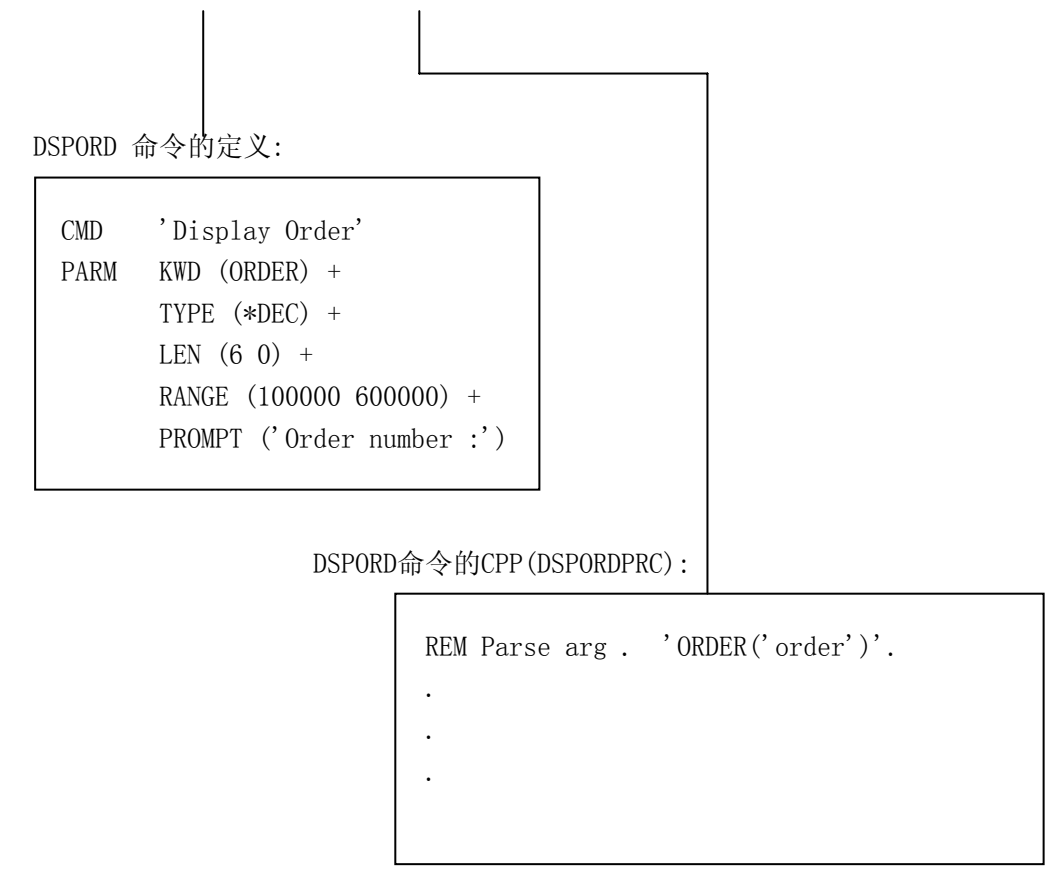


图 9—4　REXX 的命令之间关系

9.11　写一个有效性检查程序

对命令写一个有效性检查程序，在 CRTCMD 的 VLDCHK 参数中写有效性检查程序的名字。在 CRTCMD 运行时程序不一定要存在。如果用*LIBL 做库限定名，在运行生成命令时，用库列表来查找有效性检查程序。

下面是写此程序的二方面考虑：

如果命令语法正确，才调用有效性检查程序，所有参数象传送给 CPP 一样传给检查程序。

不能用检查程序来修改参数值，这是因为修改的值不能传送给 CPP。

下面介绍怎样从用 CL 写的检查程序发送信息及系统怎样识别它。

如果有效性检查程序检查到一个错误，它要给它的调用者发送诊断信息然后发送 CPF0002 逃逸信息。例如，如果需要帐号长无效的信息，那么要往一个信息文件中加下面的信息描述：

```
ADDMSGD      MSG('Account number &2 no longer valid') +
             MSGID(USR0012) +
             MSGF(QGPL/ACTMSG) +
             SEV(40) +
             FMT((*CHAR 4) (*CHAR 6))
```

替换变量&1 不在信息中，但它用 FMT 参数定义为 4 个字符。&1 由系统接收来用，必须总是 4 个字符，如果这个替换变量&1 仅是在信息中定义的替换变量，你必须保证在发送信息时 4 字节信息数据中不能有空格。

在有效检查中要规定下面命令，信息才能发送给系统：

```
SNDPGMMSG    MSGID(USR0012) MSGF(QGPL/ACTMSG) +
             MSGDTA('0000' || &ACCOUNT) MSGTYPE(*DIAG)
```

在检查程序发送完所有诊断信息后，然后发送 CPF0002，所用命令如下：

```
SNDPGMMSG    MSGID(CPF0002) MSGF(QCPFMSG) +
             MSGTYPE(*ESCAPE)
```

在系统接收 CPF0002 时，发出 CPF0001 信息给调用程序指出已找到错误。

在用户定义的有效检查程序也可以定义 CPD0006，在信息数据中可以发送立即信息。在下例中，信息前要有 4 个字符的零。

下面是有效性检查程序的例子：

```
PGM PARM(&PARM01)
DCL VAR(&PARM01) TYPE(*CHAR) LEN(10)
IF COND(&PARM01 *EQ 'ERROR') THEN(DO)
SNDPGMMSG MSGID(CPD0006) MSGF(QCPFMSG) +
          MSGDTA('0000 DIAGNOSTIC MESSAGE FROM USER-DEFINED +
```

```
            VALIDITY CHECKER INDICATING THAT PARM01 IS IN ERROR.') +
            MSGTYPE(*DIAG)
SNDPGMMSG MSGID(CPF0002) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)
ENDDO
ELSE
 .
 .
 .
ENDPGM
```

9.12 定义和生成命令的例子

这节包括定义命令以及生成命令的例子。

9.12.1 调用应用程序

可以写一个命令调用应用程序，此时 OS/400 完成对传给程序的参数的检查，但如果用 CALL 命令来调用应用程序，应用程序必须做有效性检查。

例如，写标号程序（LBLWRT）为 1 或 2 类型格式的新用户写任意一个标号，在运行 LBLWRT 时，它要三个参数：客户号、标号和所用格式类型（ONE 或 TWO）。如果直接从显示上调用此程序，第二个参数对程来说是错误的格式，在 CALL 命令中，一个数值常量总是 15 个数字 5 位小数，而 LBLWRT 程序接收 3 位无小数数字，生成一个命令就可提供程序需要的格式。

调用 LBLWRT 程序的命令定义语句是：

```
CMD PROMPT('Label Writing Program')
PARM  KWD(CUSNBR) TYPE(*CHAR) LEN(5) MIN(1) +
      PROMPT('Customer Number')
PARM  KWD(COUNT) TYPE(*DEC) LEN(3) DFT(20) RANGE(10 150) +
      PROMPT('Number of Labels')
PARM  KWD(FRMTYP) TYPE(*CHAR) LEN(3) DFT('TWO') RSTD(*YES) +
      SPCVAL(('ONE') ('TWO') ('1' 'ONE') ('2' 'TWO')) +
      PROMPT('Form Type')
```

对第二个参数 COUNT，规定 20 做缺省值，在 RANGE 参数仅允许对标号数使用从 10—150 的值。

对第三个参数 FRMTYP，SPCVAL 允许工作站用户输入‘ONE’、‘TWO’、‘1’或‘2’，命令会对它做必要的转换。

这个命令的 CPP 用应用程序 LBLWRT，如果它是 RPG/400 程序，那么程序中用来接收参数的规定应如下：

```
*ENTRY    PLIST
          PARM     CUST    5
          PARM     COUNT   30
          PARM     FORM    3
```

CRTCMD 命令为：

```
CRTCMD CMD(LBLWRT) PGM(LBLWRT) SRCMBR(LBLWRT)
```

9.12.2 替代一个缺省值

可以生成一个命令来对 IBM 提供的命令给出缺省值，这样减少工作站用户必须输入的内容。例如，可以生成一个 SAVLIBTAP 命令来格式化一个磁带，然后把一个库备份到 TAPE1 带上，这个命令给标准的 SAVLIB 提供缺省值，这样工作站用户只要输入库名即可。

SAVLIBTAP 的命令定义为：

```
CMD PROMPT('Save Library to Tape')
PARM KWD(LIB) TYPE(*NAME) LEN(10) MIN(1) +
  PROMPT('Library Name')
```

CPP 是：

```
PGM PARM(&LIB)
DCL &LIB TYPE(*CHAR) LEN(10)
INZTAP DEV(TAPE1) CHECK(*NO)
SAVLIB LIB(&LIB) DEV(TAPE1)
ENDPGM
```

CRTCMD 命令为：

```
CRTCMD CMD(SAVLIBTAP) PGM(SAVLIBTAP) SRCMBR(SAVLIBTAP)
```

9.12.3 显示一个输出队列

可以生成一个命令来显示输出队列，缺省为显示 PGMR，DSPOQ 命令可以让用户显示在库列表中库里的任何队列，也提供一个打印的选项。

DSPOQ 的命令定义语句是：

```
CMD PROMPT('WRKOUTQ.-Default to PGMR')
PARM  KWD(OUTQ) TYPE(*NAME) LEN(10) DFT(PGMR) +
      PROMPT('Output queue:')
PARM  KWD(OUTPUT) TYPE(*CHAR) LEN(6) DFT(*) RSTD(*YES)
      VALUES(* *PRINT) PROMPT('Output')
```

在 PARM 语句中的 RSTD 参数规定仅能输入值列表中的一个值。DSPOQ 的 CPP 为：

```
PGM PARM(&OUTQ &OUTPUT)
DCL &OUTQ TYPE(*CHAR) LEN(10)
DCL &OUTPUT TYPE(*CHAR) LEN(6)
WRKOUTQ OUTQ(*LIBL/&OUTQ) OUTPUT(&OUTPUT)
ENDPGM
```

CRTCMD 命令为

```
CRTCMD CMD(DSPOQ) PGM(DSPOQ) SRCMBR(DSPOQ)
```

接下来的命令 DSPOQ1 是前面命令的加些变化，它允许工作站用户输入输出队列的限定名，库名的缺省值为*LIBL。

DSPOQ1 的命令定义为：

```
            CMD PROMPT('WRKOUTQ.-Default to PGMR')
            PARM    KWD(OUTQ) TYPE(QUAL1) +
                    PROMPT('Output queue:')
            PARM    KWD(OUTPUT) TYPE(*CHAR) LEN(6) RSTD(*YES) +
                    VALUES(* *PRINT) DFT(*) +
                    PROMPT('Output')
QUAL1:      QUAL TYPE(*NAME) LEN(10) DFT(PGMR)
            QUAL TYPE(*NAME) LEN(10) DFT(*LIBL) +
            SPCVAL(*LIBL)
```

QUAL 语句用来定义用户对 OUTQ 参数可以输入的限定名，如果用户没输入此名，则用*LIBL/PGMR，用 SPCVAL 参数的原因是任何库名必须遵循有效名的原则（比如必须以 A—Z 开头），而*LIBL 不符合此原则，它规定如果输入*LIBL，OS/400 会忽略名字的有效性原则。

DSPOQ1 的 CPP 为：

```
PGM PARM(&OUTQ &OUTPUT)
DCL &OUTQ TYPE(*CHAR) LEN(20)
DCL &OBJNAM TYPE(*CHAR) LEN(10)
DCL &LIB TYPE(*CHAR) LEN(10)
DCL &OUTPUT TYPE(*CHAR) LEN(6)
CHGVAR &OBJNAM %SUBSTRING(&OUTQ 1 10)
CHGVAR &LIB %SUBSTRING(&OUTQ 11 10)
WRKOUTQ OUTQ(&LIB/&OBJNAM) OUTPUT(&OUTPUT)
ENDPGM
```

由于限定名是做为 20 字符变量的一个命令中送出，在程序中必须用%SUBSTRING 或%SST，这样才能符合 CL 语法。

9.12.4 多次显示 IBM 命令信息

CLROUTQ 命令发布完成信息 CPF3417，它给出删除项的号码、没删除的号码以及输出队列的名字，如果在 CPP 中运行 CLROUTQ 命令，仍会发出此信息，但由于它不是直接由 CPP 发出的，所以成为一个详细信息。如一个用户定义的 CLROUTQ 命令从程序员菜单发出，则不会显示信息，但你可以从自己的 CPP 来接收 IBM 信息，也可重新发送它。

例如，生成一个命令 CQ2 来清理输出队列 QPRINT2：

CQ2 的命令定义语句为：

```
        CMD PROMPT ('Clear QPRINT2 output queue')
```

CRTCMD 命令为：

```
CRTCMD CMD(CQ2) PGM(CQ2)
```

CPP 接收完成信息并显示它，CPP 为：

```
PGM /* Clear QPRINT2 output queue CPP */
DCL &MSGID TYPE(*CHAR) LEN(7)
DCL &MSGDTA TYPE(*CHAR) LEN(100)
CLROUTQ QPRINT2
RCVMSG MSGID(&MSGID) MSGDTA(&MSGDTA) MSGTYPE(*COMP)
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) MSGTYPE(*COMP)
ENDPGM
```

CPF3417 的 MSGDTA 长度为 28 字节，但变量&MSGDTA 定义为 100 字节，由于没用的位都忽略，故大多数信息都可用此方法。

9.12.5 生成缩写的命令

9.12.5.1 例 1

可生成自己的命令来简化 IBM 提供的命令，或限制用户使用的参数。例如，仅让用户能修改打印设备参数，可以生成 CJ 命令，下面三个步骤是生成 CJ 命令的方法：

第一步：命令定义源语句

```
CMD    PROMPT('Change Job')

PARM   KWD(PRTDEV) +
       TYPE(*NAME) +
       LEN(10)   +
       SPCVAL(*SAME *USRPRF *SYSVAL *WRKSTN) +
       PROMPT('Printer Device')
```

第二步：处理程序

```
PGM PARM(&PRTDEV)
DCL VAR(&PRTDEV)  TYPE(*CHAR)  LEN(10)
CHGJOB  PRTDEV(&PRTDEV)
ENDPGM
```

第三步：CRTCMD 命令

```
CRTCMD  CMD(CJ)  PGM(CJ)  SRCMBR(CJ)
```

9.12.5.2　例 2

生成一个 DW1 命令来启动打印机写入器 W1。

命令定义语句为：

```
CMD/*启动打印机写入器的命令*/
```

命令处理程序为：

```
PGM
STRPRTWTR DEV(QSYSPRT) OUTQ(QPRINT) WTR(W1)
ENDPGM
```

CRTCMD 命令为

```
CRTCMD CMD(DW1) PGM(DW1) SRCMBR(DW1)
```

9.12.6　增加或减少日期值

可以写一个命令增加或减少用户定义的日期天数，并把新值做为一个变量返回，详细内容请看 QUSRTOOL 库中 QATTINFO 文件中的成员 ADDDAT。

9.12.7　删除文件和源成员

可以生成一个命令来删除文件及在 QDDSSRC 中的相关源成员：

DFS 的命令定义源语句为：

```
CMD  PROMPT('Delete File and Sourde')
PARM  KWD(FILE) TYPE(*NAME) LEN(10) PROMPT('File Name')
```

所写的 CPP 假定文件名和源成员名相同，也假定它们所在的库都在库列表中。如果程序不能删除文件，会发送信息且命令会试图去掉源成员，如果没有源成员，发送逃逸信息。

命令处理程序是：

```
PGM PARM(&FILE)
DCL &FILE TYPE(*CHAR) LEN(10)
DCL &MSGID TYPE(*CHAR) LEN(7)
DCL &MSGDTA TYPE(*CHAR) LEN(80)
DCL &SRCFILE TYPE(*CHAR) LEN(10)
MONMSG MSGID(CPF0000) EXEC(GOTO ERROR) /* 监控所有信息 */
DLTF &FILE
MONMSG MSGID(CPF2105) EXEC(DO) /* 没找到 */
RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*INFO) +
      MSGDTA(&MSGDTA)
GOTO TRYDDS
```

```
ENDDO
RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
    /* 完成删除文件 */
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
          MSGDTA(&MSGDTA) /* 在QDDSSRC文件中试 */
TRYDDS:   CHKOBJ QDDSSRC OBJTYPE(*FILE) MBR(&FILE)
          RMVM QDDSSRC MBR(&FILE)
          CHGVAR &SRCFILE 'QDDSSRC'
          GOTO END
END:      RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
              /* 完成取消成员 */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
          MSGDTA(&MSGDTA)
          RETURN
ERROR:    RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
              /* 逃逸信息 */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
               MSGDTA(&MSGDTA)
ENDPGM
```

9.12.8 删除程序目标

可以生成一个命令来删除 HLL 程序及它们相应的源成员。

DPS 的命令定义源语句为：

```
CMD PROMPT ('Delete Program and Source')
PARM KWD(PGM) TYPE(*NAME) LEN(10) PROMPT('Program Name')
```

