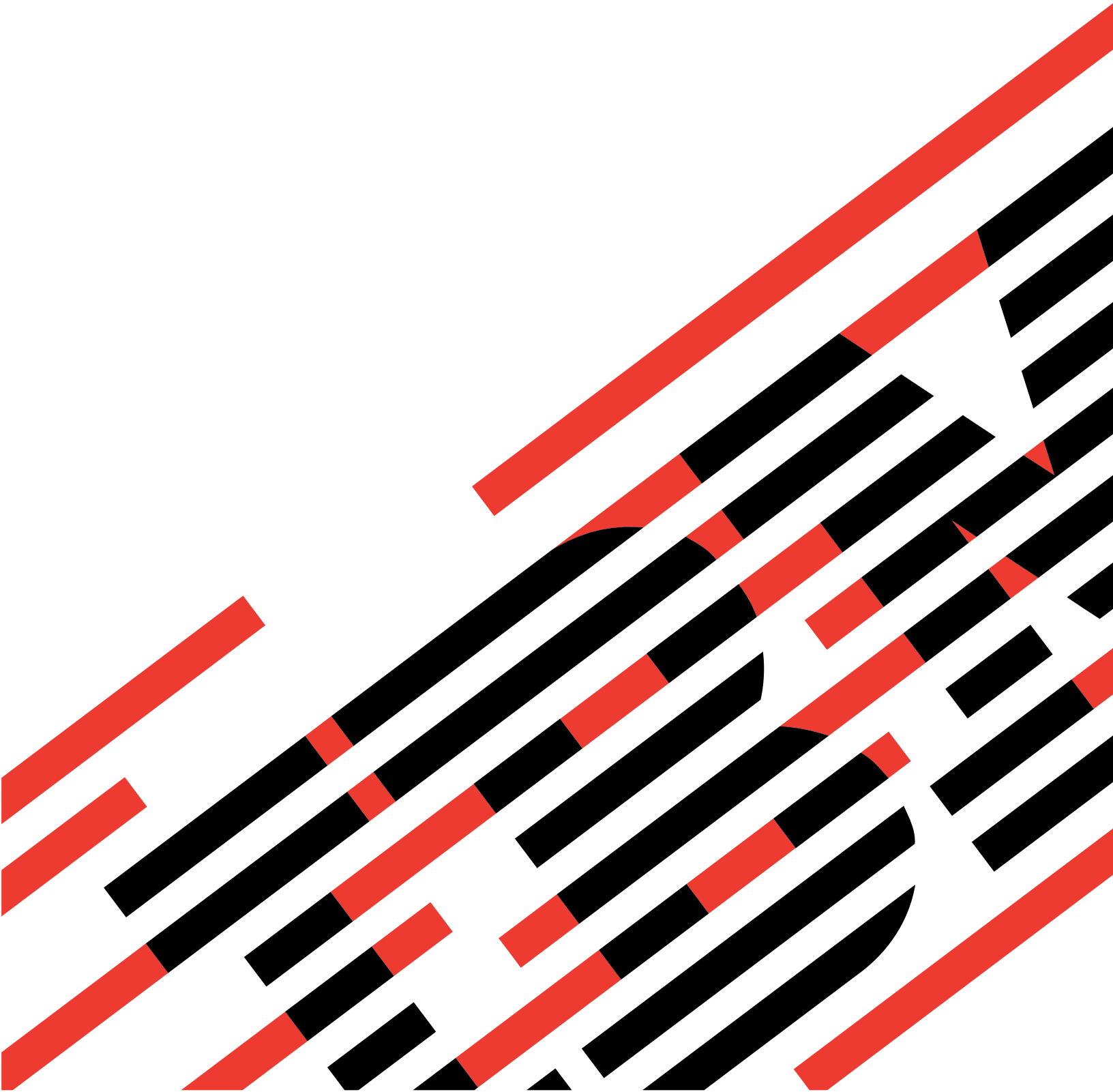
IBM

eserver

iSeries

集成文件系统介绍

版本 5



IBM

@server

iSeries

集成文件系统介绍

版本 5

目录

关于《集成文件系统介绍》

本书概述了集成文件系统，包括以下几个方面:

- 什么是集成文件系统？
- 为什么要使用它？
- 集成文件系统的概念和术语。
- 可以用来与集成文件系统进行交互的接口。
- 可以用来创建与集成文件系统进行交互的程序的 API 和技术。
- 各个文件系统的特征。

谁应该阅读《集成文件系统介绍》一书

本书供想要了解集成文件系统以及如何使用它的 iSeries 服务器用户、程序员和管理员之用。

代码不保证声明信息

本文档包含编程示例。

IBM 授予您使用所有编程代码示例的非专有版权许可证，您可以由此生成相似的定制功能以满足您特定的需要。

IBM 提供的所有样本代码仅只是出于解释的目的。并未在所有环境下完全测试这些示例。因此，IBM 不保证或默示这些程序的可靠性、可服务性或功能。

本文档中包含的所有程序是以“按现状”的基础提供的，不附有任何形式的保证。明示的不保证声明包括非侵权性、适销性和适用于某特定用途的默示保证。

第 1 章 介绍集成文件系统

下列主题描述了 iSeries 服务器上的集成文件系统，并显示了如何在您的服务器上使用它。

集成文件系统是什么

集成文件系统是 OS/400 的一个部件，它支持与个人计算机和 UNIX 操作系统类似的流输入／输出和存储管理，同时还为存储在您的服务器中的所有信息提供了一种集成结构。

集成文件系统的主要功能部件如下:

- 支持在可以包含连续的长数据串的流文件中存储信息。例如，这些数据串可以是文档的文本或图片中的图片元素。流文件支持是为了在客户机／服务器应用程序有效使用流文件而设计的。
- 分层目录结构，它允许像树枝上的果实一样组织对象。通过指定对象的目录路径，可以访问该对象。
- 公共接口，它不仅允许用户和应用程序访问流文件，而且可以访问存储在您的服务器中的数据库文件、文档和其它对象。
- 流文件的公共视图，这些流文件以本地方式存储在您的服务器、Integrated xSeries Server for iSeries 或远程 Windows NT 服务器上。也可以将流文件以远程方式存储在“局域网”（LAN）服务器、Novell NetWare 服务器、另一远程 iSeries 服务器或“网络文件系统”服务器上。

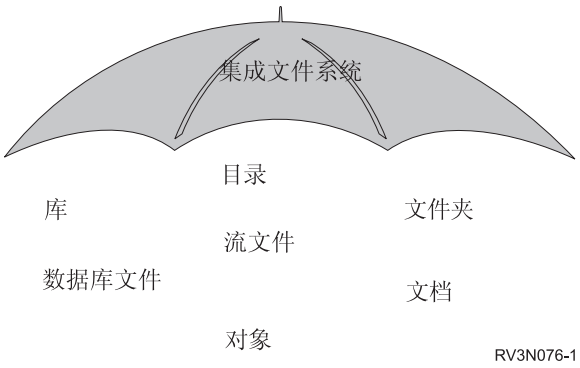


图 1. 存储在 iSeries 服务器中的所有信息的结构

为什么使用集成文件系统

集成文件系统通过附加能力增强了 OS/400 已经很广泛的数据管理能力，以更好地支持正在涌现的和将来的信息处理方式，如客户机／服务器、开放系统和多媒体。

可以使用集成文件系统来:

- 提供快速访问 OS/400 数据的能力，特别是对于使用 OS/400 文件服务器的应用程序（如 Client Access）更是如此。
- 允许更有效地处理流文件的各种类型，如图像、音频和视频。
- 提供用于支持基于 UNIX 开放系统标准（如“计算机环境的可移植操作系统接口”（POSIX）和 XPG）的文件系统库和目录库。这种文件结构和这种目录结构还为诸如“磁盘操作系统”（DOS）和 Windows 操作系统的 PC 操作系统的用户提供了一个熟悉的环境。

- 允许将具有独特能力（如面向记录的数据库文件、基于 UNIX 的流文件和文件服务）的文件支持作为独立的文件系统进行处理，以允许通过公共接口来管理所有这些文件支持。

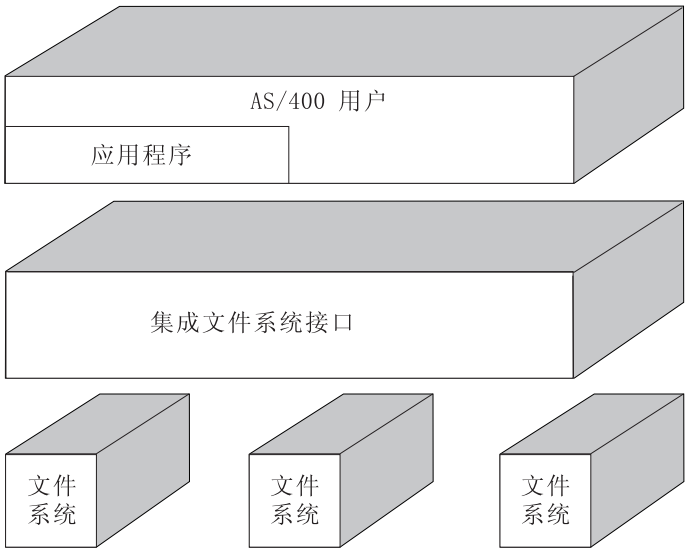


图 2. 独立文件系统的公共接口

- 允许 PC 用户更好地利用他们的图形用户界面。例如，Windows 用户可以使用 Windows 图形工具，以他们对存储在 PC 上的文件进行操作的相同方式对 iSeries 服务器流文件和其它对象进行操作。
- 跨本地语言提供对象名称的连续性和相关联的对象信息。例如，这确保单个的字符从一种语言的代码页转换至另一种语言的代码页时保持相同。

第 2 章 集成文件系统概念

流文件

流文件是可随机访问的一系列字节，系统未强加进一步的结构。集成文件系统提供了用于存储和操作流文件格式信息的支持。存储在服务器文件夹中的文档是流文件。流文件的其它示例是 PC 文件和 UNIX 系统中的文件。集成文件系统流文件是一个对象类型为 *STMF 的系统对象。

要更好地理解流文件，可以将它们与 iSeries 数据库文件进行比较。数据库文件是面向记录的，它预定义了一些细分，这些细分由一个或多个字段组成，这些字段具有特定的特征，如长度和数据类型。

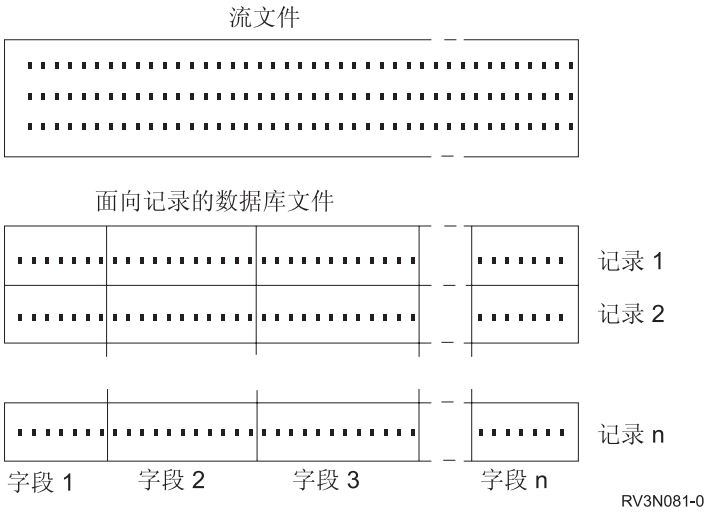


图 3. 比较流文件和面向记录的文件

流文件和面向记录的文件在结构上是不同的，这种结构上的差异影响如何使用文件。结构会影响如何编写应用程序以与文件进行交互，以及每种类型的文件最适合于用在应用程序中的哪个位置。例如，面向记录的文件最适于存储诸如名称、地址和帐户余额的客户统计信息。面向记录的文件允许使用服务器的各种编程设施来逐个访问和管理这些预定义的字段。而流文件则较适合于存储诸如客户图片的信息，该信息由表示颜色变化的连续位串组成。流文件尤其适合于存储诸如文档文本、图像、声音和视频的数据串。

有关集成文件系统中流文件的更多信息，参见:

- 第 37 页的『在流文件和数据库文件之间复制数据』。
- 『*TYPE1 和 *TYPE2 流文件』。

*TYPE1 和 *TYPE2 流文件

一个文件具有两种格式选项之一: *TYPE1 流文件或 *TYPE2 流文件。

*TYPE1 流文件与在 OS/400 的版本 4 发行版 4 以前的发行版中创建的流文件具有相同的格式。在保存到 OS/400 的版本 4 发行版 4 以前的发行版时，保存 *TYPE1 流文件比保存 *TYPE2 流文件更快。它最小为 4096 字节。

*TYPE2 流文件具有高性能的文件访问并且是 OS/400 的版本 4 发行版 4 中的新文件。它保存到 OS/400 的版本 4 发行版 4 以前的发行版的速度比 *TYPE1 流文件慢。它的对象大小最小为 4096 字节。使用 V4R4 和更高版本的系统创建的所有文件都是 *TYPE2 流文件。

尽管 *TYPE2 流文件仅与 V4R4 和更高版本的系统一起使用，但您可以保存 *TYPE2 流文件以便在 V4R4 之前的系统上进行恢复。但是，此过程可能较慢。

集成文件系统中的文件系统

文件系统支持访问按逻辑单元组织的存储器的特定段。服务器上的这些逻辑单元是文件、目录、库和对象。

每个文件系统都有一组逻辑结构和规则用于与存储器中的信息进行交互。这些结构和规则在文件系统之间可以不同。事实上，从结构和规则的角度来看，可以将用于通过库来访问数据库文件和各种其它对象类型的 OS/400 支持看成一个文件系统。同样，可以将用于通过文件夹结构来访问文档（其实是流文件）的 OS/400 支持看成一个独立的文件系统。

集成文件系统将库支持和文件夹支持作为独立的文件系统来处理。并将具有不同能力的其它类型文件管理支持也作为独立的文件系统来处理。

要查看每个文件系统的特征和限制的比较，参见第 51 页的『文件系统比较』。

集成文件系统中的文件系统是:

“根”（/）
“根”（/）文件系统。此文件系统充分利用了集成文件系统的流文件支持和分层目录结构。根文件系统具有磁盘操作系统（DOS）和 OS/2 文件系统的特征。

QOpenSys
开放系统文件系统。此文件系统与基于 UNIX 的开放系统标准（如 POSIX 和 XPG）兼容。像根文件系统一样，此文件系统也利用集成文件系统提供的流文件和目录支持。而且，它还支持区分大小写的对象名。

UDFS 用户定义文件系统。此文件系统驻留在用户选择的辅助存储池（ASP）或独立辅助存储池（ASP）上。创建和管理此文件系统。

QSYS.LIB
库文件系统。此文件系统支持服务器的库结构。此文件系统允许您访问系统和基本用户 ASP 中库支持所管理的数据库文件和所有其它 iSeries 服务器对象类型。

独立 ASP QSYS.LIB
“独立 ASP QSYS.LIB”文件系统。此文件系统支持您创建和定义的任何独立辅助存储池（ASP）中的服务器库结构。此文件系统允许您访问库支持所管理的数据库文件和所有其它 iSeries 服务器对象类型。