写的 CPP 假定程序名和源成员名是相同的，使用 IBM 提供的源文件（QCLSRC，QRPGSRC 和 QCBLSRC），程序也假定程序和源文件都在库列表中。如果不能删除程序，则发送信息，命令会试图取消源成员，如果源成员不存在，则发送逃逸信息。CPP 为：

```
PGM PARM(&PGM)
DCL &PGM TYPE(*CHAR) LEN(10)
DCL &MSGID TYPE(*CHAR) LEN(7)
DCL &MSGDTA TYPE(*CHAR) LEN(80)
DCL &SRCFILE TYPE(*CHAR) LEN(10)
MONMSG MSGID(CPF0000) EXEC(GOTO ERROR) /* 监控所有信息 */
DLTPGM &PGM;
MONMSG MSGID(CPF2105) EXEC(DO) /* 没找到*/
RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*INFO) +
      MSGDTA(&MSGDTA)
GOTO TRYCL  /* 试图删除源成员 */
ENDDO
```

```
RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
    /* 完成删除程序 */
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
     MSGDTA(&MSGDTA) /* 在QCLSRC中试 */
TRYCL:   CHKOBJ QCLSRC OBJTYPE(*FILE) MBR(&PGM)
         MONMSG MSGID(CPF9815) EXEC(GOTO TRYRPG) /* 不是CL成员 */
         RMVM QCLSRC MBR(&PGM)
         CHGVAR &SRCFILE 'QCLSRC'
         GOTO END
TRYRPG:   /* 在 QRPGSRC 文件中试 */
         CHKOBJ QRPGSRC OBJTYPE(*FILE) MBR(&PGM)
         MONMSG MSGID(CPF9815) EXEC(GOTO TRYCBL) /* 不是RPG成员 */
         RMVM QRPGSRC MBR(&PGM)
         CHGVAR &SRCFILE 'QRPGSRC'
         GOTO END
TRYCBL:   /* 在 QCBLSRC 文件中试 */
         CHKOBJ QCBLSRC OBJTYPE(*FILE) MBR(&PGM)
         /* 在最后一个源文件中发现没找到的CPF0000条件    */
         RMVM QCBLSRC MBR(&PGM)
         CHGVAR &SRCFILE 'QCBLSRC'
         GOTO END
TRYNXT:   /* 插入一个附加的源文件 */
          /* 在做完TRYCL和TRYRPG后，在TRYCBL中的CHKOBJ后加一个MONMSG。*/
END:      RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
             /*完成取消成员 */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
            MSGDTA(&MSGDTA)
          RETURN
ERROR:    RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
             /* 逃逸信息 */
          SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
             MSGDTA(&MSGDTA)
          ENDPGM
```

第十章　调试 ILE 程序

调试让你检查、诊断和排除程序中的错误，可用 ILE 源码调试程序来调试 ILE 程序。这章介绍如何使用 ILE 源码调试程序。它包括：

准备要调试的 ILE 程序
启动调试环境
从调试环境增加或取消程序
从调试环境显示程序源码
设置及取消条件或非条件断点
单步通过程序

显示变量的值
修改变量的值
显示变量的属性
使速记名等于一个变量、表达式或调试命令

在调试和检查程序时，要保证库列表改为程序直接处理有测试数据的测试库，这样才能不影响实际数据的内容。

可用下列命令避免修改在产品库中的数据库文件：

用有 UPDPROD(*NO)的 STRDBG 命令

用 CHGDBG 命令

上面两个命令的详细内容请看 CL 参考手册。

ILE 源码调试的内容请看 ILE 概念一书的第九章。

10.1 ILE 源码调试

用 ILE 源码调试来检查和排除程序目标和服务程序中的错误，也可用于：

调试任何 ILE CL 或混合的 ILE 语言应用程序
在程序运行时用调试命令监控程序流程
显示程序源码
设置和取消条件或非条件断点
单步通过一定数目的语句
显示或修改变量的值
显示变量的属性

在由于断点或单步命令程序停止时，给出在程序停止点上应用程序模块目标的内容。此时，可以输入更多的调试命令。用源码调试之前，在用 CRTCLMOD 或 CRTBNDCL 生成模块目标或程序目标时，必须用调试选项 DBGVIEW，在设置断点或其它 ILE 源码调试选项后，才能调用程序。

10.2 调试命令

做 ILE 源码调试可以使用很多调试命令。这些命令和参数是在显示模块源码和表达式求值显示底部的命令行中输入的。可以用大写、小写或混合方式输入命令。

注：在源码调试命令行输入的调试命令不是 CL 命令。

表 10-1 给出这些调试命令，联机帮助会说明这些命令以及它们的缩写：

表 10-1 ILE 源码调试命令

调试命令	说　　明
ATTR	允许显示变量的属性。这些属性是在调试符号表中的一条记录，包括大小和类型。
BREAK	在程序检查时允许输入条件及非条件断点。用 BREAK 位置 WHEN 表达式来输入一个条件断点。
CLEAR	取消条件或非条件断点。
DISPLAY	显示由 EQUATE 命令分配的名字和定义。也允许显示与‘显示模块源码’中不同的源模块。模块必须在当前程序目标中存在。
EQUATE	允许分配一个表达式变量或调试命令给一个速记名使用。

EVAL	显示或修改变量的值或显示表达式的值。
QUAL	定义随后出现在 EVAL 命令中的变量范围。
STEP	运行被调试程序的一个或多个语句。
FIND	查找当前显示模块的指定参数或串的说明。
UP	把显示窗口向前移动到显示开始的指定数。
DOWN	把显示窗口向前移动到显示结尾的指定数。
LEFT	把显示窗口向左移动到规定字符数。
RIGHT	把显示窗口向右移动到规定字符数。
TOP	使显示定位于第一行。
BOTTOM	使显示定位于最后一行。
NEXT	使显示定位于当前显示源码的下一个断点。
PREVIOUS	使显示定位于当前显示源码的前一个断点。
HELP	对源码调试命令给出可用的帮助信息。
SET	对所有后续的 FIND 是否区别大小写，也允许修改更新产品文件的值。
WATCH	显示当前活动监视条件的列表。

10.3 准备要调试的程序目标

在用 ILE 源码调试前，必须用 CRTCLMOD 或 CRTBNDCL 命令且规定 DBGVIEW 选项。

对每个要调试的 CL 模块，可以生成一个或三个视图：

根源码视图

清单视图

语句视图

10.3.1 使用根源码视图

根源码视图包括源成员中的源语句。

要一起使用 ILE 源码调试和根源码视图，ILE CL 编译程序在生成模块(*MONDLE)时生成根源码。

注：生成模块是用引用根源目标中源语句的位置，而不是把源语句复制到显示中，这样不能在生成模块和生成模块所用成员之间修改、改名或移动根源成员。

要用根源码视图调试 ILE CL 模块，在 CRTCLMOD 或 CRTBNDCL 命令中，对 DBGVIEW 规定*SOURCE 或*ALL。

下面是生成一个根源码视图的方法之一：

```
CRTCLMOD MODULE(MYLIB/MYPGM) SRCFILE(MYLIB/QCLLESRC) SRCMBR(MYPGM) TEXT('CL
Program') DBGVIEW(*SOURCE)
```

它生成 MYPEM 模块的一个根源码视图。

10.3.2 使用清单视图

一个清单视图非常类似于编译清单的源码部分或用 ILE CL 编译程序产生的假脱机文件。

要用清单视图调试模块，在用 CRTCLMOD 或 CRTBNDCL 命令生成模块时规定 DBGVIEW 为*LIST 或*ALL。

生成清单视图的一个方法为：

```
 CRTCLMOD
MODULE(MYLIB/MYPGM) SRCFILE(MYLIB/QCLLESRC) SRCMBR(MYPGM) TEXT('CL Program')
DBGVIEW(*LIST)
```

10.3.3 使用语句视图

一个语句视图不包括任何 CL 源数据，但可用过程名和在编译清单中找到的语句号加一些断点。要用语句视图调试程序，需要复制编译清单。

注：在用语句视图调试时，在‘显示模块源码’显示中没有数据给出。

要用语句视图调试程序，在用 CRTCLMOD 或 CRTBNDCL 命令生成模块时，要规定 DBGVIEW 为*STMT，*SQURCE，*LIST 或*ALL。

生成语句视图的一个方法是：

```
CRTCLMOD MODULE(MYLIB/MYPGM) SRCFILE(MYLIB/QLSRC) SRCMBR(MYPGM) TEXT('CL Program')
DBGVIEW(*STMT)
```

10.4 启动 ILE 源码调试

在生成调试视图后，可以开始调试应用程序了。用 STRDBG 命令来启动 ILE 源码调试，调试一旦启动，那么仅在输入 ENDDBG 命令后才结束。

开始，用 STRDBG 的 PGM 参数可以把多至 20 个程序加到调试环境中，它们可以是 ILE 或 OPM 程序。要启动调试三个程序的调试环境，进入下列命令：

```
 STRDBG PGM(*LIBL/MYPGM1 *LIBL/MYPGM2 *LIBL/MYPGM3)DBGMODSRC(*YES)
```

注：对加到调试环境的程序必须有*CHANGE 的权限。在输入 STRDBG 命令后，出现‘显示模块源码’的显示，其中给出程序中联编的第一个模块。

10.5 往调试环境中加一个程序目标

在启动调试环境后，可往其中加多个程序目标，在‘处理模块列表’显示中的第一行用选项 1，再写上程序名即可。在表 10-1 列出了调试命令。‘处理模块列表’显示可从‘显示模块源码’显示中按 F14 键得到。要加一个服务程序，把缺省的程序类型从*PGM 改为*SRVPGM。在任何时间对加到调试环境中的程序数目都没有限制。

```
                          Work with Module List
                                                          System:   SYSTEM01
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        weekday2       *LIBL      *PGM
          DSPWKDAY            MYLIB         *PGM
          DSPWKDAY                          *MODULE      Selected
          AABP1                             *MODULE



                                                                    Bottom
Command
===>
F3=Exit    F4=Prompt    F5=Refresh    F9=Retrieve    F12=Cancel
```

图 10-1　往调试环境中加一个 ILE 程序

在按执行键后，程序 WEEKDAY2 加到调试环境中。

```
                          Work with Module List
                                                          System:   SYSTEM01
Type options, press enter.
  1=Add program   4=Remove program   5=Display module source
  8=Work with module breakpoints

Opt      Program/module      Library         Type
          WEEKDAY2       *LIBL      *PGM
          WEEKDAY2            MYLIB         *PGM
          WEEKDAY2                          *MODULE
          DSPWKDAY            MYLIB         *PGM
          DSPWKDAY                          *MODULE      Selected
          AABP1                             *MODULE



                                                                    Bottom
Command
===>
F3=Exit    F4=Prompt    F5=Refresh    F9=Retrieve    F12=Cancel
Program WEEKDAY2 added to source debugger.
```

图 10-2　往调试环境加一个 ILE 程序

在显示底部出现的信息表示 WEEKDAY2 已加到调试环境中，在做完加程序后，用 F3 键结束，回到‘显示模块源码’的显示中，也可用选项 5 来选择和显示一个模块。

要往调试环境中加一个 OPM 程序，用 ADDPGM 命令，在任何时候都可以往调试环境中加多至 20 个 OPM 程序。

10.6　从调试环境中去掉一个程序目标

在启动调试环境后，可从中取消一个程序，在‘处理模块列表’显示中，在要取消的程序的 OPT 列写 4。看图 10-3。这个显示可从‘显示模块源码’显示中用 F14 键得到。要取消一个服务程序，把程序类型从*PGM 改为*SRVPGM。

```
                          Work with Module List
                                                          System:   SYSTEM01
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         WEEKDAY2          MYLIB        *PGM
    WEEKDAY2                          *MODULE
    DSPWKDAY          MYLIB           *PGM
    DSPWKDAY                          *MODULE     Selected
    AABP1                             *MODULE



                                                                   Bottom
Command
===>
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F12=Cancel
```

图 10-3　从调试环境中取消一个程序

在按执行键后，程序 WEEKDAY2 从调试环境中取消了。

```
                              Work with Module List
                                                                  System:   SYSTEM01
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
          DSPWKDAY          MYLIB          *PGM
          DSPWKDAY                         *MODULE     Selected
          AABP1                            *MODULE

                                                                          Bottom
Command
===>
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F12=Cancel
Program WEEKDAY2 removed from source debugger.
```

图 10-4　从调试环境中取消一个 ILE 程序

做完从调试环境取消程序时，用 F3 键回到前屏。

注：要从调试环境取消程序时，必须对程序有*CHANGE 权限。要从调试环境取消一个 OPM 程序，用 RMVPGM 命令。

10.7　程序源码视图

‘显示模块源码’显示一次给出一个程序目标的一个模块。如果模块是用下列调试视图选项编译的，就可以显示模块的源码：

DBGVIEW(*ALL)

DBGVIEW(*SQURCE)

DBGVIEW(*LISTING)

有两种方法来修改在显示中给出的内容：

修改一个视图

修改一个模块

在修改视图时，IEL 源码调试程序映象给出修改视图的相同位置。在修改模块时，把显示视图上的可执行语句放在内存，在重新显示模块时再给出视图。有断点设置的行号高亮显示。在断点、单步或信息导致程序停止给出显示时，发生事件的原行高亮度显示。

10.8 修改模块目标

可在‘处理模块列表’显示中用选项 5 来修改目标模块。从‘显示模块源码’中用 F14 键可访问图 10-5 给出的显示。要选择一个模块，在它的 OPT 列写 5。

```
                              Display Module Source

Program:   DSPWKDAY       Library:   MYLIB          Module:   DSPWKDAY
    24      500-              CALL       PGM(WEEKDAY2) PARM(&DAYOFWK)
    25      600-              IF         COND(&DAYOFWK *EQ 1) THEN(CHGVAR +
    26      700                            VAR(&WEEKDAY) VALUE('Sunday'))
    27      800-              ELSE       CMD(IF COND(&DAYOFWK *EQ 2) THEN(CHGV
    28      900                            VAR(&WEEKDAY) VALUE('Monday')))
    29     1000-              ELSE       CMD(IF COND(&DAYOFWK *EQ 3) THEN(CHGV
    30     1100                           VAR(&WEEKDAY) VALUE('Tuesday')))
    31     1200-              ELSE       CMD(IF COND(&DAYOFWK *EQ 4) THEN(CHGV
    32     1300                           VAR(&WEEKDAY) VALUE('Wednesday')))
    33     1400-              ELSE       CMD(IF COND(&DAYOFWK *EQ 5) THEN(CHGV
    34     1500                           VAR(&WEEKDAY) VALUE('Thursday')))
    35     1600-              ELSE       CMD(IF COND(&DAYOFWK *EQ 6) THEN(CHGV
    36     1700                           VAR(&WEEKDAY) VALUE('Friday')))
    37     1800-              ELSE       CMD(IF COND(&DAYOFWK *EQ 7) THEN(CHGV
    38     1900                           VAR(&WEEKDAY) VALUE('Saturday')))
                                                                        More...
Debug . . .

F3=End program   F6=Add/Clear breakpoint   F10=step    F11=Display variable
F12=Resume   F17=Watch variable   F18=Work with watch   F24=More keys
```

图 10-5　显示一个模块视图

在选择好后，用执行键，则给出选择的模块目标。

另一个方法是用 DISPLAY 调试命令，在调试命令行，写：

```
DISPLAY  MOUDLE  模块名
```

要显示的模块一定要在加到调试环境里的程序或服务程序中存在。

10.8.1 修改模块的视图

根据在生成 ILE 模块时规定的值，有几个视图可用。它们是：

根源码视图

清单视图

语句视图

可通过‘选择视图’显示来修改在‘显示模块源码’显示中给出的视图。可在‘显示模块源码’显示中用 F15 键来得到‘选择视图’的显示，如图 10-6。当前视图在窗口的顶部列出，其它的可用视图列在下边。在一个程序中的每个模块可有不同可用视图的分组，这是根据生成它时所用的调试选项决定的。

要选择一个视图，在 OPT 列写 1。

```
                              Display Module Source
.....................................................................
:                                 Select View                       :
:                                                                   :
:  Current View . . . :   CL Root Source                            :
:                                                                   :
:  Type option, press Enter.                                        :
:    1=Select                                                       :
:                                                                   :
:  Opt     View                                                     :
:          CL Root Source                                           :
:   1      CL Listing View                                          :
:                                                                   :
:                                                                   :
:                                                             Bottom  :
:  F12=Cancel                                                       :
:                                                                   :
:...................................................................:
                                                                More...
Debug . . .

F3=End Program   F6=Add/Clear breakpoint   F10=Step   F11=Display variable
F12=Resume   F17=Watch variable   F18=Work with watch   F24=More keys
```