QDLS 文档库服务文件系统。此文件系统提供对文档和文件夹的访问。

QOPT
光盘文件系统。此文件系统提供对存储在光盘介质上的流数据的访问。

QNetWare
QNetWare 文件系统。此文件系统允许您访问存储在运行 Novell NetWare 4.10 或 4.11 的服务器上的本地或远程数据和对象或访问运行 Novell NetWare 3.12、4.10、4.11 或 5.0 的独立“PC 服务器”。您可以在现有的本地文件系统上动态安装 NetWare 文件系统。

QNTC Windows NT 服务器文件系统。此文件系统提供对一些数据和对象的访问，这些数据和对象存储在运行 Windows NT 4.0 或更高版本的服务器上。它允许 iSeries 服务器应用程序使用与 Windows NT 客户

机相同的数据。这包括访问正在“集成 PC 服务器”上运行的“Windows NT 服务器”上的数据。有关详细信息，参见 OS/400-AS/400 Integration with Windows NT Server, SC41-5439-01 (SC41-5439)。

QFileSvr.400

此文件系统提供对驻留在远程 iSeries 服务器上的其它文件系统的访问。

NFS 网络文件系统。此文件系统允许您访问存储在远程 NFS 服务器上的数据和对象。NFS 服务器可以导出一个网络文件系统，然后 NFS 客户机将动态安装该网络文件系统。

可以通过公共接口与任何文件系统进行交互。与通过数据管理接口提供的记录输入／输出相反，此接口最适合于流数据的输入／输出。提供的命令、菜单、屏幕和应用程序接口（API）允许通过此公共接口与文件系统进行交互。

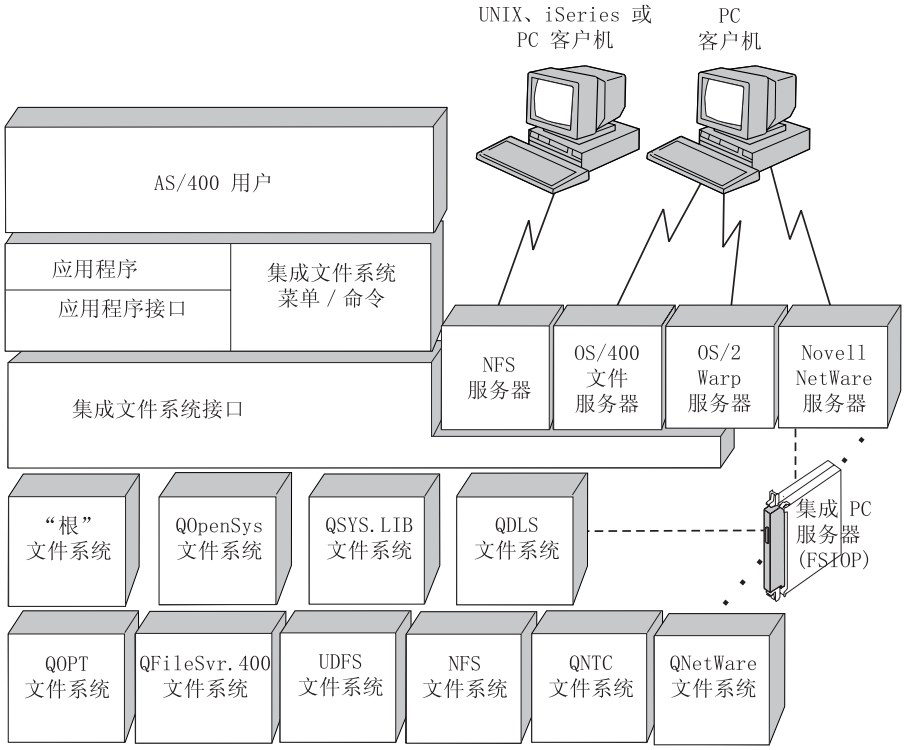


图 4. 文件系统、文件服务器和集成文件系统接口

有关更多信息，参见下列主题和出版物:

- Optical Support
- OS/400 Network File System Support

目录

目录是一个特殊对象，用来按您所指定的名称来定位对象。每个目录包含一个连接到它的对象的列表。该列表可以包括其它目录。

集成文件系统提供允许您访问服务器中所有对象的分层目录结构。可以将此目录结构看成是一个倒立的树，其中根在顶部，分支在下面。分支表示目录层次结构中的目录。这些目录分支具有附属分支，它们称为子目录。

连接到不同目录和子目录分支的是诸如文件的对象。定位一个对象需要指定该对象所连接的子目录的目录路径。连接到特定目录的对象有时描述为“在”那个目录中。

特定目录分支及其所有附属分支（子目录）和连接到那些分支的所有对象称为**子树**。每个文件系统是集成文件系统目录结构中的一个主要的子树。在 QSYS.LIB 和“独立 ASP QSYS.LIB”文件系统的子树中，库的处理方式与子目录的处理方式相同。库中的对象与子目录中的对象的处理方式也是相同的。因为数据库文件包含对象（数据库文件成员），所以它们的处理方式与子目录一样，而不是与对象一样。在文档库服务文件系统（QDLS 子树）中，文件夹的处理方式与子目录一样，而文件夹中的文档的处理方式则与子目录中的对象一样。

由于文件系统的差异，可以在目录层次结构的一个子树中执行的操作不一定能在另一个子树中执行。

集成文件系统目录支持与 DOS 文件系统提供的目录支持类似。而且，它还提供具有 UNIX 系统特征的功能部件，如存储文件一次但可以使用链接通过多个路径访问该文件的能力。

有关集成文件系统目录的更多信息，参考下列主题:

- 第 8 页的『当前目录和主目录』
- 第 29 页的『集成文件系统提供的目录』
- 第 8 页的『*TYPE2 目录』

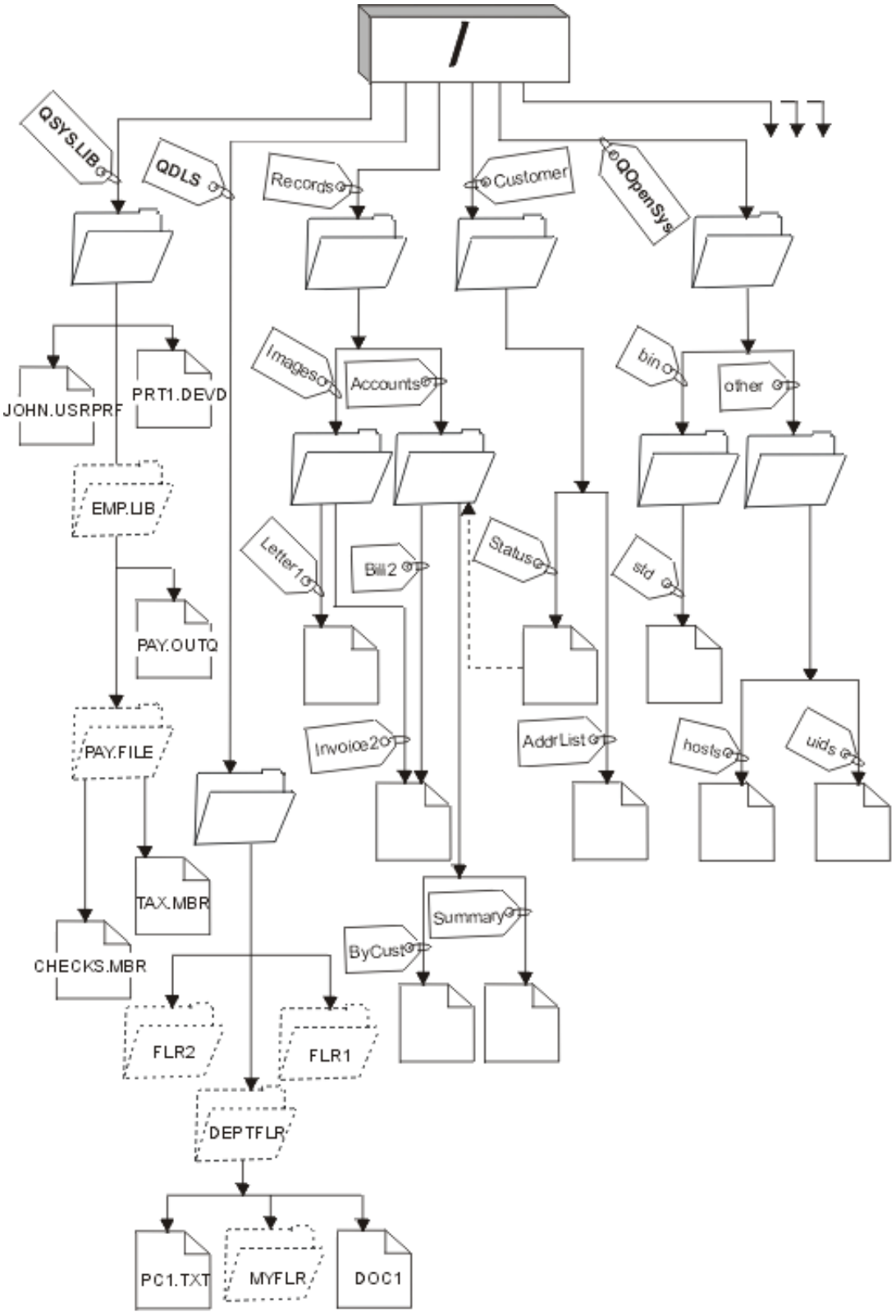


图 5. 文件系统和对象是集成文件系统目录树上的分支

当前目录和主目录

当前目录是操作系统在其中查找程序和文件并存储临时文件和输出的第一个目录。在您请求对对象（如文件）执行操作时，除非指定了一个不同的目录路径，否则系统将在当前目录中搜索该对象。当前目录与当前库的概念类似。它还称为**当前工作目录**，或就称为**工作目录**。

当您注册到系统时，**主目录**用作当前目录。主目录的名称在用户概要文件中指定。当启动作业时，系统在您的用户概要文件中查找主目录的名称。若系统上不存在那个名称的目录，则主目录更改为“根”（/）目录。

典型情况下，为用户创建用户概要文件的系统管理员还将创建用户的主目录。建议在 /home 目录下为每个用户创建各自的主目录。/home 目录是“根”（/）目录下的子目录。系统缺省值预计用户主目录的名称与用户概要文件的名称相同。

例如，命令 CRTUSRPRF USRPRF(John) HOMEDIR(*USRPRF) 将 John 的主目录指定为 /home/JOHN。若 /home/JOHN 不存在，则根（/）目录成为 John 的主目录。

| 通过使用更改当前目录（CHGCURDIR）CL 命令、chdir( ) API 或 fchdir() API，可以在注册后的任何时间指
| 定非主目录的目录作为当前目录。

缺省情况下，在进程启动期间选择的主目录将保持为每个线程的主目录。这与启动之后线程的活动用户概要文件是否更改无关。但是，有“更改作业”（QWTCHGJB）API 提供的支持可用来将线程正在使用的主目录更改为线程当前用户概要文件的主目录（或者，若该主目录不存在，则为“根”（/）目录）。辅助线程将总是继承创建它的线程的主目录。注意：当使用 QWTCHGJB 来更改线程的主目录时，进程的当前目录不会更改。当前目录的作用域为进程级，而主目录的作用域为线程级。在任何线程中更改当前工作目录会在整个进程中更改该工作目录。更改线程的主目录不会更改它的当前工作目录。

有关 QWTCHGJB API 的详细信息，参见应用程序编程接口（API）主题。

| *TYPE2 目录

| 集成文件系统中的“根”（/）、QOpenSys 和用户定义的文件系统（UDFS）支持 *TYPE2 目录格式。*TYPE2
| 目录格式是最初的 *TYPE1 目录格式的增强。*TYPE2 目录具有与 *TYPE1 目录的不同内部结构和不同实现。

| *TYPE2 目录的优点是:

- 提高了性能
- 提高了可靠性
- 增加了功能
- 较少的辅助存储空间（在许多情况下）。

| *TYPE2 目录与 *TYPE1 目录相比提高了文件系统性能，特别是当创建和删除目录时。

| *TYPE2 目录比 *TYPE1 目录更可靠。系统异常结束后，除非存在辅助存储器故障，否则完全恢复 *TYPE2 目
| 录。*TYPE1 目录可能需要使用回收存储器（RCLSTG）命令才能完全恢复。

| *TYPE2 目录提供以下增加的功能:

1. *TYPE2 目录支持重命名单字体文件系统中名称的大小写（例如，从 A 重命名为 a）。
2. 与 *TYPE1 目录的 32,767 个链接相比，*TYPE2 目录中的对象可以具有多达一百万个链接。这意味着可以具有一个流文件的多达一百万个硬链接，且 *TYPE2 目录可以包含多达一百万个子目录。
3. 使用“iSeries 导航器”，当打开具有 *TYPE 格式的目录时，自动按二进制次序排序项列表。

通常，对象数少于 350 个的 *TYPE2 目录需要的辅助存储器比具有相同对象数的 *TYPE1 目录需要的少。对象数多于 350 个的 *TYPE2 目录比 *TYPE1 目录大 10%（平均起来）。

系统上有几种方式获取 *TYPE2 目录:

- 在第一次将独立 ASP 与安装了 OS/400 V5R2 的系统联机时，将独立辅助存储池（ASP）中用户定义的文件系统（UDFS）转换为 *TYPE2 格式。
- 必须通过使用转换目录（CVTDIR）命令，将独立 ASP 中 UDFS 以外的所有其它受支持的文件系统转换为 *TYPE2。
- 预装入了 OS/400 V5R2 的新的 iSeries 服务器具有 *TYPE2 目录。不需要对 ASP 1-32 中的“根”（/）、QOpenSys 和 UDFS 进行转换。
- 在 iSeries 服务器上临时安装的 OS/400 V5R2 具有 *TYPE2 目录。不需要对 ASP 1-32 中的“根”（/）、QOpenSys 和 UDFS 进行转换。

要确定服务器上文件系统的目录格式，使用转换目录（CVTDIR）命令:

`CVTDIR OPTION(*CHECK)`。

注: *TYPE2 目录在 OS/400 V5R1 上受支持，但与正常的 *TYPE2 目录支持有一些差别。有关更多信息，参见在 OS/400 V5R1 中使用 *TYPE2 目录。

有关 *TYPE2 目录的更多信息，参考下列主题:

- 转换为 *TYPE2 目录
- “根”、QOpenSys 或 UDFS 不可用性
- 辅助存储器需求
- 符号链接注意事项
- 独立辅助存储池（ASP）
- 保存 / 恢复注意事项
- 准备 *TYPE2 转换
- 转换处理
- 示例: 转换所有文件系统（少量对象）
- 示例: 转换所有文件系统（大量对象）
- 示例: 仅转换某些 ASP

在 OS/400 V5R1 中使用 *TYPE2 目录

集成文件系统中的“根”（/）、QOpenSys 和用户定义的文件系统（UDFS）支持 OS/400 V5R1 中的 *TYPE2 目录格式。*TYPE2 目录格式是最初的 *TYPE1 目录格式的增强。*TYPE2 目录具有与 *TYPE1 目录不同的内部结构并提供改进的性能和可靠性。

如果您具有 V5R1，可以将 V5R1 目录转换为 *TYPE2 目录格式。建议在安装 OS/400 的新发行版**之前**转换为 *TYPE2 目录格式。这是必要的，因为在安装期间可能会自动执行目录转换。在安装期间自动转换的影响是将会**显著**增加安装所需要的时间。

注: 如果升级到 OS/400 V5R1 或 V5R2，**不**会自动转换为 *TYPE2 目录格式。在执行这些安装之前不需要转换目录。

V5R1 中 *TYPE2 目录的支持可通过修订（PTF）获得。转换实用程序与 V5R2 版本稍微有些不同。有关 V5R1 中 *TYPE2 目录的完整文档，参考信息性 APAR II13161。使用下列方法之一来访问 APAR:

1. 将信息性 APAR 下载到您的 iSeries 服务器并查看它。使用下列命令:

```
SNDPTFORD PTFID((II13161))
DSPPTFCVR LICPGM(INFOAS4) SELECT(II13161)
```

2. 转至 http://www-912.ibm.com Web 站点来在线查看信息性 APAR。选择**授权程序分析报告（APAR）—> V5R1 APAR —> APAR 编号 II13161**。

转换为 *TYPE2 目录

CVTDIR 命令执行从 *TYPE1 目录到 *TYPE2 目录的转换。另外，它提供有关如何将文件系统转换为 *TYPE2 目录格式的信息。CVTDIR 执行下列操作:

- 列示支持 *TYPE2 目录的现有文件系统的当前目录格式。
- 估计执行转换将花费的时间。
- 估计转换的辅助存储器需求。
- 将文件系统转换为 *TYPE2 格式。将任何现有目录转换为 *TYPE2，且在转换之后创建的任何新目录为 *TYPE2。

转换文件系统之一中的目录的方式有以下几种:

- 手工转换，通过使用 CVTDIR 命令
- 自动转换，在第一次将独立 ASP 与已安装 OS/400 V5R2 的系统联机时
- 在 IPL 期间，如果系统确定在系统异常结束时正在进行文件系统转换
- 在回收存储器（RCLSTG SELECT(*ALL)）期间，如果发现丢失的 *TYPE1 目录是转换为 *TYPE2 格式的文件系统的一部分

“根”、QOpenSys 或 UDFS 不可用性

必须当系统处于受限制状态时执行“根（/）或 QOpenSys 文件系统的转换。当转换 UDFS 时，不要求系统处于受限制状态；但是，在转换期间，该 ASP 中的 UDFS 不可用。执行转换所需要的时间长度取决于文件系统的大小。因此，有必要进行计划，以便调度最合适的时间来执行转换。CVTDIR 命令的 *ESTIMATE 选项估计转换指定的文件系统所需要的时间长度。估计的时间长度是最大估计值。它根据具有单个线程的作业中运行的转换估计时间长度。实际转换使用多个线程，因此花费的时间应该比估计的时间少。通常，可以在估计时间的 30% 到 50% 范围内转换链接数大于 40,000 的文件系统。但是，实际时间取决于服务器的硬件和配置。

如果系统在 CVTDIR 命令正在运行时异常结束，则在后续 IPL 期间，转换功能在 SCPF 作业中运行。SCPF 作业不允许多个线程为活动的。因此，当必须在该 IPL 期间完成文件系统的转换时，转换使用单个线程运行。当 IPL 期间显示 SRC C900 2A85 时，转换功能运行，且显示状态消息 CPIA089，指示转换的进度。

如果在已转换为 *TYPE2 的文件系统中有丢失的 *TYPE1 目录，转换功能在 RCLSTG 期间运行。转换功能在发出 RCLSTG 命令的作业中运行。如果发现需要转换任何丢失的目录，则转换由于系统限制而在单个线程中运行。

辅助存储器需求

将文件系统中的目录转换为 *TYPE2 格式之前，应该考虑辅助存储器需求。有几个关于辅助存储器需求的问题:

- 将目录转换为 *TYPE2 格式之后目录最终的大小
- 转换功能运行时所需要的附加存储器

在许多情况下，*TYPE2 目录最终的大小比 *TYPE1 目录小。通常，对象数少于 350 个的 *TYPE2 目录需要的辅助存储器比具有相同对象数的 *TYPE1 目录需要的少。对象数多于 350 个的 *TYPE2 目录比 *TYPE1 目录大 10%（平均起来）。

| 当转换功能运行时，需要附加存储器。转换功能要求几个目录同时存在 *TYPE1 版本和 *TYPE2 版本。此数
| 目取决于 iSeries 服务器配置和正在转换的文件系统的目录结构。

| CVTDIR 命令中的 *ESTIMATE 选项将提供指示估计在转换期间所需要的辅助存储量的信息。

| 符号链接注意事项

| 符号链接是集成文件系统中包含另一个对象的路径的对象。当可能更改对象的名称时，转换期间有一些实例。
| 如果在转换期间重命名符号链接内路径的元素之一，则符号链接的内容不再指向该对象。有关对象重命名的详
| 细信息，参见重命名的对象。

| 独立辅助存储池（ASP）

| 在第一次将独立 ASP 与安装了 OS/400 V5R2 的系统联机时，将目录转换为 *TYPE2。为了进行计划，在
| OS/400 V5R1 中提供了估计功能，以提供关于转换将运行的时间长度的信息。在将独立 ASP 与 V5R2 服务器
| 联机之前，当使独立 ASP（命名为 ASP_NAME）联机并活动时，在您 V5R1 系统上运行下列 API:

```
| CALL QP0FCVT2 (*ESTIMATE ASP_NAME *TYPE2)
```

| **注:** 建议在调用此功能之前在 V5R1 系统上的独立 ASP 上运行 RCLSTG。

| 保存／恢复注意事项

| 可以在已转换为 *TYPE2 的文件系统中保存和恢复以 *TYPE1 格式存在的目录。同样，只要当目录以 *TYPE2
| 目录格式存在时未超过 *TYPE1 限制，就可以在 *TYPE1 格式的文件系统中保存和恢复以 *TYPE2 格式存在
| 的目录。

| 准备 *TYPE2 转换

| 有几个建议在转换为 *TYPE2 目录前使用的 CL 命令和参数:

| • 回收存储器（RCLSTG）

  | 在转换任何文件系统之前，使用 RCLSTG SELECT(*ALL) 命令清理目录并确保目录完好。尽管这并不会排除目
  | 录转换期间可能遇到的所有可能的问题，但它确保可以读取文件系统中的目录。

  | 在使用 CVTDIR 命令的任何选项之前，只需要运行此命令一次。

| • 保存系统（SAVSYS）

  | 在执行 RCLSTG 之后和使用 CVTDIR 命令的 *CONVERT 选项之前，应该执行 iSeries 服务器的整个系统保
  | 存。要备份系统，使用 iSeries 上的“保存”菜单。要进入“保存”菜单，在任何命令行输入 GO SAVE，并
  | 选择选项 21。有关更多信息，参见 Backup and Recovery  。

  | 在使用 CVTDIR 命令的任何选项之前，只需要运行此命令一次。

| • CVTDIR 命令的 *ESTIMATE 选项

  | 使用 CVTDIR 命令的 *ESTIMATE 选项来确定转换目录所需要的时间。

  | 除了提供时间和辅助存储器估计外，*ESTIMATE 选项还有另外的好处。它构建一些与 *TYPE1 目录相关联的
  | 辅助对象，这使转换运行速度加快（因为不需要创建这些对象）。在使用 *CONVERT 选项将文件系统转换为
  | *TYPE2 格式之后，这些辅助对象才不会存在。*ESTIMATE 选项还读取文件系统中的所有目录，这会隐式验
  | 证目录。*ESTIMATE 选项并不保证查找实际转换期间可能发生的所有可能的错误，但它有助于查找错误。

  | 在运行 *ESTIMATE 选项之后，检查作业记录中的错误，并在转换文件系统之前执行任何建议的恢复操作。不
  | 需要在执行建议的恢复操作之后再次运行估计，但可能期望这样做以便验证找不到其它问题。

| • 辅助存储器注意事项

为包含正在转换的文件系统的 ASP 检查可用辅助存储器。CVTDIR *ESTIMATE 选项显示消息 CPIA090，它指示有多少辅助存储器可用于 ASP。另外，它还显示在转换期间期望使用的辅助存储量。还显示消息 CPIA091，它指示估计转换之后文件系统中 *TYPE2 目录的总的大小是否大于或小于现有的 *TYPE1 目录。ASP 中可用的存储器应该是未使用的辅助存储器（在消息 CPIA090 中所显示的）与 *TYPE1 目录大小和 *TYPE2 目录大小之间的差（在消息 CPIA091中所显示的）的总和。

或者，可以使用启动系统服务工具（STRSST）命令并选择使用磁盘单元选项来查找可用的辅助存储器。

注: 如果在系统上仅定义了一个 ASP，则“使用系统状态”（WRKSYSSTS）命令足以显示可用的辅助存储器信息。

最好在使用 CVTDIR 命令中的任何选项之前执行一般系统清理。如果有任何不再需要的目录或文件，则在使用 CVTDIR 命令的任何选项之前除去它们。这样做会释放辅助存储空间，提供可用辅助存储空间的更精确估计，并因为需要处理的对象较少而允许在较短的时间内完成转换。

- 对于正发出 CVTDIR 命令的作业，考虑将作业消息队列已满操作更改为 *PRTWRAP。为此，执行以下操作:
  1. 防止在作业记录已满的情况下作业异常结束
  2. 如果作业记录执行回绕，将任何已覆盖的消息打印至假脱机文件；因而，不会丢失重要消息
- 在具有独立 ASP 的系统上: 在将独立 ASP 与正在运行 OS/400 V5R2 的系统联机之前，在所有独立 ASP 上使用 V5R1 *ESTIMATE 功能。这可提供在安装之后独立 ASP 的第一次联机将花费多长时间的时间估计。有关更多信息，参见独立辅助存储池（ASP）。

转换处理

CVTDIR 命令将 *TYPE1 目录转换为 *TYPE2 目录。在转换过程期间有几件事情要考虑:

- 转换“根”（/）或 QOpenSys
- 转换用户定义的文件系统
- 创建用户概要文件
- 重命名的对象
- 用户概要文件注意事项

转换“根”或 QOpenSys

当转换“根”或 QOpenSys 文件系统时，系统**必须**处于受限制状态。在转换期间不能使用任一个文件系统。CVTDIR 命令卸装所有 UDFS 和 NFS 文件系统，并且当转换完成时，不重新安装它们。可以使用“安装”命令（MOUNT）来重新安装 UDFS 或 NFS 文件系统。

转换用户定义的文件系统

当转换 ASP 1-32 中的 UDFS 时，它们不可用于使用系统的任何用户。对于这些 ASP 中的每一个，在 /dev 目录中有一个 QASPxx 目录。当 CVTDIR 命令运行时，它从名称空间中除去 QASPxx 目录，以防止任何用户访问 ASP 中的 UDFS。当 CVTDIR 命令完成处理所有对象（包括 QASPxx 目录）时，将对象放回名称空间中并使它们对系统上的用户可用。CVTDIR 命令卸装 ASP 的 UDFS，并且当转换完成时，不重新安装它们。可以使用“安装”命令（MOUNT）来重新安装 UDFS。

注: QASP01 目录存在于每个系统上。

创建用户概要文件

转换功能创建当转换功能正在运行时所使用的用户概要文件。这些用户概要文件名称为 QP0FCVxxxx，其中，xxxx 是编号，如 0001。如果最初的所有者无法拥有它们的目录，则转换功能使用这些用户概要文件来拥有所转换的文件系统中的目录。

| 如果有可能，当转换完成时，删除这些用户概要文件。如果将目录的所有权授予这些用户概要文件之一，则发送消息 CPIA08B。

重命名的对象

*TYPE2 目录要求链接名称是有效的 UTF-16 名称。这不同于 *TYPE1 目录，该目录具有 UCS2 Level 1 名称。因此，在目录转换期间，可能会发现无效或重复的名称。当发现名称无效或重复时，将该名称更改为唯一且有效的 UTF-16 名称，且将消息 CPIA08A 发送至作业记录中，并列示原始名称和新名称。名称中包含的组合字符或无效替代字符对可能会导致重命名对象。

有关 UTF-16 的更多信息，请参考 Unicode 主页（http://www.unicode.org  ）。

组合字符: 某些字符可能由多个 Unicode 字符组成。例如，带重音或变音符号的字符。在将这些字符存储在目录中之前，需要将它们更改（即标准化）为公共格式，以便所有对象都具有唯一名称。将组合字符标准化是一种过程，通过该过程将字符变成已知的和可预测的格式。为 *TYPE2 目录选择的格式是一种规范的组成格式。如果在一个 *TYPE1 目录中有两个对象包含同一组合字符，则会将它们标准化为同一名称。这会导致冲突，即使一个对象包含组成的组合字符，而另一个对象包含已分解的组合字符。因此，在 *TYPE2 目录中链接它们其中之一前，已更改其名称。

替代字符: 某些字符不具有 Unicode 格式的有效表示。这些字符具有某些特殊值；它们由两个特定范围中的两个 Unicode 字符组成，使第一个 Unicode 字符在一个范围内（例如 0xD800-0xD8FF）而第二个 Unicode 字符位于另一个范围内（例如 0xDC00-0xDCFF）。这称为一个替代对。如果 Unicode 字符之一丢失或如果它们的次序不对，（仅部分字符），则它是无效名称。在 *TYPE1 目录中允许此类型的名称，但在 *TYPE2 目录中不允许。为了使转换功能继续，如果发现包含这些无效名称之一的名称，则在将对象链接到 *TYPE2 目录之前，更改名称。

用户概要文件注意事项

当转换正在运行时，进行各种尝试来确保拥有任何 *TYPE1 目录的相同用户概要文件继续拥有对应的 *TYPE2 目录。因为 *TYPE1 和 *TYPE2 目录瞬间同时存在，这会影响用户概要文件所拥有的存储量和用户概要文件中的项数。

更改用户概要文件的最大存储量: 在目录转换处理期间，有几个目录瞬间同时以两种格式存在并由同一用户概要文件拥有。如果由于达到了用户概要文件的最大存储量限制而不能创建 *TYPE2 目录，则增加用户概要文件的限制。将消息 CPIA08C 发送至作业记录，且转换继续。

更改目录的所有者: 如果拥有 *TYPE1 目录的用户概要文件无法拥有创建的 *TYPE2 目录，则将 *TYPE2 目录的所有者设置为创建用户概要文件中所描述的备用用户概要文件之一。将消息 CPIA08B 发送至作业记录，且转换继续。

示例: 转换所有文件系统（少量对象）

系统 A 配置了 5 个辅助存储池（ASP）：1（系统 ASP）、3、5、11 和 25。尚未将系统上任何文件系统从 *TYPE1 目录转换为 *TYPE2 目录。想要转换所有文件系统。文件系统不包含大量对象，因此计划在一天内执行所有步骤。

要转换具有少量对象的所有文件系统中的目录:

1. 使系统处于受限制状态。
2. 在命令行输入 `RCLSTG SELECT(*ALL)`。
3. 使用“保存”菜单保存系统。在命令行输入 `GO SAVE`，并选择选项 21。
4. 在命令行输入 `CVTDIR OPTION(*ESTIMATE) FILESYS(*ALL) FORMAT(*TYPE2)`。
5. 检查来自 *ESTIMATE 功能的任何错误消息。

6. 验证所有 ASP 是否具有足够可用的辅助存储空间。
7. 在命令行输入 CVTDIR OPTION(*CONVERT) FILESYS(*ALL) FORMAT(*TYPE2)。

   注: 当转换所有文件系统（*ALL）时，显示消息 CPAA084，并要求您验证是否要转换所列示的文件系统。
8. 检查来自 *CONVERT 功能的任何错误消息。
9. 使系统脱离受限制状态。

示例: 转换所有文件系统（大量对象）

系统 B 也配置了 5 个辅助存储池（ASP）: 1（系统 ASP）、3、5、11 和 25。尚未将系统上任何文件系统从 *TYPE1 目录转换为 *TYPE2 目录。想要转换所有文件系统。文件系统包含大量对象，因此计划在两个不同的周末执行转换步骤。