图 10-6　修改模块的视图

选择好后，用执行键，则给出模块的视图。

10.8.2　设置和取消断点

可设置断点来在程序运行时的某点上停止程序。无条件断点在一个规定的语句停止程

序，条件断点在规定的语句中符合规定条件时停止程序。

在程序停止时，给出‘显示目标源码’显示，给出在发生断点的那行的相应模块的源码。这行是高亮显示，在此点上，可以给变量赋值，设置断点，及运行其它调试命令。

在使用断点之前，要知道以下特点：

在旁路断点时，比如 GOTO 语句，断点就不执行

在一个语句上设置断点时，在处理此语句前发生中断

在到达条件断点语句时，在处理此语句前计算与断点相关的条件表达式

通过调试命令来规定断点功能，这些功能包括：

加断点

取消断点

显示断点信息

在达到断点后，重新运行程序

10.8.3 设置和取消无条件断点

可用下列方法设置和取消非条件断点：

从‘显示模块源码’显示用 F6 键

从‘显示模块源码’显示用 F13 键

用 BREAK 设置断点

用 CLEAR 取消断点

设置和取消断点最简单的方法是从‘显示模块源码’显示中用 F6 键。要用 F6 键来设置无条件断点，把光标放在要设断点的那行，按 F6 键。要取消无条件断点，把光标放在要取消断点的那行，按 F6 键。对要设置无条件断点的重复上面的步骤。

注：如果要设断点的行不是一个可运行语句，那么断点没在下一个可运行语句上。

在设好断点后，用 F3 键结束，也可用 F21 键来从命令行调用程序。

在到达断点时，程序停止，再次出现‘显示模块源码’的显示。这时，可以给变量赋值、设断点和运行任何调试命令。

设置和取消无条件断点的另一个方法是用 BREAK 和 CLEAR 调试命令。在调试命令行输入：

```
BREAK       行号
CLEAK       行号
```

其中行号是在当前视图中要设置或取消断点的行号。

如果使用语句视图，没有显示的行号，那么在调试命令行输入：

```
BREAK       过程名/语句号
```

过程名是 CL 模块名，语句号（来自编译清单）是要停止的语句号码。

10.8.4 设置和取消条件断点

可用下列语句设置和取消条件断点：

处理断点显示

BREAK 调试命令

CLEAR 调试命令

10.8.4.1　使用处理断点显示

注：条件断点所用的关系操作符是：

<，>，=，<=，>=，< >

设置和取消条件断点的一个方法是通过‘处理模块断点’的显示。这个显示可由‘显示模块源码’显示中用 F13 键取得，如图 10—7 所示，要设置条件断点：

在 OPT 列写 1

在要设断点的 Line 列写行号

在 Condition 列写条件表达式

然后按执行键。如图 10—7 所示。

要取消一个条件断点，在要取消的 OPT 列写 4，按执行键，也可用此方法取消无条件断点。

```
                        Work with Module Breakpoints
                                                              System:   SYSTEM01
Program  . . . :   MYPGM              Library  . . . :   MYLIB
  Module . . . :     MYMOD            Type . . . . . :   *PGM

Type options, press Enter.
  1=Add   4=Clear

Opt     Line       Condition
 1      35____     &I=21_____________________
 _      _____      ___________________________
```

10-7　设置条件断点

重复前面步骤，即可设置和取消各个断点。

注：如果要设断点的行不是一个可运行语句，那么断点没在下一个可运行语句上。

在设置或取消所有断点后，用 F3 键结束，回到‘显示模块源码’显示中，然后再用 F3 键结束此显示，也可用 F21 键从命令行调用程序。当到达条件断点的语句时，在语句运行前计算有关的条件表达式。如果结果为假，程序继续运行，如果条件为真，程序停止，显示‘模块源码显示’，此时可以给变量赋值，设置更多断点及运行任何调试命令。

10.8.4.2　用 BREAK 和 CLEAR 命令

设置和取消条件断点的另一个方法是由 BREAK 和 CLEAR 命令。

BREAK 命令格式为：

BREAK　　行号　　WHEN　　表达式

行号是当前显示中要设置断点的行号。表达式是在到断点时要计算的条件表达式，支持的关系符号为>，<，<=，>=，=，< >

用非数字条件断点表达式，在做比较前，较短的表达式要用空格来填充，它在任何民族语言分类排序（NLSS）之前来做填充，详细信息请看 10.8.4.3。

要用 CLEAR 取消一个条件断点，在调试命令行上输入：

CLEAR　　行号

行号是当前显示的要取消断点的行号。

用语句视图，不显示行号，要设置断点，在调试命令行上用：

BREAK　　过程名/语句名　　WHEN　　表达式

10.8.4.3　民族语言分类排序（NLSS）

非数字条件断点表达式分为下列二类：

字符—8：每个字符 8 位

字符—16：每个字符 16 位（DBCS）

NLSS 仅提供字符—8 的非数字条件断点表达式。表 10—2 给出非数字条件断点表达式的可能组合。

字符—8 表达式的分类由源码调试程序所用的排序表是在 CRTCLMOD 或 CRTBNDCL 命令中的 SRTSEQ 参数规定的分类排序表。

如果涉及的分类排序表是*HEX，则不用分类排序表。这样，源码调试程序使用字符的十六进制值来决定分类顺序。另外，用特别的分类排序表是在比较前把权分配给每个字节的，字节之间的转入/转出字符不分配权。

注：分类排序表的名字是在编译时保存的，在调试时，源码调试程序用从编译程序保存的名访问分类排序表。如果在编译时这个表确定为不是*HEX 或*JOBRUN，那么在启动调试之前不能替换分类排序表。如果由于损坏或删除不能访问分类排序表，则用*HEX。

表 10—2　非数字条件断点表达式

类型	可能组合
字符—8	字符变量与字符变量比较 字符变量与字符字母比较(1) 字符变量与十六进制字母比较(2) 字符字母(1)与字符变量比较 字符字母(1)与十六进制字母(2)比较 十六进制字母(2)与字符变量(1)比较 十六进制字母(2)与字符字母(1)比较 十六进制字母(2)与十六进制字母(2)比较
字符—16	DBCS 字符变量与 DBCS 字符变量比较 DBCS 字符变量与图形字母(3)比较 DBCS 字符变量与十六进制字母(2)比较 图形字母(3)与 DBCS 字符变量比较 图形字母(3)与图形字母(3)比较 图形字母(3)与十六进制字母(2)比较

	十六进制字母(2)与 DBCS 字符变量比较 十六进制字母(2)与图形字母比较

(1)字符字母的格式为’ABC’。
(2)十六进制字母的格式为 X’十六进制数字’。
(3)图形的格式为 G’<so>DBCS 数据<si>’,其中<so>为转出符，<si>为转入符。

10.8.4.4　条件断点的例子

```
CL 说明        :   DCL     VAR(&CHAR1) TYPE(*CHAR) LEN(1)
                   DCL     VAR(&CHAR2) TYPE(*CHAR) LEN(2)
                   DCL     VAR(&DEC1) TYPE(*DEC) LEN(3 1)
                   DCL     VAR(&DEC2) TYPE(*DEC) LEN(4 1)


调试命令       :   BREAK 31 WHEN &DEC1 = 48.1

调试命令       :   BREAK 31 WHEN &DEC2 > &DEC1

调试命令       :   BREAK 31 WHEN &CHAR2 <> 'A'

注释           :   'A' is implicitly padded to
                   the right with one blank character before
                   the comparison is made.


调试命令       :   BREAK 31 WHEN %SUBSTR(&CHAR2 2 1) <= X'F1'

调试命令       :   BREAK 31 WHEN %SUBSTR(&CHAR2 1 1) >= &CHAR1

调试命令       :   BREAK 31 WHEN %SUBSTR(&CHAR2 1 1) < %SUBSTR(&CHAR2 2 1)
```

%SUBSTR 内部函数允许对字符串变量取子串，第一个的变量必须是串标识，第二个变量是起始位置，第三个是单字节数或双字节字符数，变量之间要有一个或多个空格。

10.8.5　取消所有的断点

可用 CLEAR PGM 调试命令，取消在‘显示模块源码’显示中出现的模块所在的程序中的所有条件及无条件断点，在调试命令行上输入 CLEAR　PGM，即从程序或服务程序的所有模块中取消断点。

10.9　单步通过程序目标

在遇到断点后，可以运行一定数量的程序语句，然后再停止程序，返回到‘显示模块源码’显示中，程序目标从停止的下一个条件语句运行，一个断点用来停止程序目标的运行。

可用下列方法一步步的通过程序：

在‘显示模块源码’显示中用 F10 键或 F22 键。

用 STEP 调试命令。

10.9.1 在显示源码显示中用 F10 或 F22 键

一次一个语句的单步通过程序最简单的方法是在‘显示模块源码’显示中用 F10 键或 F22 键，按 F10 键或 F22 键后，运行显示中给出的下一条语句，然后程序再次停止。

注：用 F10 键或 F22 键时不能规定单步通过的语句数，它只能完成一个单步。

另一方法是用 STEP 命令，它允许在一步中运行多个语句。

10.9.2 用 STEP 调试命令

用 STEP 调试命令，运行语句数的缺省值为 1，在调试命令行输入：

```
STEP        语句数
```

语句数是在程序再次停止前在下一步运行的程序语句数。例如，如果写

```
STEP        5
```

即程序运行下 5 条语句，然后程序再停止，出现‘显示模块源码’的显示。

10.9.3 步出和步进

在调试环境中遇到调用另外程序的 CALL 语句，你可以做以下事情：

步出被调用程序

步入被调用程序

如果选择步出，则 CALL 语句和被调用程序做为一个单步运行，在调用程序在下一步停止前运行被调用程序，这是单步方式的缺省值。

如果选择步进，那么被调用程序的每个语句做一个单步运行。如果运行下步停在被调用程序时，则在这点上程序停止，且在‘显示模块源码’显示中出现的是被调用程序。（被调用程序编译时有调试数据且你有适当的权限调试它）。

10.10 步出程序目标

可用下列方法步出程序目标：

在‘显示模块源码’显示中用 F10 键。

用 STEP OVER 调试命令。

10.10.1 用 F10 键

可在调试环境中的‘显示模块源码’显示中用 F10 键来步出一个被调用的程序目标。如果要运行的下一条语句是调用另外程序的 CALL 语句，用 F10 键会在调用程序再次停止之前运行完被调用程序。

10.10.2 用步出调试命令

可用 STEP OVER 调试命令来步出一个被调用程序。在调试命令行上写：

```
STEP        语句数        OVER
```

语句数是在程序再次停止之前在下一步想要运行的语句数。如果其中包括调用程序的 CALL 语句，则调试程序步出被调用的程序。

10.11 步入程序目标

可用下列方法步入程序目标：

在‘显示模块源码’显示中用 F22 键。

用 STEP INTO 调试命令。

10.11.1 用 F22 键

可在‘显示模块源码’显示中用 F22 键来步入被调用的程序。如果下一个语句是调用另外程序的 CALL，用 F22 键运行被调用程序的第一个语句，然后在‘显示模块源码’的显示中出现被调用程序。

注：被调用的程序必须有在‘显示模块源码’中给出的有关调试数据。

10.11.2 使用步入命令

可用 STEP INTO 调试命令来步入一个被调用程序，在调试命令行中输入：

```
STEP    语句数    INTO
```

语句数是在程序再次停止之前下一步要运行的程序的语句数。如果要运行的语句中包括一个调用其它程序的 CALL 语句，则调试程序步入被调用程序。在被调用程序中的每个语句都在单步中计算。如果单步在被调用程序中结束，那么在‘显示模块源码’中给出被调用程序。例如，用 STEP 5 INTO 命令，则运行程序的下五条语句，如果第三条语句是 CALL，那么运行调用程序的两条语句，然后运行被调用程序的三条语句。

10.12 显示变量

可用下列方法显示变量：

在‘显示模块源码’显示中用 F11 键。

用 EVAL 调试命令。

用在 EVAL 命令中变量的范围是由 QUAL 命令定义的，但不需要规定包括在 CL 模块中变量的范围，因为它们全是全程范围。

10.12.1 用 F11 键

要用 F11 键显示变量，把光标放在要显示的变量下，按 F11 键，在‘显示模块源码’的底部信息行出现变量的当前值。

```
                        Display Module Source

Program:   DSPWKDAY      Library:   MYLIB          Module:   DSPWKDAY
     4              DCL        VAR(&MSGTEXT) TYPE(*CHAR) LEN(20)
     5              CALL       PGM(WEEKDAY2) PARM(&DAYOFWK)
```

```
  6              IF         COND(&DAYOFWK *EQ 1) THEN(CHGVAR +
  7                           VAR(&WEEKDAY) VALUE('Sunday'))
  8              ELSE       CMD(IF COND(&DAYOFWK *EQ 2) THEN(CHGVAR +
  9                           VAR(&WEEKDAY) VALUE('Monday')))
 10              ELSE       CMD(IF COND(&DAYOFWK *EQ 3) THEN(CHGVAR +
 11                           VAR(&WEEKDAY) VALUE('Tuesday')))
 12              ELSE       CMD(IF COND(&DAYOFWK *EQ 4) THEN(CHGVAR +
 13                           VAR(&WEEKDAY) VALUE('Wednesday')))
 14              ELSE       CMD(IF COND(&DAYOFWK *EQ 5) THEN(CHGVAR +
 15                           VAR(&WEEKDAY) VALUE('Thursday')))
 16              ELSE       CMD(IF COND(&DAYOFWK *EQ 6) THEN(CHGVAR +
 17                           VAR(&WEEKDAY) VALUE('Friday')))
 18              ELSE       CMD(IF COND(&DAYOFWK *EQ 7) THEN(CHGVAR +
                                                                More...
Debug . . .

F3=End program   F6=Add/Clear breakpoint   F10=Step   F11=Display variable
F12=Resume    F17=Watch Variable    F18=Work with watch     F24=More keys
&DAYOFWK = 3.
```

图 10-8　用 F11 键显示变量

也可用 EVAL 调试命令来确定变量值，在调试命令行输入：

EVAL　变量名

如果变量的值能在一行上显示，则变量的值出现在信息行上，如果一行显示不完，则会在‘表达式赋值’显示中给出。

例如，要显示图 10-8 中变量&DAYOFWK 的值，输入 EVAL　&DAYOFWK，则在‘显示模块源码’显示中的信息行给出&DAYOFWK = 3。

如图 10-8 底部所示。

10.12.2　显示逻辑变量的例子

CL 变量　　:　　DCL　VAR(&LGL1)　TYPE(*LGL)　VALUE(‘1’)

调试命令　　:　　EVAL　&LGL1

结果　　:　　&LGL1 = ‘1’

10.12.3　显示字符变量的例子

```
CL 说明        :      DCL     VAR(&CHAR1) TYPE(*CHAR) LEN(10) VALUE('EXAMPLE')

调试命令       :      EVAL &CHAR1

结果           :      &CHAR1 = 'EXAMPLE    '


调试命令       :      EVAL %SUBSTR(&CHAR1 5 3)

结果           :      %SUBSTR(&CHAR1 5 3) = 'PLE'


调试命令       :      EVAL %SUBSTR(&CHAR1 7 4)

结果           :      %SUBSTR(&CHAR1 7 4) = 'E    '
```

10.12.4 显示十进变量的例子

```
CL 说明        :     DCL     VAR(&DEC1) TYPE(*DEC) LEN(4 1) VALUE(73.1)

CL 说明        :     DCL     VAR(&DEC2) TYPE(*DEC) LEN(3 1) VALUE(12.5)

调试命令       :      EVAL &DEC1

结果           :      &DEC1 = 073.1

调试命令       :      EVAL &DEC2

结果           :      &DEC2 = 12.5
```

10.12.5 显示十六进制变量的例子

可用 EVAL 命令显示变量的十六进制值，在调试命令行输入：

```
EVAL  变量名：  X   字节数
```

其中 X 指出是要用十六进制给出变量的值，字节数指出要显示的字节数。如果在 X 之后没有规定长度，那么变量的长度做显示的长度，最少可显示 16 字节。如果长度小于 16 字节，余下的字节用零填充，直到满 16 字节。

```
CL 说明        :   DCL     VAR(&CHAR1) TYPE(*CHAR) LEN(10) VALUE('ABC')
                   DCL     VAR(&CHAR2) TYPE(*CHAR) LEN(10) VALUE('DEF')

调试命令       :    EVAL &CHAR1:X 32
```

结果:

```
00000      C1C2C340 40404040 4040C4C5 C6404040 ABC        DEF
00010      40404040 00000000 00000000 00000000      ............
```

10.13　修改变量的值

可用 EVAL 命令和等号（=）一起来修改变量的值。

在 EVAL 命令中使用的变量范围是由 QUAL 定义的，但你不用规定在 CL 模块中使用变量的范围，因为它们全是全程范围。可以用 EVAL 给变量分配数值、字符和十六进制数据，来提供与变量定义匹配的值。