第一个周末:

1. 使系统处于受限制状态。
2. 在命令行输入 RCLSTG SELECT(*ALL)。
3. 使用“保存”菜单保存系统。在命令行输入 GO SAVE，并选择选项 21。
4. 使系统脱离受限制状态。

在一周的工作日内:

5. 在命令行输入 CVTDIR OPTION(*ESTIMATE) FILESYS(*ALL) FORMAT(*TYPE2)。
6. 检查来自 *ESTIMATE 功能的任何错误消息。
7. 验证所有 ASP 是否具有足够可用的辅助存储空间。

第二个周末:

8. 使系统处于受限制状态
9. 在命令行输入 CVTDIR OPTION(*CONVERT) FILESYS(*ALL) FORMAT(*TYPE2)。

   注: 当转换所有文件系统（*ALL）时，显示消息 CPAA084，并要求您验证是否要转换所列示的文件系统。
10. 检查来自 *CONVERT 功能的任何错误消息。
11. 使系统脱离受限制状态。

示例: 仅转换某些 ASP

系统 C 配置了 6 个辅助存储池（ASP）: 1（系统 ASP）、2、4、8、10 和 30。尚未转换系统上的任何文件系统。只想转换 ASP 4、10 和 30 中的 UDFS。

要转换某些 ASP 中的 UDFS 中的目录:

1. 验证文件系统的目录格式。为此，在命令行中输入 CVTDIR OPTION(*CHECK)。
2. 使系统处于受限制状态。
3. 在命令行输入 RCLSTG SELECT(*ALL)。
4. 使用“保存”菜单保存系统。在命令行输入 GO SAVE，并选择选项 21。
5. 将系统从受限制状态释放。
6. 在命令行输入 CVTDIR OPTION(*ESTIMATE) FILESYS(*UDFS) ASP(4) FORMAT(*TYPE2)。
7. 检查来自 *ESTIMATE 功能的任何错误消息。
8. 在命令行输入 CVTDIR OPTION(*ESTIMATE) FILESYS(*UDFS) ASP(10) FORMAT(*TYPE2)。

9. 检查来自 *ESTIMATE 功能的任何错误消息。
10. 在命令行输入 `CVTDIR OPTION(*ESTIMATE) FILESYS(*UDFS) ASP(30) FORMAT(*TYPE2)`。
11. 检查来自 *ESTIMATE 功能的任何错误消息。
12. 验证所有 ASP 是否具有足够可用的辅助存储空间。
13. 在命令行输入 `CVTDIR OPTION(*CONVERT) FILESYS(*UDFS) ASP(4) FORMAT(*TYPE2)`。
14. 检查来自 *CONVERT 功能的任何错误消息。
15. 在命令行输入 `CVTDIR OPTION(*CONVERT) FILESYS(*UDFS) ASP(10) FORMAT(*TYPE2)`。
16. 检查来自 *CONVERT 功能的任何错误消息。
17. 在命令行输入 `CVTDIR OPTION(*CONVERT) FILESYS(*UDFS) ASP(30) FORMAT(*TYPE2)`。
18. 检查来自 *CONVERT 功能的任何错误消息。

路径名

路径名（还在某些系统上称为 **pathname** ）告诉服务器如何定位对象。路径名表示为一系列目录名后跟该对象的名称。各个目录和对象名用斜杠（/）字符隔开; 例如:

```
directory1/directory2/file
```

为了方便起见，可以在集成文件系统命令中使用反斜杠（\）来代替斜杠。

有两种指示路径名的方法:

- **绝对路径名**从最高级即“根”目录（它由 / 字符标识）开始。例如，考虑从 / 目录到名为 Smith 的文件的下列路径。

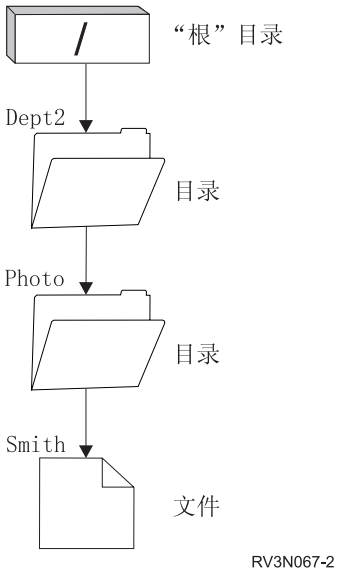


图 6. 路径名的各个部分

Smith 文件的绝对路径名如下所示:

```
/Dept2/Photo/Smith
```

绝对路径名也称为**全路径名**。

- 若路径名不以 / 字符开始，则系统假定路径从当前目录开始。这种类型的路径名称为**相对路径名**。例如，若当前路径为 Dept2，并且它具有包含文件 Smith 的子目录 Photo，则该文件的相对路径名为:

  ```
  Photo/Smith
  ```

  注意: 该路径名不包括当前目录的名称。路径名中的第一项是当前目录之*下的下一级*目录或对象。

链接

链接是目录和对象之间的命名连接。用户或程序可以通过指定对象链接的名称告诉服务器在何处查找对象。链接可用作路径名或路径名的一部分。

对于基于目录的文件系统的用户，不妨将对象（例如文件）视为具有名称的某些事物，该名称向服务器标识该事物。事实上，是对象的目录路径标识对象。有时可以通过只给出对象的“名称”来访问对象。您之所以可以这样做，只是因为系统设计成在某些情况下假定了路径的目录部分。链接概念利用了这样一个事实，即目录路径标识了对象。对链接而不是对象指定名称。

一旦习惯于链接有名称而不是对象有名称这样的概念，就会开始看到以前隐藏的可能性。可以存在到同一对象的多个链接。例如，两个用户可以通过从每个用户的主目录获取一个到文件的链接来共享该文件（参见第 8 页的『当前目录和主目录』）。某些类型的链接可以跨越文件系统，并可以在对象不存在的情况下存在。

有两种类型的链接: **硬链接**和**符号链接**。

有关链接的更多信息，参考下列主题:

- 硬链接
- 符号链接
- 比较: 硬链接和符号链接

硬链接

硬链接有时就称为链接，除非将它链接到一个实际的对象，否则它不能存在。在目录中创建对象时（例如，通过将文件复制到目录），就在目录和对象之间建立了第一个硬链接。用户和应用程序可以添加其它硬链接。每个硬链接由目录中的一个独立目录项指示。来自同一目录的链接不能具有相同的名称，但来自不同目录的链接可以具有相同的名称。

若文件系统支持，则可以有到一个对象的多个硬链接，或者来自同一目录，或者来自不同目录。例外情况是对象是另一个目录。从一个目录到另一个目录，只能有一个硬链接。

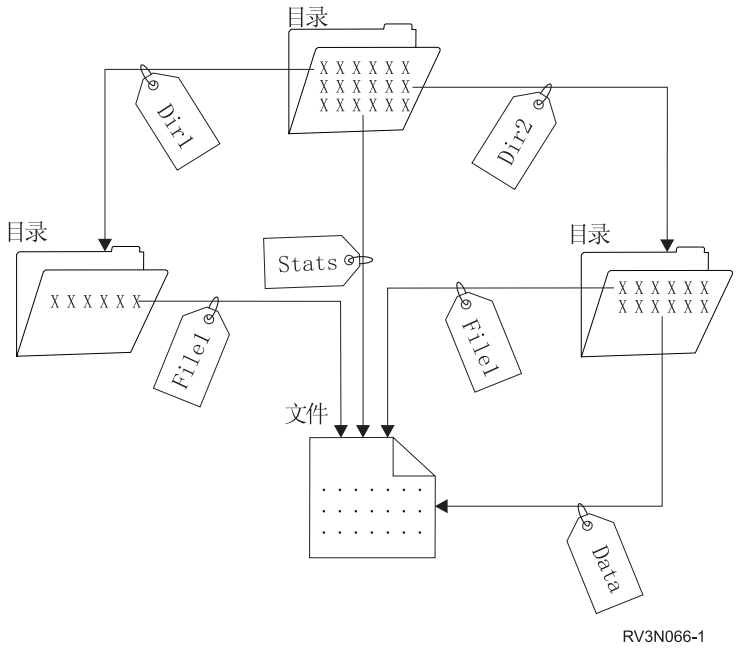


图 7. 目录项定义每个硬链接。

可以除去硬链接而不影响对象的存在性，只要至少保留一个到该对象的硬链接。当除去最后一个硬链接时，除非应用程序打开了该对象，否则也会从服务器中除去该对象。打开该对象的每个应用程序可以继续使用它，直到该应用程序关闭它为止。当最后一个使用对象的应用程序关闭它时，便会从服务器中除去它。在除去最后一个硬链接之后，不能打开对象。

| 硬链接的概念也可以应用于 QSYS.LIB 或“独立 ASP QSYS.LIB”文件系统和文档库服务（QDLS）文件系统，但有限制。事实上，对于库中每个对象，库都有一个到它的硬链接。同样，对于文件夹中每个文档，文件夹都有一个到它的硬链接。但是，在 QSYS.LIB、“独立 ASP QSYS.LIB”或 QDLS 中不允许到*同一对象*的多个硬链接。

| 硬链接不能跨越文件系统。例如，QOpenSys 文件系统中的目录不能具有到 QSYS.LIB 或“独立 ASP QSYS.LIB”文件系统中对象的硬链接或到 QDLS 文件系统中文档的硬链接。

符号链接

符号链接也称为软链接，它是包含在文件中的路径名。当系统遇到符号链接时，它沿着符号链接提供的路径名前行，然后继续沿着符号链接后面的任何其余路径前行。如果路径名以一个 / 开始，则系统返回到 /（“根”）目录，并从该目录开始沿着路径前行。如果路径名未以 / 开始，则系统返回到前一级目录，并从那个目录开始沿着符号链接中的路径名前行。

考虑关于如何使用符号链接的下列示例:

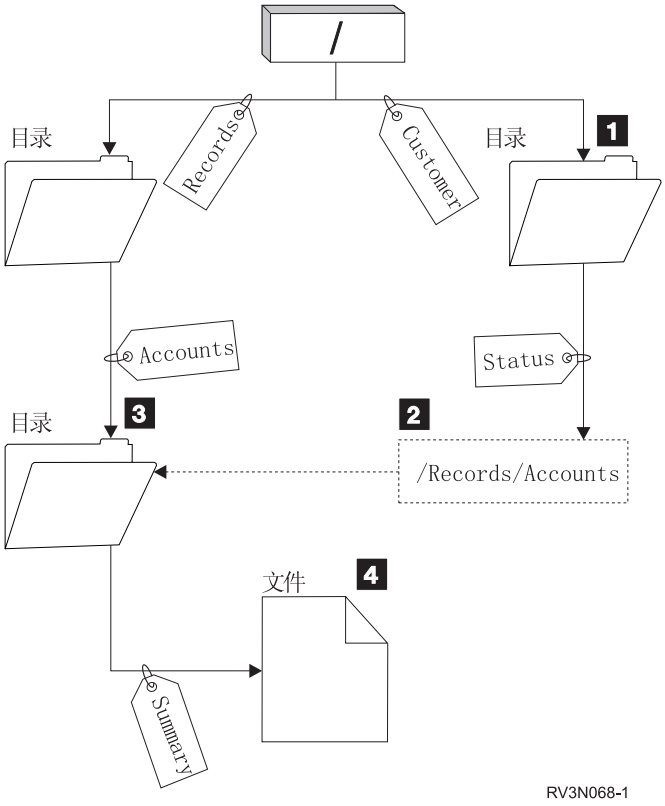


图 8. 使用符号链接的示例

选择一个菜单选项来显示客户帐号的状态。显示菜单的程序使用下列路径名:

```
/Customer/Status/Summary
```

系统沿着 *Customer* 链接前行，该链接导向目录 **1**，然后沿着 *Status* 链接前行。*Status* 链接是一个符号链接，它包含路径名 **2**。由于该路径名以 / 开始，系统返回到 /（“根”）目录，并依次沿着链接 *Records* 和 *Accounts* 前行。此路径导向另一个目录 **3**。现在，系统完成了程序提供的路径名中的路径。它沿着*Summary* 链接前行，该链接导向包含您将需要的数据的文件 **4**。

与硬链接不同，符号链接是一个对象（对象类型为 *SYMLNK）；它可以存在而不必指向存在的对象。例如，可以使用符号链接来提供以后将添加或替换的文件的路径。

| 与硬链接不同，符号链接还可以跨越文件系统。例如，如果您正在一个文件系统中工作，可以使用符号链接来
| 访问另一个文件系统中的文件。虽然 QSYS.LIB、“独立 ASP QSYS.LIB” 和 QDLS 文件系统不支持创建和
| 存储符号链接，但您可以在“根”（/）或 QOpenSys 文件系统中创建符号链接，以便:

| • 访问 QSYS.LIB 或“独立 ASP QSYS.LIB” 文件系统中的数据库文件成员。
| • 在 QDLS 文件系统中访问文档。

另见『比较: 硬链接和符号链接』。

比较: 硬链接和符号链接

在程序中使用路径名时，可以选择使用硬链接或符号链接（参见第 16 页的『链接』）。每种类型的链接各有优点和缺点。一种类型的链接优于另一类型的一些情况如下:

表 1. 比较硬链接和符号链接

项	硬链接	符号链接
名称解析	较快。硬链接包含对象的直接引用。	较慢。符号链接包含对象的路径名，必须解析该路径名才能找到对象。
对象存在性	必需的。对象必须存在，才能创建到对象的硬链接。	可选的。可以在引用的对象不存在时创建符号链接。
对象删除	受限制。必须取消（除去）到对象的硬链接才能删除对象。	不受限制。即使引用对象的符号链接存在，也可以删除对象。
动态对象（属性更改）	较慢。对象的很多属性存储在每个硬链接中。因此，当到动态对象的硬链接数目增加时，则对该对象的更改较慢。	较快。对动态对象的更改不受符号链接的影响。
静态对象（属性不更改）	较快。对于静态对象，名称解析是性能上主要关心的问题。使用硬链接时，名称解析较快。	较慢。使用符号链接时，名称解析较慢。
作用域	受限制。硬链接不能跨越文件系统。	不受限制。符号链接可以跨越文件系统。

扩展属性

扩展属性（EA）是与对象相关联的信息，它提供了关于对象的附加详细信息。EA 由一个用来表示它的名称和一个值组成。值可以是文本、二进制数据或其它类型的数据。

只要对象存在，对象的 EA 便存在。

EA 具有很多变体，可以用来包含各种信息。特别是可能需要知道下列三个 EA:

.SUBJECT
对象内容或目的的简要描述。

.TYPE 对象中数据的类型。数据类型可以为文本、二进制、程序来源、编译后的程序或其它信息。

.CODEPAGE
要用于对象的代码页。用于对象的代码页也用于与该对象相关联的 EA。

句点（.）作为名称的第一个字符意味着该 EA 是标准的系统 EA（SEA），保留它供系统使用。

各种文件系统中的各种对象可能有也可能没有 EA。QSYS.LIB 和“独立 ASP QSYS.LIB”文件系统支持三种预定义的 EA: .SUBJECT、.TYPE 和 .CODEPAGE。在文档库服务（QDLS）文件系统中，文件夹和文档可以具有任何种类的 EA。某些文件夹和文档可能具有 EA，而某些则可能不具有。在“根”（/）、开放系统（QOpenSys）和用户定义文件系统中，所有目录、流文件和符号链接可以具有任何种类的 EA。但是，某些对象可能根本不具有任何 EA。

“使用对象链接”（WRKLNK）命令可以用来显示对象的 .SUBJECT 扩展属性（EA）。应用程序或用户没有其它的集成文件系统支持可用来访问和更改 EA。此规则的唯一例外是“显示 UDFS”（DSPUDFS）和“显示安装的文件系统信息”（DSPMFSINF）CL 命令，这些命令对用户显示扩展属性。

但是，可以通过分层文件系统（HFS）提供的接口来更改与 QDLS 中的某些对象相关联的 EA。有关这些文件系统的更多信息，参见第 64 页的『文档库服务文件系统（QDLS）』和第 66 页的『光盘文件系统（QOPT）』。

若通过 OS/2 或 Windows 将客户机 PC 连接到 iSeries 服务器，则相应操作系统的编程接口（如 DosQueryFileInfo 和 DosSetFileInfo）可用来查询和设置任何文件对象的 EA。OS/2 用户可以通过使用设置笔记本（即通过在与对象相关联的弹出菜单中选择设置）在桌面上更改对象的 EA。

若定义扩展属性，则使用下列命名准则:

- EA 的名称长度最多为 255 个字符。
- 不要使用句点（.）作为名称的第一个字符。其名称以句点开始的 EA 将解释为标准系统 EA。
- 要使名称冲突的可能性减至最小，对 EA 使用一致的命名结构。建议使用以下格式:

```
CompanyNameProductName.Attribute_Name
```

名称连续性

| 当使用“根”（/）、QOpenSys 和用户定义的文件系统时，可以利用系统支持来确保对象名称中的字符保持相
| 同。当跨具有不同字符编码模式（代码页）的 iSeries 服务器和已连接的设备使用这些文件系统时，这一点也适
| 用。服务器以一种 16 位格式存储名称中的字符，该格式对于 *TYPE1 目录称为 UCS2 Level 1（也称为
| **Unicode**），对于 *TYPE2 目录称为 UTF-16。有关目录格式的更多信息，参考 *TYPE2 目录。UCS2 Level
| 1 和 UTF-16 是 ISO 10646 标准的子集。使用该名称时，系统将字符的存储格式转换为正在使用的代码页中
| 的正确字符表示。与每个对象相关联的扩展属性的名称也按相同方式处理。

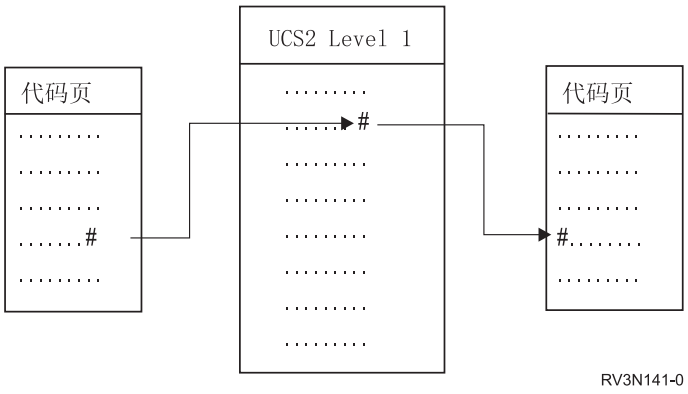


图 9. 保持字符与跨编码模式时相同

此支持便于从使用不同代码页的设备与服务器进行交互。例如，PC 用户可以使用相同文件名来访问 iSeries 文件，即使他们的 PC 没有与您的服务器相同的代码页。服务器自动处理代码页之间的转换。当然，设备使用的代码页必须包含名称中所使用的字符。

第 3 章 使用传统系统界面访问集成文件系统

用来使用系统的库、对象、数据库文件、文件夹和文档的所有用户界面（如菜单、命令和屏幕）仍然与在引入集成文件系统之前一样起作用。但是，这些界面不能用来使用集成文件系统支持的流文件、目录和其它对象。

为集成文件系统提供了一组独立的用户界面。可以对能通过集成文件系统访问的任何文件系统中的对象使用这些界面。

可以通过使用菜单和屏幕或通过使用控制语言（CL）命令从服务器与集成文件系统的目录和对象进行交互。另外，您还可以使用应用程序接口（API）来利用集成文件系统的流文件、目录和其它支持。

还可以通过“iSeries 导航器”（用于从 Windows 桌面管理和控制服务器的图形用户界面）与集成文件系统进行交互。

有以下几种与集成文件系统进行交互的方式:

使用 API
对集成文件系统目录和流文件执行操作的应用程序接口（API）具有 C 语言函数的格式。

使用 CL 命令
CL 命令可以对能通过集成文件系统访问的任何文件系统中的文件和其它对象进行操作。

使用 iSeries 菜单和屏幕
通过使用服务器提供的一组菜单和屏幕，可以对集成文件系统中的文件和其它对象执行操作。

使用“iSeries 导航器”
“iSeries 导航器”是用于从 Windows 桌面管理和控制服务器的图形用户界面。

使用 PC
若将您的 PC 连接到 iSeries 服务器，则可以与集成文件系统的目录和对象进行交互，就好象它们存储在您的 PC 上一样。

使用 iSeries 菜单和屏幕执行操作

通过使用服务器提供的一组菜单和屏幕，可以对集成文件系统中的文件和其它对象执行操作。要显示集成文件系统菜单:

1. 注册到您的服务器。
2. 按**执行**键继续。
3. 从 iSeries 主菜单选择**文件、库和文件夹**选项。
4. 从“文件、库和文件夹”菜单选择**集成文件系统**选项。

从此处，可以根据您的需要使用集成文件系统中的“目录”命令、“对象”命令或“安全性”命令。但是，如果您知道将使用的 CL 命令，可以在屏幕底部的命令行输入它，并按**执行**键，绕过选项菜单。

另外，可以通过执行下列步骤从您的服务器上的任何菜单访问集成文件系统:

1. 在任何命令行上输入 `GO DATA` 以显示“文件、库和文件夹”菜单。
2. 选择选项集成文件系统。

要查看“网络文件系统”命令的菜单，在任何命令行输入 `GO CMDNFS`。要查看用户定义的文件系统命令的菜单，在任何命令行输入 `GO CMDUDFS`。

从集成文件系统菜单，可以请求多个屏幕，可以在这些屏幕上执行下列操作:

- 创建、转换和除去目录
- 显示和更改当前目录的名称
- 添加、显示、更改和除去对象链接
- 复制、移动和重命名对象
- 检出和检入对象
- 保存（备份）和恢复对象
- 显示和更改对象所有者和用户权限
- 在流文件和数据库文件成员之间复制数据
- 创建、删除和显示用户定义文件系统的状态
- 从服务器中导出文件系统
- 在客户机上安装和卸装文件系统

某些文件系统并非支持所有这些操作。有关特定文件系统的限制，参见第 4 页的『集成文件系统中的文件系统』。

有关集成文件系统菜单和屏幕的更多信息，参考以下主题:

- CL 命令和屏幕的路径名规则

使用 CL 命令执行操作

可以通过集成文件系统菜单和屏幕（参见第 21 页的『使用 iSeries 菜单和屏幕执行操作』）执行的所有操作可以通过输入控制语言（CL）命令来执行。这些命令可以对能通过集成文件系统接口访问的任何文件系统中的文件和其它对象进行操作。

表 1 概述了集成文件系统命令。有关具体与用户定义的文件系统、“网络文件系统”和已安装的一般文件系统相关的 CL 命令的更多信息，参见第 56 页的『用户定义文件系统（UDFS）』和第 75 页的『网络文件系统（NFS）』。在命令与 OS/2 或 DOS 命令执行相同操作的地方，提供了别名（替代命令名），以方便 OS/2 和 DOS 用户。

表 2. 集成文件系统命令

命令	描述	别名
ADDLNK	添加链接。在目录和对象之间添加链接。	
ADDMFS	添加安装的文件系统。将导出的远程服务器文件系统安装在本地客户机目录。	MOUNT
APYJRNCHG[2]	应用记入日志的更改。使用日志项来应用自保存记入日志的对象以来已发生的更改或应用直到指定点的更改。	
CHGATR	更改属性。更改单个对象、一组对象或某个目录树（已更改其目录、目录的内容及其所有子目录的内容的属性）的属性。	
CHGAUD	更改审计值。打开或关闭对对象的审计。	
CHGAUT	更改权限。将对象的特定权限授予用户或用户组。	
CHGCURDIR	更改当前目录。更改要用作当前目录的目录。	CD 和 CHDIR
CHGNFSEXP	更改网络文件系统导出。将目录树添加至导出到 NFS 客户机的导出表或从该表除去目录树。	EXPORTFS
CHGOWN	更改所有者。将对象所有权从一个用户移交给另一用户。	

表 2. 集成文件系统命令 (续)

命令	描述	别名
CHGPGP	更改主组。将主组从一个用户更改为另一用户。	
CHKIN	检入。检入先前检出的对象。	
CHKOUT	检出。检出一个对象，以防止其它用户更改该对象。	
CPY	复制。复制单个对象或一组对象。	COPY
CPYFRMSTMF	从流文件中进行复制。将数据从流文件复制到数据库文件成员。	
CPYTOSTMF	复制到流文件。将数据从数据库文件成员复制到流文件。	
CRTDIR	创建目录。将新的目录添加到系统。	MD，MKDIR
CRTUDFS	创建 UDFS。创建用户定义文件系统。	
CVTDIR	转换目录。提供关于将集成文件系统目录从 *TYPE1 格式转换到 *TYPE2 格式的信息，或执行转换。	
CVTRPCSRC	转换 RPC 源文件。从“远程过程调用”（RPC）语言编写的输入文件中生成 C 代码。	RPCGEN
DLTUDFS	删除 UDFS。删除用户定义文件。	
DSPAUT	显示权限。显示对象的授权用户及他们对对象的权限的列表。	
DSPCURDIR	显示当前目录。显示当前目录的名称。	
DSPLNK	显示对象链接。显示目录中对象的列表，并提供一些选项来显示关于这些对象的信息。	
DSPF	显示流文件。显示流文件或数据库文件。	
DSPMFSINF	显示安装的文件系统信息。显示关于安装的文件系统的信息。	STATFS
DSPUDFS	显示 UDFS。显示用户定义文件系统。	
EDTF	编辑流文件。编辑流文件或数据库文件。	
ENDJRN[2]	结束日志。结束对象或对象列表的更改的日志记录。	
ENDNFSSVR	结束网络文件系统服务器。结束服务器和客户机上的一个或所有 NFS 守护程序。	
ENDRPCBIND	结束 RPC 绑定程序守护程序。结束“远程过程调用”（RPC）RPCBind 守护程序。	
MOV	移动。将对象移动到另一目录。	MOVE
RLSIFSLCK	释放集成文件系统锁定。释放客户机或对象上保持的所有 NFS 字节范围锁定。	
RMVDIR	除去目录。从系统中除去目录。	RD，RMDIR
RMVLNK	除去链接。除去到对象的链接。	DEL，ERASE
RMVMFS	除去安装的文件系统。将导出的远程服务器文件系统从本地客户机目录中除去。	UNMOUNT
RNM	重命名。更改目录中对象的名称。	REN
RPCBIND	启动 RPC 绑定程序守护程序。启动“远程过程调用”（RPC）RPCBind 守护程序。	
RST	恢复。将一个对象或一组对象从备份设备复制到系统。	
RTVCURDIR	检索当前目录。检索当前目录的名称并将它存放在指定的变量中（用于 CL 程序）。	
SAV	保存。将一个对象或一组对象从系统复制到备份设备。	
SNDJRNE[2]	发送日志项。将用户日志项（可选择与记入日志的对象相关联）添加到日志接收器中。	

表 2. 集成文件系统命令 (续)

命令	描述	别名
STRJRN[2]	启动日志。开始将更改（对一个对象或对象列表所做的更改）记入到特定的日志。	
STRNFSSVR	启动网络文件系统服务器。启动服务器和客户机上的一个或所有 NFS 守护程序。	
WRKAUT	使用权限。显示用户及其权限的列表，并提供用于添加用户、更改用户权限或除去用户的选项。	
WRKLNK	使用对象链接。显示目录中对象的列表，并提供用于对这些对象执行操作的选项。	
WRKOBJOWN[1]	按所有者使用对象。显示用户概要文件拥有的对象列表，并提供用于对这些对象执行操作的选项。	
WRKOBJPGP[1]	按主组使用对象。显示主组控制的对象列表，并提供用于对这些对象执行操作的选项。	

注:

1. WRKOBJOWN 和 WRKOBJPGP 命令可以显示所有对象类型，但可能并非在所有文件系统中都能完全运行。
2. 有关更多信息，参见 “iSeries 信息中心” 中的日志管理。

有关集成文件系统 CL 命令和对于在特定文件系统中使用这些命令的限制的更多信息，参考下列主题:

- 集成文件系统中的文件系统
- CL 命令和屏幕的路径名规则
- “iSeries 信息中心” 中的 CL 主题

CL 命令和屏幕的路径名规则

使用集成文件系统命令或屏幕对对象进行操作时，通过提供其路径名来标识该对象。以下是指定路径名时应牢记的规则摘要。这些规则中**对象**这个术语表示任何目录、文件、链接或其它对象。

- 对象名在每个目录中必须是唯一的。
- 传送到集成文件系统 CL 命令的路径名必须用对作业当前有效的 CCSID 来表示。若作业的 CCSID 为 65535，则路径名必须用作业的缺省 CCSID 来表示。因为文本字符串通常用 CCSID 37 来编码，所以有必要在将路径传送到命令之前，将硬编码的路径名转换为作业 CCSID。
- 在命令行中输入路径名时，必须将它用撇号（'）标记括起来。在屏幕中输入路径名时，这些标记是可选的。但是，若路径名包括任何加引号的字符串，则还是必须包括封闭的 ' ' 标记。
- 从左到右输入路径名，从最高级目录开始，以命令要进行操作的对象的名称结束。路径中每个部分的名称用斜杠（/）或反斜杠（\）隔开；例如:

  ```
  'Dir1/Dir2/Dir3/UsrFile'
  ```

  或

  ```
  'Dir1\Dir2\Dir3\UsrFile'
  ```

- 不能在路径名的各个部分中使用 / 和 \ 字符以及空字符（因为 / 和 \ 用作分隔符）。 命令不会将小写字母更改为大写字母。名称可能更改也可能不更改为大写，这取决于包含对象的文件系统是否是区分大小写的，以及是否正在创建或搜索该对象。

- 对象名的长度受对象所在文件系统和命令串的最大长度限制。命令将接受多达 255 个字符长的对象名和多达 5000 个字符长的路径名。

  有关每个文件系统中路径名的限制，参见集成文件系统中的文件系统。
- 路径名开始处的 / 或 \ 字符表示该路径从最高目录即“根”（/）目录开始；例如:

  ```
  '/Dir1/Dir2/Dir3/UsrFile'
  ```
- 若路径名不以 / 或 \ 字符开始，则假定路径从用户输入命令的当前目录开始；例如:

  ```
  'MyDir/MyFile'
  ```