要修改变量的值，在调试行输入：

```
EVAL    变量名 = 值
```

例如：EVAL　&COUNTER = 3.0

就把变量的值改为 3.0，在信息行中给出&COUNTER = 3.0。

在给字符变量分配值时，注意下列原则：

如果原表达式的长度小于结果表达式的长度，要在结果中左对齐，余下部分填空格。

如果原表达式的长度大于结果表达式的长度，要在结果中左对齐，截断余下部分。

注：DBCS 变量可用下列方法之一分配：

另外一个 DBCS 变量

格式为 G'<so>DBCS 数据<si>'的图形字母

格式为 X'十六进制数字'的十六进制字母

10.13.1　修改逻辑变量的例子

```
CL 说明          :    DCL     VAR(&LGL1) TYPE(*LGL) VALUE('1')
                      DCL     VAR(&LGL2) TYPE(*LGL)


调试命令         :    EVAL &LGL1

结果             :    &LGL1 = '1'


调试命令         :    EVAL &LGL1 = X'F0'

结果             :    &LGL1 = X'F0' = '0'


调试命令         :    EVAL &LGL2 = &LGL1

结果             :    &LGL2 = &LGL1 = '0'
```

10.13.2 修改字符变量的例子

```
CL 说明        :   DCL     VAR(&CHAR1) TYPE(*CHAR) LEN(1) VALUE('A')
                   DCL     VAR(&CHAR2) TYPE(*CHAR) LEN(10)

调试命令       :   EVAL &CHAR1 = 'B'

结果           :   &CHAR1 = 'B' = 'B'

调试命令       :   EVAL &CHAR1 = X'F0F1F2F3'

结果           :   &CHAR1 = 'F0F1F2F3' = '0'

调试命令       :   EVAL &CHAR2 = 'ABC'

结果           :   &CHAR2 = 'ABC' = 'ABC       '

调试命令       :   EVAL %SUBSTR(CHAR2 1 2) = %SUBSTR(&CHAR2 3 1)

结果           :   %SUBSTR(CHAR2 1 2) = %SUBSTR(&CHAR2 3 1) = 'C '

注释           :   Variable &CHAR contains 'C C       '
```

10.13.3 修改十进制变量的例子

```
CL 说明        :    DCL    VAR(&DEC1) TYPE(*DEC) LEN(3 1) VALUE(73.1)
                    DCL    VAR(&DEC2) TYPE(*DEC) LEN(2 1) VALUE(3.1)

调试命令       :    EVAL &DEC1 = 12.3

结果           :    &DEC1 = 12.3 = 12.3

调试命令       :    EVAL &DEC1 = &DEC2

结果           :    &DEC1 = &DEC2 = 03.1
```

10.14 变量属性的例子

ATTR 调试命令让你显示变量的属性，属性是记录在表 10—1 和表 10—2 的调试符号表

中的变量大小（字节数）和类型，下面是 ATTR 的例子：

```
CL 说明        :   DCL     VAR(&CHAR2) TYPE(*CHAR) LEN(10)

调试命令       :   ATTR &CHAR2

结果           :   TYPE = FIXED LENGTH STRING, LENGTH = 10 BYTES


CL 说明        :   DCL     VAR(&DEC) TYPE(*DEC) LEN(3 1)

调试命令       :   ATTR &DEC

结果           :   TYPE = PACKED(3,1), LENGTH = 2 BYTES
```

10.15　与一个变量、表达式或命令等同的名字

可用 EQUATE 调试命令把一个名字等同于一个变量、表达式或调试命令以便速记用，可把此名用在另外的表达式中。

这时，名字和值是由等同之前的表达式确定，这个名字在调试环境结束或取消后不能用。

要把变量表达式或调试命令等同一个名字，在调试命令行上输入：

```
EQUATE      速记名  定义
```

速记名是要等同的名字，定义是要等同的变量、表达式或命令。例如，要定义一个 DC 的名字等于&COUNTER 变量的内容，在调试命令行上输入：

```
EQUATE  DC  EVAL  &COUNTER
```

此后，当在调试命令行上输入 DC，就运行 EVAL &COUNTER 命令，能在 EQUATE 命令中输入的最多字符数为 144。如果不支持某个定义而在前面的 EQUATE 命令定义了名字，则取消前面的定义，如果前面没定义名字，则给出错误信息。

要看用 EQUATE 定义的名字，在调试命令行上输入：

```
DISPLAY  EQUATE
```

在‘表达式赋值’显示中列出活动的名字。

10.16　ILE CL 的源码调试民族语言支持

在处理 ILE　CL 的源码调试民族语言支持时，存在下列情况：

在‘显示模块源码’显示中给出一个视图时，源码调试程序把所有数据转换成调试作业的 CCSID。

在给变量分配字母时，源码调试程序对引号字母（比如‘abc’），不做 CCSID 转换，而且引号内的字母区分大小写。

10.16.1 处理*SOURCE 视图

仅在处理 CL 根源码视图时，下列条件为真：

如果源文件的 CCSID 不同于模块的 CCSID，则源码调试程序不识别包含特殊字符（#、@、$）的 CL 标识。

模块的 CCSID 可用 DSPMOD 命令来找到，如果需要处理 CL 根源码视图而源文件与模块的 CCSID 不同，可采取下列动作之一：

保证 CL 源文件的 CCSID 与编译时作业的 CCSID 相同。

把编译时作业的 CCSID 改为 65535 然后再编译。

如果前两个作法都不可能，使用 CL 清单视图。

有关调试中的 NLS 限制，请看 ILE 概念一书。

10.16.2 在调试时使用 COPY、SAVE、RESTORE、CRTDUPOBJ 和 CHKOBJITG

在用 CL 命令规定库和程序时，断点或单步可临时从已调试的程序中移出。在运行完 CL 命令后，断点或单步可以重存回来。在移出断点或单步时，会在作业日志中记录 CPD190A 信息，在重存时也会记录另一个 CPD190A 信息，下表是能引起断点或单步临时移出的 CL 命令。

CHKOBJITG	CPY	CPROBJ	RSTLIB	SAVLIB
	CPYLIB	CRTDUPOBJ	RSTOBJ	SAVOBJ
				SAVSYS
				SAVCHGOBJ

注：在程序中处理 CL 命令时，在发出 BREAK 或 STEP 命令，要收到 CPF7102 错误信息。

附录：CL 命令表

1 ADDACC (Add Access Code) Command
2 ADDAJE (Add Autostart Job Entry) Command
3 ADDALRACNE (Add Alert Action Entry) Command
4 ADDALRD (Add Alert Description) Command
5 ADDALRSLTE (Add Alert Selection Entry) Command
6 ADDAUTLE (Add Authorization List Entry) Command
7 ADDBKP (Add Breakpoint) Command
8 ADDBNDDIRE (Add Binding Directory Entry) Command
9 ADDCCTRTE (Add Circuit Route) Command
10 ADDCCTSRV (Add Circuit Service) Command
11 ADDCFGLE (Add Configuration List Entries) Command
12 ADDCMNE (Add Communications Entry) Command
13 ADDCNNLE (Add Connection List Entry) Command
14 ADDCOMSNMP (Add Community for SNMP) Command

ADDDIRE (Add Directory Entry) Command
ADDDIRSHD (Add Directory Shadow System) Command
ADDDLOAUT (Add Document Library Object Authority) Command
ADDDSTLE (Add Distribution List Entry) Command
ADDDSTQ (Add Distribution Queue) Command
ADDDSTRTE (Add Distribution Route) Command
ADDDSTSYSN (Add Distribution Secondary System Name) Command
ADDDTADFN (Add Data Definition) Command
ADDEMLCFGE (Add Emulation Configuration Entry) Command
ADDENVVAR (Add Environment Variable) Command
ADDEWCBCDE (Add Extended Wireless Controller Bar Code Entry) Command
ADDEWCM (Add Extended Wireless Controller Member) Command
ADDEWCPTCE (Add Extended Wireless Controller PTC Entry) Command
ADDEWLM (Add Extended Wireless Line Member) Command
ADDEXITPGM (Add Exit Program) Command
ADDFNTTBLE (Add Font Table Entry) Command
ADDICFDEVE (Add Intersystem Communications Function Program Device Entry) Co
mmand
ADDIPIADR (Add IP over IPX Address) Command
ADDIPIIFC (Add IP over IPX Interface) Command
ADDIPIRTE (Add IP over IPX Route) Command
ADDIPSIFC (Add IP over SNA Interface) Command
ADDIPSLOC (Add IP over SNA location entry) Command
ADDIPSRTE (Add IP over SNA Route) Command
ADDIPXCCT (Add IPX Circuit) Command
ADDJOBQE (Add Job Queue Entry) Command
ADDJOBSCDE (Add Job Schedule Entry) Command
ADDLANADPI (Add Local Area Network Adapter Information) Command
ADDLFM (Add Logical File Member) Command
ADDLIBLE (Add Library List Entry) Command
ADDLICKEY (Add License Key Information) Command
ADDLNK (Add Link) Command
ADDMFS (Add Mounted File System) Command
ADDMSGD (Add Message Description) Command
ADDNCK (Add Nickname) Command
ADDNETJOBE (Add Network Job Entry) Command
ADDNETTBLE (Add Network Table Entry) Command
ADDNODLE (Add Node List Entry) Command
ADDNWSSTGL (Add Network Server Storage Link) Command
ADDOPTCTG (Add Optical Cartridge) Command
ADDOPTSVR (Add Optical Server) Command
ADDPCLTBLE (Add Protocol Table Entry) Command
ADDPEXDFN (Add Performance Explorer Definition) Command
ADDPFCST (Add Physical File Constraint) Command

ADDPFM (Add Physical File Member) Command
ADDPFRCOL (Add Performance Collection) Command
ADDPFTRG (Add Physical File Trigger) Command
ADDPGM (Add Program) Command
ADDPJE (Add Prestart Job Entry) Command
ADDPRBACNE (Add Problem Action Entry) Command
ADDPRBSLTE (Add Problem Selection Entry) Command
ADDRDBDIRE (Add Relational Database Directory Entry) Command
ADDREXBUF (Add REXX Buffer) Command
ADDRMTDFN (Add Remote Definition) Command
ADDRMTSVR (Add Remote Server) Command
ADDRPYLE (Add Reply List Entry) Command
ADDRTGE (Add Routing Entry) Command
ADDSCHIDXE (Add Search Index Entry) Command
ADDSNILOC (Add SNA over IPX Location) Command
ADDSOCE (Add Sphere of Control Entry) Command
ADDSRVTBLE (Add Service Table Entry) Command
ADDSVRAUTE (Add Server Authentication Entry) Command
ADDTAPCTG (Add Tape Cartridge) Command
ADDTCPHTE (Add TCP/IP Host Table Entry) Command
ADDTCPIFC (Add TCP/IP Interface) Command
ADDTCPPORT (Add TCP/IP Port Restriction) Command
ADDTCPRSI (Add TCP/IP Remote System Information) Command
ADDTCPRTE (Add TCP/IP Route) Command
ADDTRC (Add Trace) Command
ADDUSFCNNE (Add Ultimedia System Facilities Connection Entry) Command
ADDUSFDEVE (Add Ultimedia System Facilities Device Entry) Command
ADDUSFSVRE (Add Ultimedia System Facilities Server Entry) Command
ADDWSE (Add Work Station Entry) Command
ALCOBJ (Allocate Object) Command
ANSLIN (Answer Line) Command
ANSQST (Answer Questions) Command
ANZDFTPWD (Analyze Default Passwords) Command
ANZPRB (Analyze Problem) Command
ANZPRFACT (Analyze Profile Activity) Command
ANZQRY (Analyze Query) Command
ANZUSROBJ (Analyze User Objects) Command
APING Command
APYJRNCHG (Apply Journaled Changes) Command
APYPTF (Apply Program Temporary Fix) Command
AREXEC Command
ASKQST (Ask Question) Command
BCHJOB (Batch Job) Command
CALL (Call Program) Command

CALLPRC (Call Bound Procedure) Command
CD (Change Current Directory) Command
CFGDEVMLB (Configure Device Media Library) Command
CFGDSTSRV (Configure Distribution Services) Command
CFGIPI (Configure IP over IPX) Command
CFGIPS (Configure IP over SNA Interface) Command
CFGIPX (Configure IPX) Command
CFGSYSSEC (Configure System Security) Command
CFGTCP (Configure TCP/IP) Command
CFGTCPAPP (Configure TCP/IP Applications) Command
CFGTCPBP (Configure TCP/IP BOOTP) Command
CFGTCPPTP (Configure Point-to-Point TCP/IP) Command
CFGTCPRTD (Configure TCP/IP RouteD) Command
CFGTCPSNMP (Configure TCP/IP SNMP) Command
CHDIR (Change Current Directory) Command
CHGACGCDE (Change Accounting Code) Command
CHGACTPRFL (Change Active Profile List) Command
CHGACTSCDE (Change Activation Schedule Entry) Command
CHGAJE (Change Autostart Job Entry) Command
CHGALRACNE (Change Alert Action Entry) Command
CHGALRD (Change Alert Description) Command
CHGALRSLTE (Change Alert Selection Entry) Command
CHGALRTBL (Change Alert Table) Command
CHGAUD (Change Auditing Value) Command
CHGAUT (Change Authority) Command
CHGAUTLE (Change Authorization List Entry) Command
CHGBCKUP (Change Backup Options) Command
CHGBPA (Change BOOTP Attributes) Command
CHGCCTRTE (Change Circuit Route) Command
CHGCCTSRV (Change Circuit Service) Command
CHGCFGL (Change Configuration List) Command
CHGCFGLE (Change Configuration List Entry) Command
CHGCLS (Change Class) Command
CHGCDEFNT (Change Coded Font) Command
CHGCMD (Change Command) Command
CHGCMDDFT (Change Command Default) Command
CHGCMNE (Change Communications Entry) Command
CHGCNNL (Change Connection List) Command
CHGCNNLE (Change Connection List Entry) Command
CHGCOMSNMP (Change Community for SNMP) Command
CHGCOSD (Change Class-of-Service Description) Command
CHGCRQD (Change Change Request Description) Command
CHGCSI (Change Communications Side Information) Command
CHGCTLAPPC (Change Controller Description (APPC)) Command

CHGCTLASC (Change Controller Description (Async)) Command
CHGCTLBSC (Change Controller Description (BSC)) Command
CHGCTLFNC (Change Controller Description (Finance)) Command
CHGCTLHOST (Change Controller Description (SNA Host)) Command
CHGCTLLWS (Change Controller Description (Local Work Station)) Command
CHGCTLNET (Change Controller Description (Network)) Command
CHGCTLRTL (Change Controller Description (Retail)) Command
CHGCTLRWS (Change Controller Description (Remote Work Station)) Command
CHGCTLTAP (Change Controller Description (TAPE)) Command
CHGCTLVWS (Change Controller Description (Virtual Work Station)) Command
CHGCURDIR (Change Current Directory) Command
CHGCURLIB (Change Current Library) Command
CHGDBG (Change Debug) Command
CHGDDMF (Change Distributed Data Management File) Command
CHGDDMTCPA (Change DDM TCP/IP Attributes) Command
CHGDEVAPPC (Change Device Description (APPC)) Command
CHGDEVASC (Change Device Description (Async)) Command
CHGDEVBSC (Change Device Description (BSC)) Command
CHGDEVDKT (Change Device Description (Diskette)) Command
CHGDEVDSP (Change Device Description (Display)) Command
CHGDEVFNC (Change Device Description (Finance)) Command
CHGDEVHOST (Change Device Description (SNA Host)) Command
CHGDEVINTR (Change Device Description (Intrasystem)) Command
CHGDEVMLB (Change Device Description (Media Library)) Command
CHGDEVNET (Change Device Description (Network)) Command
CHGDEVOPT (Change Device Description (Optical)) Command
CHGDEVPRT (Change Device Description (Printer)) Command
CHGDEVRTL (Change Device Description (Retail)) Command
CHGDEVSNPT (Change Device Description (SNA Pass-Through)) Command
CHGDEVSNUF (Change Device Description (SNUF)) Command
CHGDEVTAP (Change Device Description (Tape)) Command
CHGDHCPA (Change DHCP Attributes) Command
CHGDIRE (Change Directory Entry) Command
CHGDIRSHD (Change Directory Shadow System) Command
CHGDKTF (Change Diskette File) Command
CHGDLOAUD (Change Document Library Object Audit) Command
CHGDLOAUT (Change Document Library Object Authority) Command
CHGDLOOWN (Change Document Library Object Owner) Command
CHGDLOPGP (Change Document Library Object Primary Group) Command
CHGDOCD (Change Document Description) Command
CHGDSPF (Change Display File) Command
CHGDSTA (Change Distribution Attributes) Command
CHGDSTD (Change Distribution Description) Command
CHGDSTL (Change Distribution List) Command