  其中 MyDir 是用户当前目录的子目录。
- 在路径名开始处后跟斜杠（或反斜杠）的代字号（[]）字符表示路径从用户输入命令的主目录开始；例如:

  ```
  '~/UsrDir/UsrObj'
  ```
- 在路径名开始处后跟用户名再跟斜杠（或反斜杠）的代字号（[]）字符表示路径从用户名标识的用户的主目录开始；例如:

  ```
  '~user-name/UsrDir/UsrObj'
  ```
- 在某些命令中，可以在路径名的最后一个部分中使用星号（*）或问号（?）来搜索名称模式。* 告诉系统搜索在 * 位置中有任何数目的字符的名称。? 告诉系统搜索在 ? 位置中有单个字符的名称。下列示例搜索其名称以 *d* 开始并以 *txt* 结束的所有对象:

  ```
  '/Dir1/Dir2/Dir3/d*txt'
  ```

  下列示例搜索其名称以后跟任何单个字符的 *d* 开始并以 *txt* 结束的对象:

  ```
  '/Dir1/Dir2/Dir3/d?txt'
  ```
- 要避免与 iSeries 服务器特殊值产生混淆，路径名不能以单个星号（*）字符开始。要在路径名的开始处执行模式匹配，使用两个星号（*）；例如:

  ```
  '**.file'
  ```

  注: 这只适合于其中星号（*）前没有其它字符的相对路径名。
- 当对 QSYS.LIB 文件系统中的对象执行操作时，组成名称的格式必须为 *name.object-type*；例如:

  ```
  '/QSYS.LIB/PAY.LIB/TAX.FILE'
  ```

  有关详细信息，参见第 60 页的『库文件系统（QSYS.LIB）』。
- 当对“独立 ASP QSYS.LIB”文件系统中的对象执行操作时，部分名的格式必须为 *name.object-type*；例如:

  ```
  '/asp_name/QSYS.LIB/PAYDAVE.LIB/PAY.FILE
  ```

  有关详细信息，参见第 62 页的『独立 ASP QSYS.LIB』。
- 若部分名中使用了下列任何字符，则必须将路径名用成对的附加撇号（'）或引号（"）括起来:
  - 星号（*）
  - 问号（?）
  - 撇号（'）
  - 引号（"）
  - 代字号（[]），若用作路径名的第一个部分名中的第一个字符（若在任何其它位置使用，则将代字号解释为另一个字符）

  例如:

  ```
  ''/Dir1/Dir/A*Smith''
  ```

或

```
'''/Dir1/Dir/A*Smith'''
```

建议不要使用此习惯，因为命令串中的字符意义可能产生混淆，并很有可能不会正确输入命令串。

- 不要在路径名中使用冒号（:）。冒号在系统中有特殊意义。
- 命令和相关联的用户屏幕的处理支持不会将十六进制数 40 以下的代码点识别为可以在命令串或屏幕中使用的字符。若使用这些代码点，则必须以十六进制表示的形式输入它们，例如:

  ```
  crtdir dir(X'02')
  ```

  因此，建议不要在路径名中使用十六进制数 40 以下的代码点。此限制只适用于命令和相关联的屏幕，不适用于 API（参见第 40 页的『使用 API 执行操作』）。

有关使用特定命令的限制，参见命令帮助或“iSeries 信息中心”中的控制语言（CL）主题。

使用 PC 执行操作

若将您的 PC 连接到 iSeries 服务器，则可以与集成文件系统的目录和对象进行交互，就好象它们存储在您的 PC 上一样。可以通过使用 Windows“资源管理器”的拖放能力在目录之间复制对象。需要时，可以通过选择服务器驱动器中的对象并将该对象拖动到 PC 驱动器来将对象从服务器实际上复制到 PC。

通过使用 Windows 界面在 iSeries 服务器和 PC 之间复制的任何对象可以自动在 EBCDIC 和 ASCII 之间转换。EBCDIC 是扩充的二－十进制交换码，而 ASCII 则是美国信息交换标准码。可以配置 iSeries Access 自动执行这种转换，甚至可以指定对具有特定扩展名的文件执行转换。从 OS/400 V4R4 开始，也可以配置 iSeries NetServer 来对文件执行转换。

根据对象类型的不同，可以使用 PC 接口和可能的 PC 应用程序来使用对象。例如，可以使用 PC 编辑器来编辑包含文本的流文件。

若使用 PC 连接到 iSeries 服务器，则集成文件系统使服务器的目录和对象可用于 PC。PC 可以通过使用构建到 Windows 操作系统内部的文件共享客户机、FTP 客户机或“iSeries 导航器”（iSeries Access 的一部分）来使用集成文件系统中的文件。PC 使用 Windows 文件共享客户机来访问在 iSeries 服务器上运行的 iSeries NetServer。

使用 FTP 传送文件
FTP 客户机允许您传送 iSeries 服务器上找到的文件，包括在“根”（/）、QSYS.LIB、“独立 ASP QSYS.LIB”、QOpenSys、QOPT 和 QFileSvr.400 文件系统中找到的那些文件。它还允许您传送文档库服务（QDLS）文件系统中的文件夹和文档。

使用“iSeries 导航器”来使用文件
iSeries Access 包括“iSeries 导航器”，它连接至 iSeries 服务器并使集成文件系统可用于 PC。“iSeries 导航器”是用于从 Windows 桌面管理和控制 iSeries 服务器的图形用户界面。

使用 iSeries NetServer 来使用文件
iSeries NetServer 是 OS/400 的一个部件，它允许构建到 Windows 客户机内部的文件和打印共享来使用服务器。

注: iSeries Access 的新版本完全依靠 NetServer 来访问集成文件系统。NetServer 支持只可用于与运行 OS/400 V4R2 和更高版本的 iSeries 服务器的 TCP/IP 连接。

使用 FTP 传送文件

“文件传输协议”（FTP）客户机允许您传送 iSeries 服务器上找到的文件，包括“根”、QOpenSys、QSYS.LIB、“独立 ASP QSYS.LIB”、QOPT 和 QFileSvr.400 文件系统中找到的那些文件。它还

| 允许您传送文档库服务（QDLS）文件系统中的文件夹和文档。可以以无人照管批处理方式交互式运行 FTP 客户机，其中，从文件读取客户机子命令并将对这些子命令的响应写入文件。它还包括用于管理服务器上文件的其它功能部件。

| 可以使用 FTP 支持将文件传送到下列任何文件系统或执行相反操作:

- “根”（/）文件系统
- 开放系统文件系统（QOpenSys）
- 库文件系统（QSYS.LIB）
- “独立 ASP QSYS.LIB”文件系统
- 文档库服务文件系统（QDLS）
- 光盘文件系统（QOPT）
- 网络文件系统（NFS）
- NetWare 文件系统（QNetWare）
- Windows NT 服务器文件系统（QNTC）

| 但是，应知道下列限制:

- 集成文件系统将 FTP 支持限于只传送文件数据。不能使用 FTP 来传送属性数据。
- QSYS.LIB 和“独立 ASP QSYS.LIB”文件系统将 FTP 支持限制于物理文件成员、源物理文件成员和保存文件。不能使用 FTP 来传送其它对象类型，如程序（*PGM）。但是，可以将其它对象类型保存到某个保存文件，传送该保存文件，然后恢复这些对象。

| 有关 FTP 的信息，参见“iSeries 信息中心”的**联网**类别中的下列主题:

- FTP
- 使用 FTP 传送文件

使用 iSeries NetServer 来使用文件

| Windows“网上邻居”的 iSeries 支持（iSeries NetServer）是 IBM Operating System/400 版本 5（OS/400）功能，它使 Windows 客户机能够访问 OS/400 共享目录路径和共享输出队列。iSeries NetServer 允许运行 Windows 软件的 PC 无缝访问 iSeries 管理的数据和打印机。网络上的 PC 客户机只需使用包括在它们的操作系统中的文件和打印共享功能。这意味着您不需要在 PC 上安装任何附加软件也能使用 iSeries NetServer。

| 安装了 Samba 客户机软件的 LINUX 客户机也可以通过 iSeries NetServer 无缝访问数据和打印机。可以用与从 iSeries 安装 NFS 文件系统类似的方法来从 iSeries NetServer 安装 Samba 文件系统（smbfs）。有关更多信息，参见“iSeries 信息中心”中的 iSeries NetServer 主题。

| iSeries NetServer 文件共享是 iSeries NetServer 与 iSeries 网络中的客户机共享的目录路径。文件共享可以由 iSeries 上的任何集成文件系统目录组成。必须先创建 iSeries NetServer 文件共享，并且，如果有必要，使用“iSeries 导航器”更改 iSeries NetServer 文件共享，然后才能使用 iSeries NetServer 来使用文件共享。

| 要使用 iSeries NetServer 访问集成文件系统文件共享:

1. 右键单击**开始**，并选择**资源管理器**来打开 Windows PC 上的 Windows“资源管理器”。
2. 打开**工具**菜单，并选择**映射网络驱动器**。
3. 为文件共享选择一个空闲的驱动器盘符（如 I:\ 驱动器）。
4. 输入 iSeries NetServer 文件共享的名称。例如，可以输入以下语法: **\\QSYSTEM1\Sharename**

   注: QSYSTEM1 是 iSeries NetServer 的系统名称，而 Sharename 是要使用的文件共享的名称。

| 5. 单击“确定”。

| **注**: 当使用 iSeries NetServer 连接时，服务器名称可以与 iSeries Access 所使用的名称不同。例如，iSeries NetServer 名称可以是 QAS400X，而使用文件的路径可以是 \\QAS400X\QDLS\MYFOLDER.FLR\MYFILE.DOC。但是，iSeries Access 名称可以是 AS400X，而使用文件的路径可以是 \\AS400X\QDLS\MYFOLDER.FLR\MYFILE.DOC。

| 选择将哪些目录与使用 iSeries NetServer 的网络共享。那些目录表现为服务器名称之下的第一级。例如，如果用名称 fredsdir共享 /home/fred 目录，则用户将能够从具有名称 \\QAS400X\FREDSDIR 的 PC 或从具有名称 //qas400x/fredsdir 的 LINUX 客户机访问该目录。

| “根”（/）文件系统为 PC 文件服务提供了比其它 iSeries 文件系统更好的性能。您可能要将文件移动到“根”（/）文件系统。有关更多信息，参见将对象移动到另一个文件系统时的注意事项。

| 有关 iSeries NetServer 和文件共享的更多信息，参见“iSeries 信息中心”的**联网**类别中的下列主题:

| • iSeries NetServer
| • iSeries NetServer 文件共享
| • 使用 Windows PC 客户机访问 iSeries NetServer 文件共享

将对象移动到另一个文件系统

在使用集成文件系统在文件系统之间移动对象之前，检查第 29 页的『将对象移动到另一个文件系统时的注意事项』。

要将对象移动到另一个文件系统，执行下列步骤:

1. 保存您计划移动的所有对象的副本。

   若发现应用程序不能访问对象正在迁移至的文件系统中的对象，则具有备份允许您将对象恢复到原来的文件系统。

   注: 不能从一个文件系统保存对象而将它们恢复至另一文件系统。

2. 使用创建目录（CRTDIR）命令在您想要将对象移动至的文件系统中创建目录。

   应仔细检查对象当前所在目录的属性，以确定是否要在创建的目录中复制这些属性。例如，目录的所有者是创建该目录的用户，而不是拥有旧目录的用户。若文件系统支持设置目录的所有者，则可能要在创建目录之后移交目录的所有权。

3. 使用移动（MOV）命令将文件移动到已选择的文件系统。

   之所以建议使用 MOV，是因为若文件系统支持设置对象的所有权，则该命令可以保持对象的所有权。但是，可以使用复制（CPY）命令来通过使用 OWNER(*KEEP) 参数保存对象的所有权。记住，这仅对支持设置对象所有者的文件系统有效。使用 MOV 或 CPY 时注意下列事项:

   - 属性可能不匹配，并可能被废弃。
   - 可能废弃扩展属性。
   - 权限可能不等价，并可能被废弃。

   这意味着，若决定将对象返回到其原来的文件系统，由于已经废弃的属性和权限，可能不想只移动或复制回该对象。返回对象的安全方法是恢复它的保存版本。

将对象移动到另一个文件系统时的注意事项

每个文件系统有其自己的独有特征。但是，将对象移动到另一个文件系统可能意味着失去对象当前所在文件系统的优点。您可能要将对象从一个文件系统移动到另一个文件系统以利用这些特征。在将对象移动到另一个文件系统之前，应熟悉集成文件系统上的文件系统及其特征。有关更多信息，参见第 4 页的『集成文件系统中的文件系统』。

还应该考虑下列事项:

- 是否正在使用利用了对象当前所在文件系统的优点的应用程序？

  某些文件系统支持不是集成文件系统支持的一部分的接口。使用这些接口的应用程序可能再也无法访问迁移到另一文件系统的对象。例如，QDLS 和 QOPT 文件系统支持分层文件系统（HFS）。API 和命令使用文档和文件夹对象。不能对其它文件系统中的对象使用这些接口。

- 哪些对象特性比较重要？

  所有文件系统并非都支持所有特性。例如，QSYS.LIB 或“独立 ASP QSYS.LIB”文件系统仅支持存储和检索少量的扩展属性，而“根”（/）和 QOpenSys 文件系统支持存储和检索所有扩展属性。因此，QSYS.LIB 和“独立 ASP QSYS.LIB”不适合于存储具有扩展属性的对象。QDLS 支持很多“office”属性，但其它文件系统不支持。因此，QDLS 适合于保存 office 文档。

  存储在 QDLS 中的 PC 文件适合于移动。大多数 PC 应用程序应该能继续使用从 QDLS 移动到其它文件系统的 PC 文件。“根”（/）、QOpenSys、QNetWare 和 QNTC 文件系统是存储这些 PC 文件的好的选择。因为它们支持很多 OS/2 文件系统特性，所以这些文件系统可以提供对文件的快速访问。

集成文件系统提供的目录

若下列目录尚不存在，集成文件系统会在系统重新启动时创建它们:

/tmp /tmp 目录给应用程序提供存储临时文件的位置。此目录是“根”（/）目录的子目录，所以它的路径名为 /tmp。

一旦应用程序将文件存放在 /tmp 目录，该文件就保持在那个位置，直到您或应用程序除去它为止。系统不会自动从 /tmp 中除去文件，或对 /tmp 中的文件执行任何其它特殊处理。

可以使用支持集成文件系统的用户屏幕和命令来管理 /tmp 目录及其文件。例如，可以使用“使用对象链接”屏幕或 WRKLNK 命令来复制、除去或重命名 /tmp 目录或该目录中的文件。系统授予所有用户对该目录的 *ALL 权限，这意味着他们可以对该目录执行大多数有效操作。

应用程序可以使用支持集成文件系统的应用程序接口（API）来管理 /tmp 及其文件（参见第 40 页的『使用 API 执行操作』）。例如，应用程序可以使用 unlink() API 来除去 /tmp 中的文件。

若除去 /tmp，下次系统重新启动时会自动再次创建它。

/home 系统管理员使用 /home 目录来存储每个用户的独立目录。系统管理员经常在 /home 中设置与用户概要文件相关联的主目录作为用户目录，例如 /home/john。有关更多信息，参见第 8 页的『当前目录和主目录』。

/etc /etc 存储管理文件、配置文件和其它系统文件。

/usr /usr 目录包括一些子目录，这些子目录包含系统使用的信息。/usr 中的文件一般不会经常更改。

/usr/bin

/usr/bin 目录包含标准的实用程序。

/QIBM /QIBM 目录是系统目录并与系统一起提供。

/QIBM/ProdData

/QIBM/ProdData 目录是用于“许可程序”产品数据的系统目录。

/QIBM/UserData

/QIBM/UserData 目录是用于“许可程序”用户数据（如配置文件）的系统目录。

/QOpenSys/QIBM

/QOpenSys/QIBM 目录是 QOpenSys 文件系统的系统目录。

/QOpenSys/QIBM/ProdData

/QOpenSys/QIBM/ProdData 目录是 QOpenSys 文件系统的系统目录，用于“许可程序”产品数据。

/QOpenSys/QIBM/UserData

/QOpenSys/QIBM/UserData 目录是 QOpenSys 文件系统的系统目录，用于“许可程序”用户数据，如配置文件。

/asp_name/QIBM

/asp_name/QIBM 目录是存在于系统上的任何独立 ASP 的系统目录，其中 asp_name 是独立 ASP 的名称。

/asp_name/QIBM/UserData

/asp_name/QIBM/UserData 目录是一个系统目录，用于“许可程序”用户数据，如存在于系统上的任何独立 ASP 的配置文件，其中 asp_name 是独立 ASP 的名称。

第 4 章 使用“iSeries 导航器”访问集成文件系统

“iSeries 导航器”是用于从 Windows 桌面管理和控制系统的图形用户界面。“iSeries 导航器”使操作和管理系统更容易和更高效。例如，可以通过将用户概要文件从一个 iSeries 服务器拖动到另一个 iSeries 服务器来将用户概要文件复制到另一个系统上。向导指导您完成设置安全性和 TCP/IP 服务以及应用程序。

有许多任务可以使用“iSeries 导航器”执行。以下列示的是帮助您入门的某些公共文件系统任务:

使用文件和文件夹

- 第 34 页的『创建文件夹』
- 第 34 页的『除去文件夹』
- 『检入文件』
- 第 32 页的『检出文件』
- 第 32 页的『设置对文件或文件夹的许可权』
- 第 32 页的『设置文件文本转换』
- 第 33 页的『将文件或文件夹发送至另一个系统』
- 第 33 页的『更改软件包定义的选项』
- 第 33 页的『调度发送文件或文件夹的日期和时间』

使用文件共享

- 第 34 页的『创建文件共享』
- 第 34 页的『更改文件共享』

使用用户定义的文件系统

- 第 35 页的『创建新的用户定义的文件系统』
- 第 35 页的『安装用户定义的文件系统』
- 第 36 页的『卸装用户定义的文件系统』

将对象记入日志

- 第 36 页的『启动日志记录』
- 第 36 页的『结束日志记录』

检入文件

要检入文件:

1. 在 **iSeries 导航器**中，右键单击要检入的文件。
2. 选择**特性**。
3. 选择**文件特性 –> 使用页面**。
4. 单击**检入**。

检出文件

要检出文件:

1. 在 **iSeries 导航器**中，右键单击要检出的文件。
2. 选择**特性**。
3. 选择**文件特性 –> 使用页面**。
4. 单击**检出**。

设置对文件或文件夹的许可权

添加对对象的许可权允许您控制其它用户操纵该对象的能力。使用许可权，您可以允许某些用户只查看对象，而允许其它用户实际编辑对象。

要设置对文件或文件夹的许可权:

1. 在 **iSeries 导航器**窗口中展开要使用的系统。
2. 展开**文件系统**。
3. 展开**集成文件系统**。继续展开，直到可看到要对其添加许可权的对象为止。
4. 右键单击要对其添加许可权的对象，并选择**许可权**。
5. 单击**许可权**对话框中的**添加**。
6. 选择一个或多个用户和组，或在**添加**对话框中的用户或组名称字段中输入用户或组的名称。
7. 单击**确定**。这将会将用户或组添加至列表的顶部。
8. 单击**详细信息**按钮来实现详细的许可权。
9. 通过选择适当的复选框旁边的框来对用户应用想要的许可权。
10. 单击**确定**。

设置文件文本转换

| 可以在“iSeries 导航器”中设置自动文本文件转换。自动文本文件转换允许您使用文件扩展名进行文件数据转
| 换。当在 iSeries 和 PC 之间转换数据文件时，集成文件系统可以转换数据文件。当从 PC 访问数据文件时，
| 可以将该文件当做是 ASCII 格式一样处理它。

要设置文件文本转换:

1. 在 **iSeries 导航器**中展开要使用的系统。
2. 展开**文件系统**。
3. 右键单击**集成文件系统**并选择**特性**。
4. 在**自动文本文件转换的文件扩展名**文本框中输入要自动转换的文件扩展名，并单击**添加**。
5. 对所有要自动转换的文件扩展名重复步骤 4。
6. 单击**确定**。

将文件或文件夹发送至另一个系统

要将文件或文件夹发送至另一个系统:

1. 在 **iSeries 导航器**中展开要使用的系统。
2. 展开**文件系统**。
3. 展开**集成文件系统**。继续展开，直到可看到要发送的文件或文件夹为止。
4. 右键单击文件或文件夹并选择**发送**。文件或文件夹出现在**从中发送文件**对话框的“已选择的文件和文件夹”列表中。
5. 展开可用系统和组的列表。
6. 选择一个系统并单击**添加**来将该系统添加至**目标系统和组**列表中。对要发送此文件或文件夹的所有系统重复此步骤。
7. 单击**确定**来用当前缺省软件包定义和调度信息发送文件或文件夹。

还可以『更改软件包定义的选项』或『调度发送文件或文件夹的日期和时间』。

当创建软件包定义时，可以保存它并在任何时间重新使用它来将已定义的一组文件和文件发送至多个端点系统或系统组。如果选择创建文件的快照，则可以保存同一组文件的副本的多个版本。发送快照可以确保在分发期间不会更新文件，以便最后一个目标系统与第一个目标系统接收相同的对象。

更改软件包定义的选项

| 软件包定义使您能够将一组 OS/400 对象或集成文件系统文件组合在一起。通过获取文件的快照来保存它们供
| 以后分发，软件包定义还允许您将此同一组文件看作一个逻辑组或看作一个物理组。

要更改软件包定义的选项:

1. 完成『将文件或文件夹发送至另一个系统』的步骤。
2. 单击**选项**选项卡。缺省选项是在封装和发送文件时包括子文件夹并用将要发送的文件替换现有文件。
3. 按要求更改这些选项。
4. 单击**高级**设置高级保存和恢复选项。
5. 单击**确定**保存高级选项。
6. 单击**确定**发送文件，或单击**调度**设置用于发送文件的时间。

相关主题:

- 『调度发送文件或文件夹的日期和时间』。

调度发送文件或文件夹的日期和时间

使用调度程序功能使您可以灵活地在方便执行工作时执行工作。要调度发送文件或文件夹的日期和时间:

1. 完成『将文件或文件夹发送至另一个系统』的步骤。
2. 单击**调度**。
3. 选择何时要发送文件或文件夹的选项。

创建文件夹

要创建文件夹:

1. 在 **iSeries 导航器**中展开要使用的系统。
2. 展开**文件系统**。
3. 展开**集成文件系统**。
4. 右键单击要将新文件夹添加至的文件系统并选择**新文件夹**。
5. 在**新文件夹**对话框中输入对象的新名称。
6. 单击**确定**。

当在 iSeries 服务器上创建文件夹时，需要考虑是否要使用日志管理来保护新文件夹（或对象）。有关更多信息，参见日志管理。

相关主题:

- 启动日志记录
- 结束日志记录

除去文件夹

要除去文件夹:

1. 在 **iSeries 导航器**中展开要使用的系统。
2. 展开**文件系统**。
3. 展开**集成文件系统**。继续展开，直到可看到要除去的文件或文件夹为止。
4. 右键单击文件或文件夹并选择**删除**。

创建文件共享

文件共享是 iSeries NetServer 与 iSeries 网络中的 PC 客户机共享的目录路径。文件共享可以由 iSeries 上的任何集成文件系统目录组成。

要创建文件共享:

1. 在 **iSeries 导航器**中展开您的系统。
2. 展开**文件系统**。
3. 展开**集成文件系统**。
4. 展开包含要为其创建共享的文件夹的文件系统。
5. 右键单击要为其创建共享的文件夹并选择**共享**。
6. 选择**新建共享**。

更改文件共享

文件共享是 iSeries NetServer 与 iSeries 网络中的 PC 客户机共享的目录路径。文件共享可以由 iSeries 上的任何集成文件系统目录组成。

要更改文件共享:

1. 在 **iSeries 导航器**中展开您的系统。

2. 展开**文件系统**。
3. 展开**集成文件系统**。
4. 展开您要更改的已为其定义共享的文件夹。
5. 右键单击您要**共享**的已为其定义共享的文件夹。
6. 选择**新建共享**。

创建新的用户定义的文件系统

| 用户定义的文件系统（UDFS）是您创建并定义其属性的文件系统。UDFS 驻留在系统上的辅助存储池（ASP）
| 中。

要创建新的用户定义的文件系统（UDFS）：

1. 在 **iSeries 导航器**中展开您的系统。
2. 展开**文件系统**。
3. 展开**集成文件系统**。
4. 展开**根**。
5. 展开 **Dev**。
6. 单击要包含新的 UDFS 的辅助存储池（ASP）。
7. 从**文件**菜单选择**新建 UDFS**。
8. 在**新的用户定义的文件系统**对话框中指定 UDFS 名称、描述（可选）、审计值、缺省文件格式以及新 UDFS 中的文件是否将具有区分大小写的文件名。

安装用户定义的文件系统

| 用户定义的文件系统（UDFS）是您创建并定义其属性的文件系统。UDFS 驻留在系统上的辅助存储池（ASP）
| 中。要访问或查看 UDFS 中的数据存储，必须安装 UDFS。

| 当安装 UDFS 时，它覆盖在文件夹层次结构中的安装点下存在的任何文件系统、目录或对象。这使那些文件系
| 统、目录或对象在卸装 UDFS 之前不可访问。要确保维护对集成文件系统中所有数据的访问，在空文件夹上安
| 装 UDFS。安装 UDFS 之后，将可从该文件夹中访问 UDFS 中的文件。文件夹中所作的任何更改将是对 UDFS
| 的更改，而不是对已覆盖的文件夹的更改。

| **注:** 不能安装独立 ASP 上的 UDFS。

要安装用户定义的文件系统（UDFS）：

1. 在 **iSeries 导航器**中展开您的系统。
2. 展开**文件系统**。
3. 展开**集成文件系统**。
4. 展开**根**。
5. 展开 **Dev**。
6. 单击包含要安装的 UDFS 的辅助存储池（ASP）。
7. 在“操作导航器”的右窗格的 **UDFS 名称**列中右键单击要安装的 UDFS。
8. 选择**安装**。

如果您喜欢拖放，则可以通过将 UDFS 拖动到同一服务器中集成文件系统内的文件夹中来安装它。不能在 /dev、/dev/QASPxx、/dev/asp_name、另一个系统或桌面上放下 UDFS。

卸装用户定义的文件系统

当安装 UDFS 时，它会覆盖在文件夹层次结构中的安装点下存在的任何文件系统、目录或对象。这使那些文件系统、目录或对象在卸装 UDFS 之前不可访问。

要卸装用户定义的文件系统（UDFS）：

1. 在**操作导航器**中展开您的系统。
2. 展开**文件系统**。
3. 展开**集成文件系统**。
4. 展开**根**。
5. 展开 **Dev**。
6. 单击包含要卸装的 UDFS 的辅助存储池（ASP）。
7. 在“iSeries 导航器”的右窗格的 **UDFS 名称**列中右键单击要卸装的 UDFS。
8. 选择**卸装**。

启动日志记录

日志记录的主要目的是使您能够恢复对象自上次保存以来发生的更改。

要启动对对象的日志记录:

1. 在 **iSeries 导航器**中展开您的系统。
2. 展开**文件系统**。
3. 右键单击要记入日志的对象，并选择**日志记录...**。
4. 选择适当的日志记录选项之后，单击**启动**。

有关集成文件系统对象日志记录的更详细的信息，参考“iSeries 信息中心”中的日志管理。

结束日志记录

日志记录的主要目的是使您能够恢复对象自上次保存以来发生的更改。有关如何启动对象日志记录的更多信息，参见启动日志记录。一旦日志记录已对对象启动，则不论因为什么原因，您可能要结束对此对象的日志记录。

要结束对对象的日志记录:

1. 在 **iSeries 导航器**中展开您的系统。
2. 展开**文件系统**。
3. 右键单击要停止日志记录的对象，并选择**日志记录...**。
4. 单击**结束**。

有关集成文件系统对象日志记录的更详细的信息，参考“iSeries 信息中心”中的日志管理。

第 5 章 集成文件系统的编程支持

将集成文件系统添加到 iSeries 服务器不会影响现有 iSeries 服务器应用程序。诸如数据描述规范的编程语言、实用程序和系统支持与它们在添加集成文件系统之前一样起作用。

但是，要利用集成文件系统的流文件、目录和其它支持，必须使用一组为访问集成文件系统函数而提供的 C 语言应用程序接口（API）。

另外，添加集成文件系统允许您在物理数据库文件和流文件之间复制数据。可以使用 CL 命令、iSeries Access 的数据传送功能或 API 执行此复制。

下列主题解释如何将复制功能用于集成文件系统流文件，并介绍 API 作为访问集成文件系统功能的一种方式:

- 在流文件和数据库文件之间复制数据
- 在流文件和保存文件之间复制数据
- 使用 API 执行操作
- 套接字支持
- 命名和国际支持
- 数据转换

在流文件和数据库文件之间复制数据

若熟悉使用诸如数据描述规范（DDS）的面向记录设施对数据库文件进行操作，则可以发现对流文件执行操作的方式的一些基本差异。这些差异是由于流文件与数据库文件相比在结构上（或也许没有结构）不同而导致的。要访问流文件中的数据，应指示一个字节偏移量和一个长度。要访问数据库文件中的数据，一般应定义要使用的字段和要处理的记录数目。

因为预先定义了面向记录文件的格式和特性，所以操作系统知道该文件并可以帮助您避免执行不适合于该文件格式和特性的操作。使用流文件时，操作系统知道很少或根本不知道文件的格式。应用程序必须知道文件的外观和如何正确对该文件进行操作。而流文件允许极灵活的编程环境，但代价是不能从操作系统获得多少帮助或根本没有帮助。流文件较适合于某些编程情况；而面向记录的文件则较适合于其它编程情况。

在集成文件系统中，在流文件和数据库文件之间复制数据有几种方式:

- 使用 CL 命令复制数据
- 使用 API 复制数据
- 使用数据传送功能复制数据

使用 CL 命令复制数据

有两组 CL 命令允许您在流文件和数据库文件成员之间复制数据:

- CPYTOSTMF 和 CPYFRMSTMF
- CPYTOIMPF 和 CPYFRMIMPF

CPYTOSTMF 和 CPYFRMSTMF 命令

可以使用“从流文件复制”（CPYFRMSTMF）和“复制到流文件”（CPYTOSTMF）命令在流文件和数据库文件成员之间复制数据。可以通过使用 CPYTOSTMF 命令从数据库文件成员创建流文件。也可以通过使用 CPYFRMSTMF 命令从流文件创建数据库文件成员。若是复制目标的文件或成员不存在，则会创建它。

但是，有一些限制。数据库文件必须是只包含一个字段的程序描述的物理文件，或只包含一个文本字段的源物理文件。这些命令给您提供了各种选项来转换和重新格式化正在复制的数据。