CHGDSTPWD (Change Dedicated Service Tools Password) Command
CHGDSTQ (Change Distribution Queue) Command
CHGDSTRTE (Change Distribution Route) Command
CHGDTAARA (Change Data Area) Command
CHGEMLCFGE (Change Emulation Configuration Entry) Command
CHGENVVAR (Change Environment Variable) Command
CHGEWCBCDE (Change Extended Wireless Controller Bar Code Entry) Command
CHGEWCM (Change Extended Wireless Controller Member) Command
CHGEWCPTCE (Change Extended Wireless Controller PTC Entry) Command
CHGEWLM (Change Extended Wireless Line Member) Command
CHGEXPSCDE (Change Expiration Schedule Entry) command
CHGFNTRSC (Change Font Resource) Command
CHGFNTTBLE (Change Font Table Entry) Command
CHGFTR (Change Filter) Command
CHGGRPA (Change Group Attributes) Command
CHGHLLPTR (Change High-Level Language Pointer) Command
CHGICFDEVE (Change ICF Program Device Entry) Command
CHGICFF (Change Intersystem Communications Function File) Command
CHGIPIADR (Change IP over IPX Address) Command
CHGIPIIFC (Change IP over IPX Interface) Command
CHGIPLA (Change IPL Attributes) Command
CHGIPSIFC (Change IP over SNA Interface) Command
CHGIPSLOC (Change IP over SNA Location) Command
CHGIPSTOS (Change IP over SNA Type of Service) Command
CHGIPXCCT (Change IPX Circuit) Command
CHGIPXD (Change IPX Description) Command
CHGJOB (Change Job) Command
CHGJOBD (Change Job Description) Command
CHGJOBQE (Change Job Queue Entry) Command
CHGJOBSCDE (Change Job Schedule Entry) Command
CHGJRN (Change Journal) Command
CHGKBDMAP (Change Keyboard Map) Command
CHGLANADPI (Change LAN Adapter Information) Command
CHGLF (Change Logical File) Command
CHGLFM (Change Logical File Member) Command
CHGLIB (Change Library) Command
CHGLIBL (Change Library List) Command
CHGLICINF (Change License Information) Command
CHGLINASC (Change Line Description (Async)) Command
CHGLINBSC (Change Line Description (BSC)) Command
CHGLINDDI (Change Line Description (DDI Network)) Command
CHGLINETH (Change Line Description (Ethernet)) Command
CHGLINFAX (Change Line Description (Fax)) Command
CHGLINFR (Change Line Description (Frame Relay Network)) Command

CHGLINIDLC (Change Line Description (IDLC)) Command
CHGLINNET (Change Line Description (Network)) Command
CHGLINPPP (Change Line Description (PPP)) Command
CHGLINSDLC (Change Line Description (SDLC)) Command
CHGLINTDLC (Change Line Description (TDLC)) Command
CHGLINTRN (Change Line Description (Token-Ring Network)) Command
CHGLINWLS (Change Line Description (Wireless)) Command
CHGLINX25 (Change Line Description (X.25)) Command
CHGMNU (Change Menu) Command
CHGMOD (Change Module) Command
CHGMODD (Change Mode Description) Command
CHGMSGD (Change Message Description) Command
CHGMSGF (Change Message File) Command
CHGMSGQ (Change Message Queue) Command
CHGM36 (Change AS/400 Advanced 36 Machine) Command
CHGM36CFG (Change AS/400 Advanced 36 Machine Configuration) Command
CHGNCK (Change Nickname) Command
CHGNETA (Change Network Attributes) Command
CHGNETJOBE (Change Network Job Entry) Command
CHGNFSEXP (Change Network File System Export) Command
CHGNODGRPA (Change Node Group Attributes) Command
CHGNTBD (Change NetBIOS Description) Command
CHGNWIATM (Change Network Interface (ATM Network)) Command
CHGNWIFR (Change Network Interface (Frame Relay Network)) Command
CHGNWIISDN (Change Network Interface Description for ISDN) Command
CHGNWSA (Change Network Server Attributes) Command
CHGNWSALS (Change Network Server Alias) Command
CHGNWSD (Change Network Server Description) Command
CHGNWSUSRA (Change Network Server User Attributes) Command
CHGOBJAUD (Change Object Auditing) Command
CHGOBJD (Change Object Description) Command
CHGOBJOWN (Change Object Owner) Command
CHGOBJPGP (Change Object Primary Group) Command
CHGOPTA (Change Optical Attributes) Command
CHGOPTVOL (Change Optical Volume) Command
CHGOUTQ (Change Output Queue) Command
CHGOWN (Change Owner) Command
CHGPDGPRF (Change Print Descriptor Group Profile) Command
CHGPEXDFN (Change Performance Explorer Definition) Command
CHGPF (Change Physical File) Command
CHGPFCST (Change Physical File Constraint) Command
CHGPFM (Change Physical File Member) Command
CHGPFRCOL (Change Performance Collection) Command
CHGPGM (Change Program) Command

CHGPGMVAR (Change Program Variable) Command
CHGPGP (Change Primary Group) Command
CHGPJ (Change Prestart Job) Command
CHGPJE (Change Prestart Job Entry) Command
CHGPRB (Change Problem) Command
CHGPRBACNE (Change Problem Action Entry) Command
CHGPRBSLTE (Change Problem Selection Entry) Command
CHGPRF (Change Profile) Command
CHGPRTF (Change Printer File) Command
CHGPSFCFG (Change Print Services Facility Configuration) Command
CHGPTR (Change Pointer) Command
CHGPWD (Change Password) Command
CHGPWRSCD (Change Power On/Off Schedule) Command
CHGPWRSCDE (Change Power On/Off Schedule Entry) Command
CHGQRYA (Change Query Attributes) Command
CHGQSTDB (Change Question-and-Answer Database) Command
CHGRCYAP (Change Recovery for Access Paths) Command
CHGRDBDIRE (Change Relational Database Directory Entry) Command
CHGRMTDFN (Change Remote Definition) Command
CHGRPYLE (Change Reply List Entry) Command
CHGRTDA (Change RouteD Attributes) Command
CHGRTGE (Change Routing Entry) Command
CHGRWSPWD (Change RWS Controller Password) Command
CHGSAVF (Change Save File) Command
CHGSBSD (Change Subsystem Description) Command
CHGSCHIDX (Change Search Index) Command
CHGSECA (Change Security Attributes) Command
CHGSECAUD (Change Security Auditing Values) Command
CHGSHRPOOL (Change Shared Storage Pool) Command
CHGSNILOC (Change SNA over IPX Location) Command
CHGSNMPA (Change SNMP Attributes) Command
CHGSPLFA (Change Spooled File Attributes) Command
CHGSRCPF (Change Source Physical File) Command
CHGSRVA (Change Service Attributes) Command
CHGSRVPGM (Change Service Program) Command
CHGSSNMAX (Change Session Maximum) Command
CHGSVRAUTE (Change Server Authentication Entry) Command
CHGSYSDIRA (Change System Directory Attributes) Command
CHGSYSJOB (Change System Job) Command
CHGSYSLIBL (Change System Library List) Command
CHGSYSVAL (Change System Value) Command
CHGS36 (Change System/36) Command
CHGS36A (Change System/36 Attributes) Command
CHGS36MSGL (Change System/36 Message List) Command

CHGS36PGMA (Change System/36 Program Attributes) Command
CHGS36PRCA (Change System/36 Procedure Attributes) Command
CHGS36SRCA (Change System/36 Source Attributes) Command
CHGTAPCTG (Change Tape Cartridge) Command
CHGTAPF (Change Tape File) Command
CHGTCPA (Change TCP/IP Attributes) Command
CHGTCPDMN (Change TCP/IP Domain) Command
CHGTCPHTE (Change TCP/IP Host Table Entry)
CHGTCPIFC (Change TCP/IP Interface) Command
CHGTCPRTE (Change TCP/IP Route) Command
CHGTFTPA (Change TFTP Server Attributes) Command Syntax
CHGUSFDEVE (Change Ultimedia System Facilities Device Entry) Command
CHGUSRAUD (Change User Audit) Command
CHGUSRPRTI (Change User Print Information) Command
CHGUSRPRF (Change User Profile) Command
CHGVAR (Change Variable) Command
CHGWSE (Change Work Station Entry) Command
CHGWTR (Change Writer) Command
CHKCMNTRC (Check Communications Trace) Command
CHKDKT (Check Diskette) Command
CHKDLO (Check Document Library Object) Command
CHKIGCTBL (Check DBCS Font Table) Command
CHKIN (Check In) Command
CHKOBJ (Check Object) Command
CHKOBJITG (Check Object Integrity) Command
CHKOUT (Check Out) Command
CHKPRDOPT (Check Product Option) Command
CHKPWD (Check Password) Command
CHKRCDLCK (Check Record Locks) Command
CHKTAP (Check Tape) Command
CLOF (Close File) Command
CLRDKT (Clear Diskette) Command
CLRJOBQ (Clear Job Queue) Command
CLRLIB (Clear Library) Command
CLRMSGQ (Clear Message Queue) Command
CLROUTQ (Clear Output Queue) Command
CLRPFM (Clear Physical File Member) Command
CLRPOOL (Clear Pool) Command
CLRSAVF (Clear Save File) Command
CLRTRCDTA (Clear Trace Data) Command
CMPJRNIMG (Compare Journal Images) Command
CMPPTFLVL (Compare PTF Level) Command
COMMIT (Commit) Command
COPY (Copy) Command

COPYRIGHT (COPYRIGHT) Command
CPROBJ (Compress Object) Command
CPY (Copy) Command
CPYCFGL (Copy Configuration List) Command
CPYDOC (Copy Document) Command
CPYF (Copy File) Command
CPYFRMDIR (Copy From Directory) Command
CPYFRMDKT (Copy from Diskette) Command
CPYFRMPCFF (Copy From PCF File) Command
CPYFRMQRYF (Copy From Query File) Command
CPYFRMSTMF (Copy from Stream File) Command
CPYFRMTAP (Copy from Tape) Command
CPYIGCTBL (Copy DBCS Font Table) Command
CPYLIB (Copy Library) Command
CPYOPT (Copy Optical) Command
CPYPTF (Copy Program Temporary Fix) Command
CPYSPLF (Copy Spooled File) Command
CPYSRCF (Copy Source File) Command
CPYTODIR (Copy To Directory) Command
CPYTODKT (Copy to Diskette) Command
CPYTOPCFF (Copy To PCF File) Command
CPYTOSTMF (Copy to Stream File) Command
CPYTOTAP (Copy to Tape) Command
CRTALRTBL (Create Alert Table) Command
CRTAUTHLR (Create Authority Holder) Command
CRTAUTL (Create Authorization List) Command
CRTBNDCL (Create Bound Control Language Program) Command
CRTBNDDIR (Create Binding Directory) Command
CRTCFGL (Create Configuration List) Command
CRTCLMOD (Create Control Language Module) Command
CRTCLPGM (Create Control Language Program) Command
CRTCLS (Create Class) Command
CRTCMD (Create Command) Command
CRTCNNL (Create Connection List) Command
CRTCOSD (Create Class-of-Service Description) Command
CRTCRQD (Create Change Request Description) Command
CRTCSI (Create Communications Side Information) Command
CRTCTLAPPC (Create Controller Description (APPC)) Command
CRTCTLASC (Create Controller Description (Async)) Command
CRTCTLBSC (Create Controller Description (BSC)) Command
CRTCTLFNC (Create Controller Description (Finance)) Command
CRTCTLHOST (Create Controller Description (SNA Host)) Command
CRTCTLLWS (Create Controller Description (Local Work Station)) Command
CRTCTLNET (Create Controller Description (Network)) Command

CRTCTLRTL (Create Controller Description (Retail)) Command
CRTCTLRWS (Create Controller Description (Remote Work Station))
CRTCTLTAP (Create Controller Description (Tape)) Command
CRTCTLVWS (Create Controller Description (Virtual Work Station)) Command
CRTDDMF (Create Distributed Data Management File) Command
CRTDEVAPPC (Create Device Description (APPC)) Command
CRTDEVASC (Create Device Description (Async)) Command
CRTDEVBSC (Create Device Description (BSC)) Command
CRTDEVDKT (Create Device Description (Diskette)) Command
CRTDEVDSP (Create Device Description (Display)) Command
CRTDEVFNC (Create Device Description (Finance)) Command
CRTDEVHOST (Create Device Description (SNA Host)) Command
CRTDEVINTR (Create Device Description (Intrasystem)) Command
CRTDEVMLB (Create Device Description (Media Library)) Command
CRTDEVNET (Create Device Description (Network)) Command
CRTDEVOPT (Create Device Description (Optical)) Command
CRTDEVPRT (Create Device Description (Printer)) Command
CRTDEVRTL (Create Device Description (Retail)) Command
CRTDEVSNUF (Create Device Description (SNUF)) Command
CRTDEVSNPT (Create Device Description (SNA Pass-Through)) Command
CRTDEVTAP (Create Device Description (Tape)) Command
CRTDIR (Create Directory) Command
CRTDKTF (Create Diskette File) Command
CRTDOC (Create Document) Command
CRTDSPF (Create Display File) Command
CRTDSTL (Create Distribution List) Command
CRTDTAARA (Create Data Area) Command
CRTDTADCT (Create a Data Dictionary) Command
CRTDTAQ (Create Data Queue) Command
CRTDUPOBJ (Create Duplicate Object) Command
CRTEDTD (Create Edit Description) Command
CRTFLR (Create Folder) Command
CRTFNTTBL (Create Font Table) Command
CRTFNTRSC (Create Font Resources) Command
CRTFORMDF (Create Form Definition) Command
CRTFTR (Create Filter) Command
CRTGSS (Create Graphics Symbol Set) Command
CRTICFF (Create Intersystem Communications Function File) Command
CRTIGCDCT (Create DBCS Conversion Dictionary) Command
CRTIPXD (Create IPX Description) Command
CRTJOBD (Create Job Description) Command
CRTJOBQ (Create Job Queue) Command
CRTJRN (Create Journal) Command
CRTJRNRCV (Create Journal Receiver) Command

CRTLF (Create Logical File) Command
CRTLIB (Create Library) Command
CRTLINASC (Create Line Description (Async)) Command
CRTLINBSC (Create Line Description (BSC)) Command
CRTLINDDI (Create Line Description (DDI Network)) Command
CRTLINETH (Create Line Description (Ethernet)) Command
CRTLINFAX (Create Line Description (Fax)) Command
CRTLINFR (Create Line Description (Frame Relay Network)) Command
CRTLINIDLC (Create Line Description (IDLC)) Command
CRTLINNET (Create Line Description (Network)) Command
CRTLINPPP (Create Line Description (PPP)) Command
CRTLINSDLC (Create Line Description (SDLC)) Command
CRTLINTDLC (Create Line Description (TDLC)) Command
CRTLINTRN (Create Line Description (Token-Ring Network)) Command
CRTLINWLS (Create Line Description (Wireless)) Command
CRTLINX25 (Create Line Description (X.25))
CRTLOCALE (Create Locale) Command
CRTMNU (Create Menu) Command
CRTMODD (Create Mode Description) Command
CRTMSGF (Create Message File) Command
CRTMSGFMNU (Create Message File Menu) Command
CRTMSGQ (Create Message Queue) Command
CRTM36 (Create AS/400 Advanced 36 Machine) Command
CRTM36CFG (Create AS/400 Advanced 36 Machine Configuration) Command
CRTNODGRP (Create Node Group) Command
CRTNODL (Create Node List) Command
CRTNTBD (Create NetBIOS Description) Command
CRTNWIATM (Create Network Interface (ATM Network)) Command
CRTNWIFR (Create Network Interface (Frame Relay Network)) Command
CRTNWIISDN (Create Network Interface Description for ISDN) Command
CRTNWSALS (Create Network Server Alias) Command
CRTNWSD (Create Network Server Description) Command
CRTNWSSTG (Create Network Server Storage Space) Command
CRTOUTQ (Create Output Queue) Command
CRTOVL (Create Overlay) Command
CRTPAGDFN (Create Page Definition) Command
CRTPAGSEG (Create Page Segment) Command
CRTPDG (Create Print Descriptor Group) Command
CRTPF (Create Physical File) Command
CRTPGM (Create Program) Command
CRTPNLGRP (Create Panel Group) Command
CRTPRTF (Create Printer File) Command
CRTPSFCFG (Create Print Services Facility Configuration) Command
CRTQMFORM (Create Query Management Form) Command

CRTQMQRY (Create Query Management Query) Command
CRTQSTDB (Create Question-and-Answer Database) Command
CRTQSTLOD (Create Question-and-Answer Load) Command
CRTSAVF (Create Save File) Command
CRTSBSD (Create Subsystem Description) Command
CRTSCHIDX (Create Search Index) Command
CRTSPADCT (Create Spelling Aid Dictionary) Command
CRTSQLPKG (Create Structured Query Language Package) Command
CRTSRCPF (Create Source Physical File) Command
CRTSRVPGM (Create Service Program) Command
CRTS36DSPF (Create System/36 Display File) Command
CRTS36MNU (Create System/36 Menu) Command
CRTS36MSGF (Create System/36 Message File) Command
CRTTAPCGY (Create Tape Category) Command
CRTTAPF (Create Tape File) Command
CRTTBL (Create Table) Command
CRTUDFS (Create User-Defined File System) Command
CRTUSRPRF (Create User Profile) Command
CRTVLDL (Create Validation List) Command
CRTWSCST (Create Work Station Customizing Object) Command
CVTCLSRC (Convert CL Source) Command
CVTDAT (Convert Date) Command
CVTDLSNAM (Convert Document Library Services Name) Command
CVTEDU (Convert Education) Command
CVTIPSIFC (Convert IP over SNA Interface) Command
CVTIPSLOC (Convert IP over SNA Location Entry) Command
CVTOPTBKU (Convert Optical Backup) Command
CVTPFRDTA (Convert Performance Data) Command
CVTPFRTHD (Convert Performance Thread Data) Command
CVTRPCSRC (Convert RPC Source) Command
CVTTCPCL (Convert TCP/IP CL Source) Command
DATA (Data) Command
DCL (Declare CL Variable) Command
DCLF (Declare File) Command
DCPOBJ (Decompress Object) Command
DEL (Remove Link) Command
DLCOBJ (Deallocate Object) Command
DLTALR (Delete Alert) Command
DLTALRTBL (Delete Alert Table) Command
DLTAPARDTA (Delete APAR Data) Command
DLTAUTHLR (Delete Authority Holder) Command
DLTAUTL (Delete Authorization List) Command
DLTBNDDIR (Delete Binding Directory) Command
DLTCFGL (Delete Configuration List) Command

DLTCLS (Delete Class) Command
DLTCMD (Delete Command) Command
DLTCMNTRC (Delete Communications Trace) Command
DLTCNNL (Delete Connection List) Command
DLTCOSD (Delete Class-of-Service Description) Command
DLTCRQD (Delete Change Request Description) Command
DLTCSI (Delete Communications Side Information) Command
DLTCTLD (Delete Controller Description) Command
DLTDEVD (Delete Device Description) Command
DLTDKTLBL (Delete Diskette Label) Command
DLTDLO (Delete Document Library Object) Command
DLTDOCL (Delete Document List) Command
DLTDST (Delete Distribution) Command
DLTDSTL (Delete Distribution List) Command
DLTDTAARA (Delete Data Area) Command
DLTDTADCT (Delete Data Dictionary) Command
DLTDTAQ (Delete Data Queue) Command
DLTEDTD (Delete Edit Description) Command
DLTF (Delete File) Command
DLTFNTTBL (Delete Font Table) Command
DLTFNTRSC (Delete Font Resources) Command
DLTFORMDF (Delete Form Definition) Command
DLTFTR (Delete Filter) Command
DLTGSS (Delete Graphics Symbol Set) Command
DLTIGCDCT (Delete DBCS Conversion Dictionary) Command
DLTIGCTBL (Delete DBCS Font Table) Command
DLTIPXD (Delete IPX Description) Command
DLTJOBD (Delete Job Description) Command
DLTJOBQ (Delete Job Queue) Command
DLTJRN (Delete Journal) Command
DLTJRNRCV (Delete Journal Receiver) Command
DLTLIB (Delete Library) Command
DLTLICPGM (Delete Licensed Program) Command
DLTLIND (Delete Line Description) Command
DLTLOCALE (Delete Locale) Command
DLTMNU (Delete Menu) Command
DLTMOD (Delete Module) Command
DLTMODD (Delete Mode Description) Command
DLTMSGF (Delete Message File) Command
DLTMSGQ (Delete Message Queue) Command
DLTM36 (Delete AS/400 Advanced 36 Machine) Command
DLTM36CFG (Delete AS/400 Advanced 36 Machine Configuration) Command
DLTNETF (Delete Network File) Command
DLTNODGRP (Delete Node Group) Command

DLTNODL (Delete Node List) Command
DLTNTBD (Delete NetBIOS Description) Command
DLTNWID (Delete Network Interface Description) Command
DLTNWSALS (Delete Network Server Alias) Command
DLTNWSAPP (Delete Network Server Application) Command
DLTNWSD (Delete Network Server Description) Command
DLTNWSSTG (Delete Network Server Storage Space) Command
DLTOUTQ (Delete Output Queue) Command
DLTOVL (Delete Overlay) Command
DLTOVR (Delete Override) Command
DLTOVRDEVE (Delete Override Device Entry) Command
DLTPAGDFN (Delete Page Definition) Command
DLTPAGSEG (Delete Page Segment) Command
DLTPDG (Delete Print Descriptor Group) Command
DLTPEXDTA (Delete Performance Explorer Data) Command
DLTPGM (Delete Program) Command
DLTPNLGRP (Delete Panel Group) Command
DLTPRB (Delete Problem) Command
DLTPSFCFG (Delete Print Services Facility Configuration) Command
DLTPTF (Delete PTF) Command
DLTQMFORM (Delete Query Management Form) Command
DLTQMQRY (Delete Query Management Query) Command
DLTQRY (Delete Query) Command
DLTQST (Delete Question) Command
DLTQSTDB (Delete Question-and-Answer Database) Command
DLTSBSD (Delete Subsystem Description) Command
DLTSCHIDX (Delete Search Index) Command
DLTSHF (Delete Bookshelf) Command
DLTSPADCT (Delete Spelling Aid Dictionary) Command
DLTSPLF (Delete Spooled File) Command
DLTSQLPKG (Delete Structured Query Language Package) Command
DLTSRVPGM (Delete Service Program) Command
DLTTAPCGY (Delete Tape Category) Command
DLTTBL (Delete Table) Command
DLTUDFS (Delete User-Defined File System) Command
DLTUSRIDX (Delete User Index) Command
DLTUSRPRF (Delete User Profile) Command
DLTUSRQ (Delete User Queue) Command
DLTUSRSPC (Delete User Space) Command
DLTVLDL (Delete Validation List) Command
DLTWSCST (Delete Work Station Customizing Object) Command
DLYJOB (Delay Job) Command
DMPCLPGM (Dump Control Language Program) Command
DMPDLO (Dump Document Library Object) Command

DMPJOB (Dump Job) Command
DMPJOBINT (Dump Job Internal) Command
DMPOBJ (Dump Object) Command
DMPSYSOBJ (Dump System Object) Command
DMPTAP (Dump Tape) Command
DMPTRC (Dump Trace) Command
DO (Do) Command
DSCJOB (Disconnect Job) Command
DSPACC (Display Access Code) Command
DSPACCAUT (Display Access Code Authority) Command
DSPACTPJ (Display Active Prestart Jobs) Command
DSPACTPRFL (Display Active Profile List) Command
DSPACTSCD (Display Activation Schedule) Command
DSPAPPNINF (Display APPN Information) Command
DSPAUDJRNE (Display Audit Journal Entries) Command
DSPAUT (Display Authority) Command
DSPAUTHLR (Display Authority Holder) Command
DSPAUTL (Display Authorization List) Command
DSPAUTLDLO (Display Authorization List Document Library Objects) Command
DSPAUTLOBJ (Display Authorization List Objects) Command
DSPAUTUSR (Display Authorized Users) Command
DSPBCKSTS (Display Backup Status) Command
DSPBCKUP (Display Backup Options) Command
DSPBCKUPL (Display Backup List) Command
DSPBKP (Display Breakpoints) Command
DSPBNDDIR (Display Binding Directory) Command
DSPCCTRTE (Display Circuit Route) Command
DSPCCTSRV (Display Circuit Service) Command
DSPCDEFNT (Display Coded Font) Command
DSPCFGL (Display Configuration List) Command
DSPCLS (Display Class) Command
DSPCMD (Display Command) Command
DSPCNNL (Display Connection List) Command
DSPCNNSTS (Display Connection Status) Command
DSPCOSD (Display Class-of-Service Description) Command
DSPCPCST (Display Check Pending Constraint) Command
DSPCSI (Display Communications Side Information) Command
DSPCTLD (Display Controller Description) Command
DSPCURDIR (Display Current Directory) Command
DSPDBG (Display Debug) Command
DSPDBGWCH (Display Debug Watches) Command
DSPDBR (Display Database Relations) Command
DSPDDMF (Display Distributed Data Management File) Command
DSPDEVD (Display Device Description) Command

DSPDIRE (Display Directory Entries) Command
DSPDKT (Display Diskette) Command
DSPDLOAUD (Display Document Library Object Audit) Command
DSPDLOAUT (Display Document Library Object Authority) Command
DSPDLONAM (Display Document Library Object Name) Command
DSPDOC (Display Document) Command
DSPDSTL (Display Distribution List) Command
DSPDSTLOG (Display Distribution Log) Command
DSPDSTSRV (Display Distribution Services) Command
DSPDTAARA (Display Data Area) Command
DSPDTADCT (Display Data Dictionary) Command
DSPEDTD (Display Edit Description) Command
DSPEWCBCDE (Display Extended Wireless Controller Bar Code Entry) Command
DSPEWCM (Display Extended Wireless Controller Member) Command
DSPEWCPTCE (Display Extended Wireless Controller PTC Entry) Command
DSPEWLM (Display Extended Wireless Line Member) Command
DSPEXPSCD (Display Expiration Schedule) Command
DSPFD (Display File Description) Command
DSPFFD (Display File Field Description) Command
DSPFLR (Display Folder) Command
DSPFNTTBL (Display Font Table) Command
DSPFNTRSCA (Display Font Resource Attributes) Command
DSPHDWRSC (Display Hardware Resources) Command
DSPHFS (Display Hierarchical File Systems) Command
DSPHLPDOC (Display Help Document) Command
DSPIGCDCT (Display DBCS Conversion Dictionary) Command
DSPIPLA (Display IPL Attributes) Command
DSPIPXCCT (Display IPX Circuit) Command
DSPIPXD (Display IPX Description) Command
DSPJOB (Display Job) Command
DSPJOBD (Display Job Description) Command
DSPJOBLOG (Display Job Log) Command
DSPJOBTBL (Display Job Tables) Command
DSPJRN (Display Journal) Command
DSPJRNRCVA (Display Journal Receiver Attributes) Command
DSPKBDMAP (Display Keyboard Map) Command
DSPLANADPP (Display Local Area Network Adapter Profile) Command
DSPLANMLB (Display LAN Media Library) Command
DSPLANSTS (Display Local Area Network Status) Command
DSPLIB (Display Library) Command
DSPLIBD (Display Library Description) Command
DSPLIBL (Display Library List) Command
DSPLICKEY (Display License Key Information) Command
DSPLIND (Display Line Description) Command

718 DSPLNK (Display Object Links) Command
719 DSPLOG (Display Log) Command
720 DSPMFSINF (Display Mounted File System Information) Command
721 DSPMNUA (Display Menu Attributes) Command
722 DSPMOD (Display Module) Command
723 DSPMODD (Display Mode Description) Command
724 DSPMODSRC (Display Module Source) Command
725 DSPMODSTS (Display Mode Status) Command
726 DSPMSG (Display Messages) Command
727 DSPMSGD (Display Message Descriptions) Command
728 DSPM36 (Display AS/400 Advanced 36 Machine) Command
729 DSPM36CFG (Display AS/400 Advanced 36 Machine Configuration) Command
730 DSPNCK (Display Nickname) Command
731 DSPNETA (Display Network Attributes) Command
732 DSPNODGRP (Display Node Group) Command
733 DSPNTBD (Display NetBIOS Description) Command
734 DSPNWID (Display Network Interface Description) Command
735 DSPNWSALS (Display Network Server Alias) Command
736 DSPNWSA (Display Network Server Attributes) Command
737 DSPNWSD (Display Network Server Description) Command
738 DSPNWSSSN (Display Network Server Session) Command
739 DSPNWSSTC (Display Network Server Statistics) Command
739.1 DSPNWSSTG (Display Network Server Storage Space) Command
740 DSPNWSUSR (Display Network Server Users) Command
741 DSPNWSUSRA (Display Network Server User Attributes) Command
742 DSPOBJAUT (Display Object Authority) Command
743 DSPOBJD (Display Object Description) Command
744 DSPOPCLNK (Display OptiConnect Link Status) Command
745 DSPOPT (Display Optical) Command
746 DSPOPTLCK (Display Optical Locks) Command
747 DSPOPTSVR (Display Optical Server) Command
748 DSPOVR (Display Override) Command
749 DSPPDGPRF (Display Print Descriptor Group Profile) Command
750 DSPPFM (Display Physical File Member) Command
751 DSPPGM (Display Program) Command
752 DSPPGMADP (Display Programs that Adopt) Command
753 DSPPGMREF (Display Program References) Command
754 DSPPGMVAR (Display Program Variable) Command
755 DSPPRB (Display Problem) Command
756 DSPPSFCFG (Display Print Services Facility Configuration) Command
757 DSPPTF (Display Program Temporary Fix) Command
758 DSPPWRSCD (Display Power On/Off Schedule) Command
759 DSPRCDLCK (Display Record Locks) Command
760 DSPRCYAP (Display Recovery for Access Paths) Command

DSPRDBDIRE (Display Relational Database Directory Entry) Command
DSPRMTDFN (Display Remote Definition) Command
DSPSAVF (Display Save File) Command
DSPSBSD (Display Subsystem Description) Command
DSPSECA (Display Security Attributes) Command
DSPSECAUD (Display Security Auditing Values) Command
DSPSFWRSC (Display Software Resources) Command
DSPSOCSTS (Display Sphere of Control Status) Command
DSPSPLF (Display Spooled File) Command
DSPSRVA (Display Service Attributes) Command
DSPSRVPGM (Display Service Program) Command
DSPSRVSTS (Display Service Status) Command
DSPSYSSTS (Display System Status) Command
DSPSYSVAL (Display System Value) Command
DSPS36 (Display System/36) Command
DSPTAP (Display Tape) Command
DSPTAPCGY (Display Tape Category) Command
DSPTAPCTG (Display Tape Cartridge) Command
DSPTAPSTS (Display Tape Status) Command
DSPTRC (Display Trace) Command
DSPTRCDTA (Display Trace Data) Command
DSPTM (Display Trademarks) Command
DSPUDFS (Display User-Defined File System) Command
DSPUSRPMN (Display User Permission) Command
DSPUSRPRF (Display User Profile) Command
DSPUSRPRTI (Display User Print Information) Command
DSPWSUSR (Display Work Station User) Command
DUPDKT (Duplicate Diskette) Command
DUPOPT (Duplicate Optical) Command
DUPTAP (Duplicate Tape) Command
EDTAUTL (Edit Authorization List) Command
EDTBCKUPL (Edit Backup List) Command
EDTCPCST (Edit Check Pending Constraints) Command
EDTDLOAUT (Edit Document Library Object Authority) Command
EDTDOC (Edit Document) Command
EDTIGCDCT (Edit DBCS Conversion Dictionary) Command
EDTLIBL (Edit Library List) Command
EDTOBJAUT (Edit Object Authority) Command
EDTQST (Edit Questions and Answers) Command
EDTRBDAP (Edit Rebuild of Access Paths) Command
EDTRCYAP (Edit Recovery for Access Paths) Command
EDTS36PGMA (Edit System/36 Program Attributes) Command
EDTS36PRCA (Edit System/36 Procedure Attributes) Command
EDTS36SRCA (Edit System/36 Source Attributes) Command

EDTWSOAUT (Edit Workstation Object Authority) Command
EJTEMLOUT (Eject Emulation Output) Command
ELSE (Else) Command
EMLPRTKEY (Emulate Printer Key) Command
ENDBCHJOB (End Batch Job) Command
ENDCLNUP (End Cleanup) Command
ENDCMNSVR (End Communications Server) Command
ENDCMNTRC (End Communications Trace) Command
ENDCMTCTL (End Commitment Control) Command
ENDCPYSCN (End Copy Screen) Command
ENDCTLRCY (End Controller Recovery) Command
ENDDBG (End Debug) Command
ENDDBGSVR (End Debug Server) Command
ENDDBMON (End Database Monitor) Command
ENDDEVRCY (End Device Recovery) Command
ENDDIRSHD (End Directory Shadowing) Command
ENDDO (End Do) Command
ENDDSKRGZ (End Disk Reorganization) Command
ENDGRPJOB (End Group Job) Command
ENDHOSTSVR (End Host Server) Command
ENDINP (End Input) Command
ENDIPIIFC (End IP over IPX Interface) Command
ENDIPSIFC (End IP over SNA Interface) Command
ENDIPX (End IPX) Command
ENDIPXCCT (End IPX Circuit) Command
ENDJOB (End Job) Command
ENDJOBABN (End Job Abnormal) Command
ENDJRNAP (End Journal Access Path) Command
ENDJRNPF (End Journal Physical File) Command
ENDLINRCY (End Line Recovery) Command
ENDMOD (End Mode) Command
ENDMSF (End Mail Server Framework) Command
ENDM36 (End AS/400 Advanced 36 Machine) Command
ENDNFSSVR (End Network File System Server) Command
ENDNWIRCY (End Network Interface Recovery) Command
ENDNWSAPP (End Network Server Application) Command
ENDPASTHR (End Pass-Through) Command
ENDPEX (End Performance Explorer) Command
ENDPFRCOL (End Performance Collection) Command
ENDPFRMON (End Performance Monitor) Command
ENDPGM (End Program) Command
ENDPGMPRF (End Program Profiling) Command
ENDPJ (End Prestart Jobs) Command
ENDPRTEML (End Printer Emulation) Command

ENDRCV (End Receive) Command
ENDRDR (End Reader) Command
ENDRMTSPT (End Remote Support) Command
ENDRQS (End Request) Command
ENDSBS (End Subsystem) Command
ENDSRVJOB (End Service Job) Command
ENDSYS (End System) Command
ENDS36 (End System/36) Command
ENDTCP (End TCP/IP) Command
ENDTCPCNN (End TCP/IP Connection) Command
ENDTCPIFC (End TCP/IP Interface) Command
ENDTCPPTP (End Point-to-Point TCP/IP) Command
ENDTCPSVR (End TCP/IP Server) Command
ENDTIESSN (End Technical Information Exchange Session) Command
ENDTRPMGR (End Trap Manager) Command
ENDUSF (End Ultimedia System Facilities) Command
ENDWTR (End Writer) Command
ERASE (Remove Link) Command
EXPORTFS (Change Network File System Export) Command
FILDOC (File Document) Command
GENCAT (Merge Message Catalog) Command
GO (Go to Menu) Command
GOTO (Go To) Command
GRTACCAUT (Grant Access Code Authority) Command
GRTOBJAUT (Grant Object Authority) Command
GRTUSRAUT (Grant User Authority) Command
GRTUSRPMN (Grant User Permission) Command
GRTWSOAUT (Grant Workstation Object Authority) Command
HLDCMNDEV (Hold Communications Device) Command
HLDDSTQ (Hold Distribution Queue) Command
HLDJOB (Hold Job) Command
HLDJOBQ (Hold Job Queue) Command
HLDJOBSCDE (Hold Job Schedule Entry) Command
HLDOUTQ (Hold Output Queue) Command
HLDRDR (Hold Reader) Command
HLDSPLF (Hold Spooled File) Command
HLDWTR (Hold Writer) Command
IF (If) Command
INSFLMSVR (Install FlowMark Server) Command
INSNWSAPP (Install Network Server Application) Command
INSPTF (Install Program Temporary Fix) Command
INZDKT (Initialize Diskette) Command
INZDSTQ (Initialize Distribution Queue) Command
INZOPT (Initialize Optical) Command

INZPCS (Initialize Client Access) Command
INZPFM (Initialize Physical File Member) Command
INZSYS (Initialize System) Command
INZTAP (Initialize Tape) Command
IPXPING Command
LNKDTADFN (Link Data Definition) Command
LODPTF (Load Program Temporary Fix) Command
LODQSTDB (Load Question-and-Answer Database) Command
LODRUN (Load and Run Media Program) Command
MD (Create Directory) Command
MKDIR (Create Directory) Command
MONMSG (Monitor Message) Command
MOUNT (Add Mounted File System) Command
MOV (Move) Command
MOVDOC (Move Document) Command
MOVE (Move) Command
MOVOBJ (Move Object) Command
MRGMSGCLG (Merge Message Catalog) Command
MRGMSGF (Merge Message File) Command
MRGTCPHT (Merge TCP/IP Host Table)
NETSTAT Command
OPNDBF (Open Database File)
OPNQRYF (Open Query File) Command
OVRDBF (Override with Database File) Command
OVRDKTF (Override with Diskette File) Command
OVRDSPF (Override with Display File) Command
OVRICFDEVE (Override with Intersystem Communications Function Program Device Entry) Command
OVRICFF (Override with Intersystem Communications Function File) Command
OVRMSGF (Override with Message File) Command
OVRPRTF (Override with Printer File) Command
OVRSAVF (Override with Save File) Command
OVRTAPF (Override with Tape File) Command
PGM (Program) Command
PING Command
POSDBF (Position Database File) Command
PRTADPOBJ (Print Adopting Objects) Command
PRTAFPDTA (Print Advanced Function Printer Data) Command
PRTCMDUSG (Print Command Usage) Command
PRTCMNSEC (Print Communications Security) Command
PRTCMNTRC (Print Communications Trace) Command
PRTDEVADR (Print Device Addresses) Command
PRTDOC (Print Document) Command
PRTDSKINF (Print Disk Information) Command

PRTERRLOG (Print Error Log) Command
PRTINTDTA (Print Internal Data) Command
PRTIPSCFG (Print IP over SNA Configuration) Command
PRTJOBDAUT (Print Job Description Authority) Command
PRTPUBAUT (Print Publicly Authorized Objects) Command
PRTPVTAUT (Print Private Authorities) Command
PRTQAUT (Print Queue Authority) Command
PRTSBSDAUT (Print Subsystem Description Authority) Command
PRTSQLINF (Print Structured Query Language Information) Command
PRTSWL (Print Stop Word List) Command
PRTSYSINF (Print System Information) Command
PRTSYSSECA (Print System Security Attributes) Command
PRTTRGPGM (Print Trigger Program) Command
PRTUSROBJ (Print User Objects) Command
PRTUSRPRF (Print User Profile) Command
PWRDWNSYS (Power Down System) Command
QRYDOCLIB (Query Document Library) Command
QRYDST (Query Distribution) Command
QRYPRBSTS (Query Problem Status) Command
QRYTIEF (Query Technical Information Exchange File) Command
QSH (Start QSH) Command
RCLACTGRP (Reclaim Activation Group) Command
RCLDDMCNV (Reclaim Distributed Data Management Conversations) Command
RCLDLO (Reclaim Document Library Object) Command
RCLLIB (Reclaim Library) Command
RCLOPT (Reclaim Optical) Command
RCLRSC (Reclaim Resources) Command
RCLSPLSTG (Reclaim Spool Storage) Command
RCLSTG (Reclaim Storage) Command
RCLTMPSTG (Reclaim Temporary Storage) Command
RCVDST (Receive Distribution) Command
RCVF (Receive File) Command
RCVJRNE (Receive Journal Entry) Command
RCVMSG (Receive Message) Command
RCVNETF (Receive Network File) Command
RCVTIEF (Receive Technical Information Exchange File) Command
RD (Remove Directory) Command
REN (Rename) Command
RETURN (Return) Command
RGZDLO (Reorganize Document Library Object) Command
RGZPFM (Reorganize Physical File Member) Command
RLSCMNDEV (Release Communications Device) Command
RLSDSTQ (Release Distribution Queue) Command
RLSIFSLCK (Release Integrated File System Locks) Command

980 RLSJOB (Release Job) Command
981 RLSJOBQ (Release Job Queue) Command
982 RLSJOBSCDE (Release Job Schedule Entry) Command
983 RLSOUTQ (Release Output Queue) Command
984 RLSRDR (Release Reader) Command
985 RLSRMTPHS (Release Remote Phase) Command
986 RLSSPLF (Release Spooled File) Command
987 RLSWTR (Release Writer) Command
988 RMDIR (Remove Directory) Command
989 RMVACC (Remove Access Code) Command
990 RMVAJE (Remove Autostart Job Entry) Command
991 RMVALRD (Remove Alert Description) Command
992 RMVAUTLE (Remove Authorization List Entry) Command
993 RMVBKP (Remove Breakpoint) Command
994 RMVCCTRTE (Remove Circuit Route) Command
995 RMVCCTSRV (Remove Circuit Service) Command
996 RMVBNDDIRE (Remove Binding Directory Entry) Command
997 RMVCFGLE (Remove Configuration List Entries) Command
998 RMVCMNE (Remove Communications Entry) Command
999 RMVCNNLE (Remove Connection List Entry) Command
1000 RMVCOMSNMP (Remove Community for SNMP) Command
1001 RMVDIR (Remove Directory) Command
1002 RMVDIRE (Remove Directory Entry) Command
1003 RMVDIRSHD (Remove Directory Shadow System) Command
1004 RMVDLOAUT (Remove Document Library Object Authority) Command
1005 RMVDSTLE (Remove Distribution List Entry) Command
1006 RMVDSTQ (Remove Distribution Queue) Command
1007 RMVDSTRTE (Remove Distribution Route) Command
1008 RMVDSTSYSN (Remove Distribution Secondary System Name) Command
1009 RMVEMLCFGE (Remove Emulation Configuration Entry) Command
1010 RMVEWCBCDE (Remove Extended Wireless Controller Bar Code Entry) Command
1011 RMVEWCPTCE (Remove Extended Wireless Controller PTC Entry) Command
1012 RMVEXITPGM (Remove Exit Program) Command
1013 RMVFNTTBLE (Remove Font Table Entry) Command
1014 RMVFTRACNE (Remove Filter Action Entry) Command
1015 RMVFTRSLTE (Remove Filter Selection Entry) Command
1016 RMVICFDEVE (Remove Intersystem Communications Function Program Device Entry) Command
1017 RMVIPIADR (Remove IP over IPX Address) Command
1018 RMVIPIIFC (Remove IP over IPX Interface) Command
1019 RMVIPIRTE (Remove IP over IPX Route) Command
1020 RMVIPSIFC (Remove IP over SNA interface) Command
1021 RMVIPSLOC (Remove IP over SNA Location Entry) Command
1022 RMVIPSRTE (Remove IP over SNA Route) Command

RMVIPXCCT (Remove IPX Circuit) Command
RMVJOBQE (Remove Job Queue Entry) Command
RMVJOBSCDE (Remove Job Schedule Entry) Command
RMVJRNCHG (Remove Journaled Changes) Command
RMVLANADPI (Remove LAN Adapter Information) Command
RMVLANADPT (Remove LAN Adapter) Command
RMVLIBLE (Remove Library List Entry) Command
RMVLICKEY (Remove License Key Information) Command
RMVLNK (Remove Link) Command
RMVM (Remove Member) Command
RMVMFS (Remove Mounted File System) Command
RMVMSG (Remove Message) Command
RMVMSGD (Remove Message Description) Command
RMVNCK (Remove Nickname) Command
RMVNETJOBE (Remove Network Job Entry) Command
RMVNETTBLE (Remove Network Table Entry) Command
RMVNODLE (Remove Node List Entry) Command
RMVNWSSTGL (Remove Network Server Storage Link) command
RMVOPTCTG (Remove Optical Cartridge) Command
RMVOPTSVR (Remove Optical Server) Command
RMVPCLTBLE (Remove Protocol Table Entry) Command
RMVPEXDFN (Remove Performance Explorer Definition) Command
RMVPFCST (Remove Physical File Constraint) Command
RMVPFTRG (Remove Physical File Trigger) Command
RMVPGM (Remove Program) Command
RMVPJE (Remove Prestart Job Entry) Command
RMVPTF (Remove Program Temporary Fix) Command
RMVRDBDIRE (Remove Relational Database Directory Entry) Command
RMVREXBUF (Remove REXX Buffer) Command
RMVRMTDFN (Remove Remote Definition) Command
RMVRPYLE (Remove Reply List Entry) Command
RMVRTGE (Remove Routing Entry) Command
RMVSCHIDXE (Remove Search Index Entry) Command
RMVSNILOC (Remove SNA over IPX Location) Command
RMVSOCE (Remove Sphere of Control Entry) Command
RMVSRVTBLE (Remove Service Table Entry) Command
RMVSVRAUTE (Remove Server Authentication Entry) Command
RMVTAPCTG (Remove Tape Cartridge) Command
RMVTCPHTE (Remove TCP/IP Host Table Entry) Command
RMVTCPIFC (Remove TCP/IP Interface) Command
RMVTCPPORT (Remove TCP/IP Port Restriction) Command
RMVTCPRSI (Remove TCP/IP Remote System Information) Command
RMVTCPRTE (Remove TCP/IP Route) Command
RMVTCPTBL (Remove TCP/IP Table) Command

1067 RMVTRC (Remove Trace) Command
1068 RMVUSFCNNE (Remove Ultimedia System Facilities Connection Entry) Command
1069 RMVUSFDEVE (Remove Ultimedia System Facilities Device Entry) Command
1070 RMVUSFSVRE (Remove Ultimedia System Facilities Server Entry) Command
1071 RMVWSE (Remove Work Station Entry) Command
1072 RNM (Rename) Command
1073 RNMCNNLE (Rename Connection List Entry) Command
1074 RNMDIRE (Rename Directory Entry) Command
1075 RNMDKT (Rename Diskette) Command
1076 RNMDLO (Rename Document Library Object) Command
1077 RNMDSTL (Rename Distribution List) Command
1078 RNMLANADPI (Rename Local Area Network Adapter Information) Command
1079 RNMM (Rename Member) Command
1080 RNMNCK (Rename Nickname) Command
1081 RNMOBJ (Rename Object) Command
1082 RNMTCPHTE (Rename TCP/IP Host Table Entry) Command
1083 ROLLBACK (Rollback) Command
1084 RPCBIND (Start RPC Binder Daemon) Command
1085 RPCGEN (Convert RPC Source) Command
1086 RPLDOC (Replace Document) Command
1087 RQSORDAST (Request Order Assistance) Command
1088 RRTJOB (Reroute Job) Command
1089 RSMBKP (Resume Breakpoint) Command
1090 RSMCTLRCY (Resume Controller Recovery) Command
1091 RSMDEVRCY (Resume Device Recovery) Command
1092 RSMLINRCY (Resume Line Recovery) Command
1093 RSMNWIRCY (Resume Network Interface Recovery) Command
1094 RST (Restore) Command
1095 RSTAUT (Restore Authority) Command
1096 RSTCFG (Restore Configuration) Command
1097 RSTDLO (Restore Document Library Object) Command
1098 RSTLIB (Restore Library) Command
1099 RSTLICPGM (Restore Licensed Program) Command
1100 RSTOBJ (Restore Object) Command
1101 RSTSHF (Restore Bookshelf) Command
1102 RSTS36F (Restore System/36 File) Command
1103 RSTS36FLR (Restore System/36 Folder) Command
1104 RSTS36LIBM (Restore System/36 Library Members) Command
1105 RSTUSFCNR (Restore Ultimedia System Facilities Container) Command
1106 RSTUSRPRF (Restore User Profiles) Command
1107 RTVAUTLE (Retrieve Authorization List Entry) Command
1108 RTVBCKUP (Retrieve Backup) Command
1109 RTVBNDSRC (Retrieve Binder Source) Command
1110 RTVCFGSRC (Retrieve Configuration Source) Command

RTVCFGSTS (Retrieve Configuration Status) Command
RTVCLNUP (Retrieve Cleanup) Command
RTVCLSRC (Retrieve CL Source) Command
RTVCURDIR (Retrieve Current Directory) Command
RTVDLOAUT (Retrieve Document Library Object Authority) Command
RTVDLONAM (Retrieve Document Library Object Name) Command
RTVDOC (Retrieve Document) Command
RTVDSKINF (Retrieve Disk Information) Command
RTVDTAARA (Retrieve Data Area) Command
RTVGRPA (Retrieve Group Attributes) Command
RTVJOBA (Retrieve Job Attributes) Command
RTVJRNE (Retrieve Journal Entry) Command
RTVLIBD (Retrieve Library Description) Command
RTVMBRD (Retrieve Member Description) Command
RTVMSG (Retrieve Message) Command
RTVNETA (Retrieve Network Attributes) Command
RTVOBJD (Retrieve Object Description) Command
RTVPDGPRF (Retrieve Print Descriptor Group Profile) Command
RTVPWRSCDE (Retrieve Power On/Off Schedule Entry) Command
RTVQMFORM (Retrieve Query Management Form) Command
RTVQMQRY (Retrieve Query Management Query) Command
RTVSWLSRC (Retrieve Stop Word List Source) Command
RTVSYSVAL (Retrieve System Value) Command
RTVSYSINF (Retrieve System Information) Command
RTVS36A (Retrieve System/36 Attributes) Command
RTVTBSRC (Retrieve Table Source) Command
RTVUSRPRF (Retrieve User Profile) Command
RTVUSRPRTI (Retrieve User Print Information) Command
RTVWSCST (Retrieve Work Station Customizing Object Source) Command
RUNBCKUP (Run Backup) Command
RUNLPDA (Run LPDA-2) Command
RUNQRY (Run Query) Command
RUNRMTCMD (Run Remote Command) Command
RVKACCAUT (Revoke Access Code Authority) Command
RVKOBJAUT (Revoke Object Authority) Command
RVKPUBAUT (Revoke Public Authority) Command
RVKUSRPMN (Revoke User Permission) Command
RVKWSOAUT (Revoke Workstation Object Authority) Command
SAV (Save) Command
SAVAPARDTA (Save APAR Data) Command
SAVCFG (Save Configuration) Command
SAVCHGOBJ (Save Changed Object) Command
SAVDLO (Save Document Library Object) Command
SAVLIB (Save Library) Command

SAVLICPGM (Save Licensed Program) Command
SAVOBJ (Save Object) Command
SAVRST (Save/Restore Objects) Command
SAVRSTCFG (Save/Restore Configuration) Command
SAVRSTCHG (Save/Restore Changed Object) Command
SAVRSTDLO (Save/Restore Document Library Object) Command
SAVRSTLIB (Save/Restore Library) Command
SAVRSTOBJ (Save/Restore Object) Command
SAVSAVFDTA (Save Save File Data) Command
SAVSECDTA (Save Security Data) Command
SAVSHF (Save Bookshelf) Command
SAVSTG (Save Storage) Command
SAVSYS (Save System) Command
SAVS36F (Save System/36 File) Command
SAVS36LIBM (Save System/36 Library Members) Command
SAVUSFCNR (Save Ultimedia System Facilities Container) Command
SBMDBJOB (Submit Database Jobs) Command
SBMDKTJOB (Submit Diskette Jobs) Command
SBMFNCJOB (Submit Finance Job) Command
SBMJOB (Submit Job) Command
SBMNETJOB (Submit Network Job) Command
SBMNWSCMD (Submit Network Server Command) Command
SBMRMTCMD (Submit Remote Command) Command
SETATNPGM (Set Attention Program) Command
SETCSTDTA (Set Customization Data) Command
SETKBDMAP (Set Keyboard Map) Command
SETOBJACC (Set Object Access) Command
SETTAPCGY (Set Tape Category) Command
SIGNOFF (Sign Off) Command
SLTCMD (Select Command) Command
SNDBRKMSG (Send Break Message) Command
SNDDST (Send Distribution) Command
SNDDSTQ (Send Distribution Queue) Command
SNDEMLIGC (Send DBCS 3270PC Emulation Code) Command
SNDF (Send File) Command
SNDFNCIMG (Send Finance Diskette Image) Command
SNDJRNE (Send Journal Entry) Command
SNDMSG (Send Message) Command
SNDNETF (Send Network File) Command
SNDNETMSG (Send Network Message) Command
SNDNETSPLF (Send Network Spooled File) Command
SNDNWSMSG (Send Network Server Message) Command
SNDPGMMSG (Send Program Message) Command
SNDPTFORD (Send Program Temporary Fix Order) Command

1199 SNDRCVF (Send/Receive File) Command
1200 SNDRPY (Send Reply) Command
1201 SNDSRVRQS (Send Service Request) Command
1202 SNDTIEF (Send Technical Information Exchange File) Command
1203 SNDUSRMSG (Send User Message) Command
1204 STATFS (Display Mounted File System Information) Command
1205 STRCLNUP (Start Cleanup) Command
1206 STRCMNSVR (Start Communications Server) Command
1207 STRCMNTRC (Start Communications Trace) Command
1208 STRCMTCTL (Start Commitment Control) Command
1209 STRCPYSCN (Start Copy Screen) Command
1210 STRDBG (Start Debug) Command
1211 STRDBGSVR (Start Debug Server) Command
1212 STRDBMON (Start Database Monitor) Command
1213 STRDBRDR (Start Database Reader) Command
1214 STRDIRSHD (Start Directory Shadowing) Command
1215 STRDKTRDR (Start Diskette Reader) Command
1216 STRDKTWTR (Start Diskette Writer) Command
1217 STRDSKRGZ (Start Disk Reorganization) Command
1218 STREDU (Start Education) Command
1219 STREML3270 (Start 3270 Display Emulation) Command
1220 STRFMA (Start Font Management Aid) Command
1221 STRHOSTSVR (Start Host Server) Command
1222 STRIDD (Start Interactive Data Definition Utility) Command
1223 STRINFSKR (Start InfoSeeker) Command
1224 STRIPIIFC (Start IP over IPX Interface) Command
1225 STRIPSIFC (Start IP over SNA interface) Command
1226 STRIPX (Start IPX) Command
1227 STRIPXCCT (Start IPX Circuit) Command
1228 STRITF (Start Interactive Terminal Facility) Command
1229 STRJRNAP (Start Journal Access Path) Command
1230 STRJRNPF (Start Journal Physical File) Command
1231 STRMOD (Start Mode) Command
1232 STRMSF (Start Mail Server Framework) Command
1233 STRM36 (Start AS/400 Advanced 36 Machine) Command
1234 STRM36PRC (Start AS/400 Advanced 36 Machine Procedure) Command
1235 STRNFSSVR (Start Network File System Server) Command
1236 STRNWSAPP (Start Network Server Application) Command
1237 STROBJCVN (Start Object Conversion) Command
1238 STRPASTHR (Start Pass-Through) Command
1239 STRPEX (Start Performance Explorer) Command
1240 STRPFRCOL (Start Performance Collection) Command
1241 STRPFRMON (Start Performance Monitor) Command
1242 STRPGMMNU (Start Programmer Menu) Command

1243 STRPGMPRF (Start Program Profiling) Command
1244 STRPJ (Start Prestart Jobs) Command
1245 STRPRTEML (Start Printer Emulation) Command
1246 STRPRTWTR (Start Printer Writer) Command
1247 STRQMPRC (Start Query Management Procedure) Command
1248 STRQMQRY (Start Query Management Query) Command
1249 STRQRY (Start Query) Command
1250 STRQSH (Start QSH) Command
1251 STRQST (Start Question and Answer) Command
1252 STRREXPRC (Start REXX Procedure) Command
1253 STRRMTSPT (Start Remote Support) Command
1254 STRRMTWTR (Start Remote Writer) Command
1255 STRSBS (Start Subsystem) Command
1256 STRSCHIDX (Start Search Index) Command
1257 STRSPTN (Start Support Network) Command
1258 STRSRVJOB (Start Service Job) Command
1259 STRSST (Start System Service Tools) Command
1260 STRS36 (Start System/36) Command
1261 STRS36PRC (Start System/36 Procedure) Command
1262 STRTCP (Start TCP/IP) Command
1263 STRTCPIFC (Start TCP/IP Interface) Command
1264 STRTCPPTP (Start Point-to-Point TCP/IP) Command
1265 STRTCPSVR (Start TCP/IP Server) Command
1266 STRTIESSN (Start Technical Information Exchange Session) Command
1267 STRTRPMGR (Start Trap Manager) Command
1268 STRUSF (Start Ultimedia System Facilities) Command
1269 TFRBCHJOB (Transfer Batch Job) Command
1270 TFRCTL (Transfer Control) Command
1271 TFRGRPJOB (Transfer to Group Job) Command
1272 TFRJOB (Transfer Job) Command
1273 TFRM36 (Transfer to AS/400 Advanced 36 Machine) Command
1274 TFRPASTHR (Transfer Pass-Through) Command
1275 TFRSECJOB (Transfer Secondary Job) Command
1276 TRCCPIC (Trace CPI Communications) Command
1277 TRCICF (Trace ICF) Command
1278 TRCINT (Trace Internal) Command
1279 TRCJOB (Trace Job) Command
1280 TRCREX (Trace REXX) Command
1281 UNMOUNT (Remove Mounted File System) Command
1282 UPDPGM (Update Program) Command
1283 UPDSRVPGM (Update Service Program) Command
1284 UPDSYSINF (Update System Information) Command
1285 VFYAPPCCNN (Verify APPC Connection) Command
1286 VFYCMN (Verify Communications) Command

1287 VFYIPXCNN (Verify IPX Connection) Command
1288 VFYLNKLPDA (Verify Link Supporting LPDA-2) Command
1289 VFYOPCCNN (Verify OptiConnect Connections) Command
1290 VFYOPT (Verify Optical) Command
1291 VFYPRT (Verify Printer) Command
1292 VFYTAP (Verify Tape) Command
1293 VFYTCPCNN (Verify TCP/IP Connection) Command
1294 VRYCFG (Vary Configuration) Command
1295 WAIT (Wait) Command
1296 WRKACTJOB (Work with Active Jobs) Command
1297 WRKALR (Work with Alerts) Command
1298 WRKALRD (Work with Alert Descriptions) Command
1299 WRKALRTBL (Work with Alert Tables) Command
1300 WRKAPPNSTS (Work with APPN Status) Command
1301 WRKAUT (Work with Authority) Command
1302 WRKAUTL (Work with Authorization Lists) Command
1303 WRKBNDDIR (Work with Binding Directories) Command
1304 WRKBNDDIRE (Work with Binding Directory Entries) Command
1305 WRKBPTBL (Work with BOOTP Table) Command
1306 WRKCCTRTE (Work with Circuit Routes) Command
1307 WRKCCTSRV (Work with Circuit Services) Command
1308 WRKCFGL (Work with Configuration Lists) Command
1309 WRKCFGSTS (Work with Configuration Status) Command
1310 WRKCHTFMT (Work with Chart Formats) Command
1311 WRKCLS (Work with Classes) Command
1312 WRKCMD (Work with Commands) Command
1313 WRKCMTDFN (Work with Commitment Definitions) Command
1314 WRKCNNL (Work with Connection Lists) Command
1315 WRKCNNLE (Work with Connection List Entries) Command
1316 WRKCNTINF (Work with Contact Information) Command
1317 WRKCOSD (Work with Class-of-Service Descriptions) Command
1318 WRKCSI (Work with Communications Side Information) Command
1319 WRKCTLD (Work with Controller Descriptions) Command
1320 WRKDBFIDD (Work with Database Files Using IDDU) Command
1321 WRKDDMF (Work with Distributed Data Management Files) Command
1322 WRKDEVD (Work with Device Descriptions) Command
1323 WRKDEVTBL (Work with Device Tables) Command
1324 WRKDIRE (Work with Directory Entries) Command
1325 WRKDIRLOC (Work with Directory Locations) Command
1326 WRKDIRSHD (Work with Directory Shadow Systems) Command
1327 WRKDOC (Work with Documents) Command
1328 WRKDOCLIB (Work with Document Libraries) Command
1329 WRKDOCPRTQ (Work with Document Print Queue) Command
1330 WRKDPCQ (Work with DSNX/PC Distribution Queues) Command

WRKDSKSTS (Work with Disk Status) Command
WRKDSTL (Work with Distribution Lists) Command
WRKDSTQ (Work with Distribution Queues) Command
WRKDTAARA (Work with Data Areas) Command
WRKDTADCT (Work with Data Dictionaries) Command
WRKDTADFN (Work with Data Definitions) Command
WRKDTAQ (Work with Data Queues) Command
WRKEDTD (Work with Edit Descriptions) Command
WRKENVVAR (Work with Environment Variables) Command
WRKF (Work with Files) Command
WRKFLR (Work with Folders) Command
WRKFNTRSC (Work with Font Resources) Command
WRKFORMDF (Work with Form Definitions) Command
WRKFTR (Work with Filters) Command
WRKFTRACNE (Work with Filter Action Entries) Command
WRKFTRSLTE (Work with Filter Selection Entries) Command
WRKGSS (Work with Graphics Symbol Sets) Command
WRKHDWRSC (Work with Hardware Resources) Command
WRKHLDOPTF (Work with Held Optical Files) Command
WRKIPXCCT (Work with IPX Circuits) Command
WRKIPXD (Work with IPX Descriptions) Command
WRKIPXSTS (Work with IPX Status) Command
WRKJOB (Work with Job) Command
WRKJOBD (Work with Job Descriptions) Command
WRKJOBQ (Work with Job Queues) Command
WRKJOBSCDE (Work with Job Schedule Entries) Command
WRKJRN (Work with Journal) Command
WRKJRNA (Work with Journal Attributes) Command
WRKJRNRCV (Work with Journal Receivers) Command
WRKLANADPT (Work with LAN Adapters) Command
WRKLIB (Work with Libraries) Command
WRKLICINF (Work with License Information) Command
WRKLNK (Work with Object Links) Command
WRKMLBSTS (Work with Media Library Status) Command
WRKMNU (Work with Menus) Command
WRKMOD (Work with Modules) Command
WRKMODD (Work with Mode Descriptions) Command
WRKMSG (Work with Messages) Command
WRKMSGD (Work with Message Descriptions) Command
WRKMSGF (Work with Message Files) Command
WRKMSGQ (Work with Message Queues) Command
WRKM36 (Work with AS/400 Advanced 36 Machines) Command
WRKM36CFG (Work with AS/400 Advanced 36 Machine Configurations) Command
WRKNCK (Work with Nicknames) Command

1375 WRKNETF (Work with Network Files) Command
1376 WRKNETJOBE (Work with Network Job Entries) Command
1377 WRKNETTBLE (Work with Network Table Entry) Command
1378 WRKNODL (Work with Node Lists) Command
1379 WRKNODLE (Work with Node List Entries) Command
1380 WRKNTBD (Work with NetBIOS Descriptions) Command
1381 WRKNWID (Work with Network Interface Description) Command
1382 WRKNWSALS (Work with Network Server Aliases) Command
1383 WRKNWSD (Work with Network Server Descriptions) Command
1384 WRKNWSENR (Work with Network Server User Enrollment) Command
1385 WRKNWSSSN (Work with Network Server Sessions) Command
1385.1 WRKNWSSTG (Work with Network Server Storage Spaces) Command
1386 WRKNWSSTS (Work with Network Server Status) Command
1387 WRKOBJ (Work with Objects) Command
1388 WRKOBJLCK (Work with Object Locks) Command
1389 WRKOBJOWN (Work with Objects by Owner) Command
1390 WRKOBJPGP (Work with Objects by Primary Group) Command
1391 WRKOPTDIR (Work with Optical Directories) Command
1392 WRKOPTF (Work with Optical Files) Command
1393 WRKOPTVOL (Work with Optical Volumes) Command
1394 WRKOPCACT (Work with OptiConnect Activity) Command
1395 WRKORDINF (Work with Order Information) Command
1396 WRKORDRQS (Work with Order Requests) Command
1397 WRKOUTQ (Work with Output Queues) Command
1398 WRKOUTQD (Work with Output Queue Description) Command
1399 WRKOVL (Work with Overlays) Command
1400 WRKPAGDFN (Work with Page Definitions) Command
1401 WRKPAGSEG (Work with Page Segments) Command
1402 WRKPCLTBLE (Work with Protocol Table Entry) Command
1403 WRKPFCST (Work with Physical File Constraints) Command
1404 WRKPFRCOL (Work with Performance Collection) Command
1405 WRKPGM (Work with Programs) Command
1406 WRKPGMTBL (Work with Program Tables) Command
1407 WRKPNLGRP (Work with Panel Groups) Command
1408 WRKPRB (Work with Problems) Command
1409 WRKPRDINF (Work with Product Information) Command
1410 WRKPRTSTS (Work with Printing Status) Command
1411 WRKPSFCFG (Work with Print Services Facility Configurations) Command
1412 WRKQMFORM (Work with Query Management Forms) Command
1413 WRKQMQRY (Work with Query Management Queries) Command
1414 WRKQST (Work with Questions) Command
1415 WRKRDBDIRE (Work with Relational Database Directory Entries) Command
1416 WRKRDR (Work with Readers) Command
1417 WRKREGINF (Work with Registration Information) Command

1418 WRKRMTDFN (Work with Remote Definitions) Command
1419 WRKRPYLE (Work with System Reply List Entries) Command
1420 WRKRTDCFG (Work with RouteD Configuration) Command
1421 WRKSBMJOB (Work with Submitted Jobs) Command
1422 WRKSBS (Work with Subsystems) Command
1423 WRKSBSD (Work with Subsystem Descriptions) Command
1424 WRKSBSJOB (Work with Subsystem Jobs) Command
1425 WRKSCHIDX (Work with Search Indexes) Command
1426 WRKSCHIDXE (Work with Search Index Entries) Command
1427 WRKSHRPOOL (Work with Shared Storage Pools) Command
1428 WRKSOC (Work with Sphere of Control) Command
1429 WRKSPADCT (Work with Spelling Aid Dictionaries) Command
1430 WRKSPLF (Work with Spooled Files) Command
1431 WRKSPLFA (Work with Spooled File Attributes) Command
1432 WRKSRVPGM (Work with Service Programs) Command
1433 WRKSRVPVD (Work with Service Providers) Command
1434 WRKSRVTBLE (Work with Service Table Entry) Command
1435 WRKSYSSTS (Work with System Status) Command
1436 WRKSYSVAL (Work with System Values) Command
1437 WRKS36 (Work with System/36) Command
1438 WRKS36PGMA (Work with System/36 Program Attributes) Command
1439 WRKS36PRCA (Work with System/36 Procedure Attributes) Command
1440 WRKS36SRCA (Work with System/36 Source Attributes) Command
1441 WRKTAPCTG (Work with Tape Cartridges) Command
1442 WRKTBL (Work with Tables) Command
1443 WRKTCPPTP (Work with Point-to-Point TCP/IP) Command
1444 WRKTCPSTS (Work with TCP/IP Network Status) Command
1445 WRKTIE (Work with Technical Information Exchange) Command
1446 WRKUSFCNNE (Work with Ultimedia System Facilities Connection Entries) Comman
d
1447 WRKUSFDEVE (Work with Ultimedia System Facilities Device Entries) Command
1448 WRKUSFSVRE (Work with Ultimedia System Facilities Server Entries) Command
1449 WRKUSRJOB (Work with User Jobs) Command
1450 WRKUSRPRF (Work with User Profiles) Command
1451 WRKUSRTBL (Work with User Tables) Command
1452 WRKWTR (Work with Writers) Command