CPYTOSTMF 和 CPYFRMSTMF 命令也可以用来在流文件和保存文件之间复制数据。

CPYTOIMPF 和 CPYFRMIMPF 命令

也可以使用“复制到导入文件”（CPYTOIMPF）和“从导入文件复制”（CPYFRMIMPF）命令在流文件和数据库成员之间复制数据。CPYTOSTMF 和 CPYFRMSTMF 命令不允许从外部描述的（DDS 描述的）复杂数据库文件移动数据。*导入文件*这个词表示流类型文件；该术语一般表示为了在异构数据库之间复制数据而创建的文件。

从流（或导入）文件复制时，CPYFRMIMPF 命令允许指定字段定义文件（FDF），该文件描述流文件中的数据。或者，可以指定流文件是定界的，并可以指定用来标记字符串、字段和记录边界的字符。还提供了用于转换诸如时间和日期的特殊数据类型的选项。

若目标流文件或数据库成员已存在，则在这些命令中提供数据转换。若文件不存在，则可以使用下列两步骤方法来转换数据:

1. 使用 CPYTOIMPF 和 CPYFRMIMPF 命令在外部描述的文件和源物理文件之间复制数据。
2. 使用 CPYTOSTMF 和 CPYFRMSTMF 命令（不管目标文件是否存在，这两个命令都提供完全的数据转换）在源物理文件和流文件之间进行复制。

以下是一个示例:

```
CPYTOIMPF  FROMFILE(DB2FILE) TOFILE(EXPFILE) DTAFMT(*DLM)
        FLDDLM(';') RCDDLM(X'07') STRDLM('"') DATFMT(*USA) TIMFMT(*USA)
```

DTAFMT 参数指定输入流（导入）文件是定界的；另一选项是 DTAFMT(*FIXED)，它要求指定字段定义文件。FLDDLM、RCDDLM 和 STRDLM 参数标识用作定界符或字段、记录和字符串的分隔符的字符。

DATFMT 和 TIMFMT 参数指示复制到导入文件的任何日期和时间信息的格式。

这些命令是有用的，因为可以将它们放置到程序中，并且它们完全在服务器上运行。但是，这些接口是复杂的。

有关更多信息，参见命令帮助或“iSeries 信息中心”中的命令语言（CL）主题。

使用 API 复制数据

若要在应用程序中将数据库文件成员复制到流文件，可以使用集成文件系统 open()、read() 和 write() 函数来打开成员，从成员中读取数据，以及将数据写入到成员。有关更多信息，参见 “iSeries 信息中心” 中的集成文件系统 API 主题。

使用数据传输功能复制数据

iSeries Access 数据传输应用程序具有易于使用的图形界面以及自动数字和字符数据转换的优点。但是，数据传输要求安装 iSeries Access 产品并要求使用 PC 和 iSeries 服务器资源并在它们之间进行通信。

若在 PC 和服务器上安装了 iSeries Access，则可以使用数据传输应用程序在流文件和数据库文件之间传送数据。可以将数据传送到基于现有数据库文件的新数据库文件，传送到外部描述的数据库文件，或传送到新的数据库文件定义和文件。

| 下列任务使用数据传输应用程序帮助您复制和传送数据:

- 将数据从数据库文件传送到流文件
- 将数据从流文件传送到数据库文件
- 将数据传送到新创建的数据库文件定义和文件
- 创建格式描述文件

将数据从数据库文件传送到流文件

要将文件从数据库文件传送到服务器上的流文件:

1. 建立与服务器的连接。
2. 将网络驱动器映射到 iSeries 文件系统中的适当路径。
3. 从 iSeries Access Windows 版窗口选择**从 iSeries 服务器传送数据**。
4. 选择要从中进行传送的服务器。
5. 使用 iSeries 数据库库和文件名选择要从中进行复制的文件名以及用于放置产生的流文件的网络驱动器。也可以选择 **PC 文件详细信息**来选择流文件的 PC 文件格式。数据传输支持公共 PC 文件类型，如 ASCII 文本、BIFF3、CSV、DIF、标记定界的文本或 WK4。
6. 单击**从 iSeries 传送数据**来运行文件传输。

也可以使用数据传输应用程序以批处理作业的形式执行此数据移动。按上述步骤进行，但选择**文件**菜单选项来保存传送请求。“将数据传送到 iSeries 服务器”应用程序创建一个 .DTT 或 .TFR 文件。“从 iSeries 服务器 传送数据”应用程序创建一个 .DTF 或 .TTO 文件。在 iSeries Access 目录中，两个程序都可以用批处理方式从命令行运行:

- RTOPCB 采用 .DTF 或 .TTO 文件作为参数
- RFROMPCB 采用 .DTT 或 .TFR 文件作为参数

通过使用调度程序，可以设置这两个命令中的任何一个按安排的时间运行。例如，可以使用“系统代理程序工具”（Microsoft Plus Pack 的一个部件）来指定要运行的程序（例如 `RTOPCB MYFILE.TTO`）和想要运行该程序的时间。

将数据从流文件传送到数据库文件

要将数据从流文件传送到服务器上的数据库文件。

1. 建立与服务器的连接。
2. 将网络驱动器映射到 iSeries 文件系统中的适当路径。
3. 从 iSeries Access Windows 版窗口选择**将数据传送到 iSeries 服务器**。
4. 选择要传送的 PC 文件名。对于 PC 文件名，可以选择**浏览**以获取您指定的网络驱动器，并选择一个流文件。也可以使用位于 PC 本身的流文件。
5. 选择要在其中定位外部描述的数据库文件的服务器。
6. 单击**将数据传送到 iSeries 服务器**来运行文件传输。

   注意: 若要将数据移动到服务器上的现有数据库文件定义，“将数据传送到 iSeries 服务器”应用程序要求您使用相关联的格式描述文件（FDF）。FDF 文件描述流文件的格式，该文件是“从 iSeries 服务器传送数据”应用程序在将数据从数据库文件传送到流文件时创建的。要完成将数据从流文件传送到数据库文件，单击**将数据传送到 iSeries**。如果现有的 .FDF 文件不可用，则可以快速创建一个 .FDF 文件。

也可以使用数据传输应用程序以批处理作业的形式执行此数据移动。按上述步骤进行，但选择**文件**菜单选项来保存传送请求。“将数据传送到 iSeries 服务器”应用程序创建一个 .DTT 或 .TFR 文件。“从 iSeries 服务器 传送数据”应用程序创建一个 .DTF 或 .TTO 文件。在 iSeries Access 目录中，两个程序都可以用批处理方式从命令行运行:

- RTOPCB 采用 .DTF 或 .TTO 文件作为参数
- RFROMPCB 采用 .DTT 或 .TFR 文件作为参数

通过使用调度程序，可以设置这两个命令中的任何一个按安排的时间运行。例如，可以使用“系统代理程序工具”（Microsoft Plus Pack 的一个部件）来指定要运行的程序（例如 `RTOPCB MYFILE.TTO`）和想要运行该程序的时间。

将数据传送到新创建的数据库文件定义和文件

要将数据传送到新创建的数据库文件定义和文件:

1. 建立与服务器的连接。
2. 将网络驱动器映射到 iSeries 文件系统中的适当路径。
3. 从 iSeries Access Windows 版窗口选择**将数据传送到 iSeries 服务器**。
4. 打开“将数据传送到 iSeries 服务器”应用程序的**工具**菜单。
5. 选择**创建 iSeries 数据库文件**。

   将会出现一个向导，它允许您从现有的 PC 文件创建新的 iSeries 数据库文件。将要求您指定 iSeries 文件将基于的 PC 文件的名称、要创建的 iSeries 文件的名称以及几个其它的必要详细信息。此工具对给出的流文件进行语法分析，以确定产生的数据库文件中所需要的字段数目、类型和大小。然后，该工具可以在您的服务器上创建数据库文件定义。

创建格式描述文件

若要将数据移动到服务器上的现有数据库文件定义，“将数据传送到 iSeries 服务器”应用程序要求您使用相关联的格式描述文件（FDF）。FDF 文件描述流文件的格式，该文件是“从 iSeries 服务器传送数据”应用程序在将数据从数据库文件传送到流文件时创建的。

要创建 .FDF 文件:

1. 用与源流文件匹配的格式（字段数目和数据类型）创建一个外部描述的数据库文件。
2. 在数据库文件中创建一个临时数据记录。
3. 使用从 iSeries 服务器传送数据功能来从此数据库文件创建流文件及其相关联的 .FDF 文件。
4. 现在，可以使用将数据传送到 iSeries 服务器功能。对您想要传送的源流文件指定此 .FDF 文件。

在流文件和保存文件之间复制数据

保存文件与保存和恢复命令一起用来保存将会另外写入磁带或软盘的数据。也可以象数据库文件一样使用该文件来读取或写入包含保存／恢复信息的记录。保存文件也可以用来将对象发送到 SNADS 网络中的另一用户。

可以使用 CPY 命令来将保存文件复制到流文件和从流文件复制保存文件。但是，当将流文件复制回保存文件对象时，数据必须是有效的保存文件数据（它必须源自保存文件并已复制到流文件中）。

通过使用 PC 客户机，也可以访问保存文件并将数据复制到 PC 存储器或 LAN。但是记住，不能通过“网络文件系统”（NFS）访问保存文件中的数据。

使用 API 执行操作

对集成文件系统目录和流文件执行操作的应用程序接口（API）具有 C 语言函数的格式。您可以在两组函数之间进行选择，这两组函数都可以在使用“集成语言环境”（ILE）C/400 创建的程序中使用。

- OS/400 中包括的集成文件系统 C 语言函数。
- ILE C/400 许可程序提供的 C 语言函数。

集成文件系统函数只通过集成文件系统流 I/O 支持起作用。下列 API 受支持:

表 3. 集成文件系统 API

函数	描述
access()	确定文件的可访问性
accessx()	确定用户类的文件可访问性
chdir()	更改当前目录
chmod()	更改文件授权
chown()	更改文件的所有者和组
close()	关闭文件描述符
closedir()	关闭目录
creat()	创建新文件或覆盖现有的文件
creat64()	创建新文件或覆盖现有文件（允许大文件）
DosSetFileLocks()	锁定和解锁文件的字节范围。
DosSetFileLocks64()	锁定和解锁文件的字节范围（允许大文件）
DosSetRelMaxFH()	更改文件描述符的最大数目
dup()	复制打开的文件描述符
dup2()	将打开的文件描述符复制到另一个描述符
faccessx()	通过描述符确定用户类的文件可访问性
fchdir()	通过描述符更改当前目录
fchmod()	通过描述符更改文件授权
fchown()	通过描述符更改文件的所有者和组
fcntl()	执行文件控制操作
fpathconf()	通过描述符获取可配置的路径名变量
fstat()	通过描述符获取文件信息
fstat64()	通过描述符获取文件信息（允许大文件）
fstatvfs()	通过描述符获取信息
fstatvfs64()	通过描述符获取信息（允许 64 位）
fsync()	使文件的更改同步
ftruncate()	截断文件
ftruncate64()	截断文件（允许大文件）
getcwd()	获取当前目录的路径名
getegid()	获取有效的组标识
geteuid()	获取有效的用户标识
getgid()	获取真实的组标识
getgrgid()	使用组标识获取组信息
getgrnam()	使用组名获取组信息
getgroups()	获取组标识
getwpnam()	获取用户名的用户信息
getpwuid()	获取用户标识的用户信息
getuid()	获取真实的用户标识
givedescriptor()	将文件访问权授予另一作业

表 3. 集成文件系统 API (续)

函数	描述
ioctl()	执行文件 I/O 控制操作
link()	创建到文件的链接
lseek()	设置文件读 / 写偏移量
lseek64()	设置文件读 / 写偏移量（允许大文件）
lstat()	获取文件或链接信息
lstat64()	获取文件或链接信息（允许大文件）
mmap()	创建内存映射
mmap64()	创建内存映射（允许大文件）
mprotect()	更改内存映射保护
msync()	使内存映射同步
munmap()	除去内存映射
mkdir()	生成目录
mkfifo()	生成 FIFO 特殊文件
open()	打开文件
open64()	打开文件（允许大文件）
opendir()	打开目录
pathconf()	获取可配置的路径名变量
pipe()	创建与套接字的进程间通道
pread()	使用偏移量读取描述符
pread64()	使用偏移量读取描述符（允许大文件）
pwrite()	使用偏移量写入到描述符
pwrite64()	使用偏移量写入到描述符（允许大文件）
QjoEndJournal()	结束日志记录
QjoRetrieveJournal Information()	检索日志信息
QJORJIDI()	检索日志标识符信息
QJOSJRNE()	发送日志项
QjoStartJournal()	启动日志记录
QlgAccess()	确定文件可访问性（使用启用了 NLS 的路径名）
QlgAccessx()	确定用户类的文件可访问性（使用启用了 NLS 的路径名）
QlgChdir()	更改当前目录（使用启用了 NLS 的路径名）
QlgChmod()	更改文件授权（使用启用了 NLS 的路径名）
QlgChown()	更改文件的所有者和组（使用启用了 NLS 的路径名）
QlgCreat()	创建新文件或覆盖现有文件（使用启用了 NLS 的路径名）
QlgCreat64()	创建新文件或覆盖现有文件（允许大文件并使用启用了 NLS 的路径名）
QlgCvtPathToQSYSObjName()	将“集成文件系统”路径名解析为“QSYS 对象名称”（使用启用了 NLS 的路径名）
QlgGetAttr()	获取对象的系统属性（使用启用了 NLS 的路径名）
QlgGetcwd()	获取当前目录的路径名（使用启用了 NLS 的路径名）

表 3. 集成文件系统 API (续)

函数	描述
QlgGetPathFromFileID()	从对象的文件标识获取对象的路径名（使用启用了 NLS 的路径名）
QlgGetpwnam()	获取用户名的用户信息（使用启用了 NLS 的路径名）
QlgGetpwnam_r()	获取用户名的用户信息（使用启用了 NLS 的路径名）
QlgGetpwuid()	获取用户标识的用户信息（使用启用了 NLS 的路径名）
QlgGetpwuid_r()	获取用户标识的用户信息（使用启用了 NLS 的路径名）
QlgLchown()	更改符号链接的所有者和组（使用启用了 NLS 的路径名）
QlgLink()	创建到文件的链接（使用启用了 NLS 的路径名）
QlgLstat()	获取文件或链接信息（使用启用了 NLS 的路径名）
QlgLstat64()	获取文件或链接信息（允许大文件并使用启用了 NLS 的路径名）
QlgMkdir()	生成目录（使用启用了 NLS 的路径名）
QlgMkfifo()	生成 FIFO 特殊文件（使用启用了 NLS 的路径名）
QlgOpen()	打开文件（使用启用了 NLS 的路径名）
QlgOpen64()	打开文件（允许大文件并使用启用了 NLS 的路径名）
QlgOpendir()	打开目录（使用启用了 NLS 的路径名）
QlgPathconf()	获取可配置的路径名变量（使用启用了 NLS 的路径名）
QlgProcessSubtree()	处理目录树中的目录或对象（使用启用了 NLS 的路径名）
QlgReaddir()	读取目录项（使用启用了 NLS 的路径名）
QlgReaddir_r()	读取目录项（线程安全并使用启用了 NLS 的路径名）
QlgReadlink()	读取符号链接的值（使用启用了 NLS 的路径名）
QlgRenameKeep()	重命名文件或目录，保持*新建*（若它存在）（使用启用了 NLS 的路径名）
QlgRenameUnlink()	重命名文件或目录，解除*新建*链接（若它存在）（使用启用了 NLS 的路径名）
QlgRmdir()	除去目录（使用启用了 NLS 的路径名）
QlgSaveStgFree()	保存对象数据并释放其存储器（使用启用了 NLS 的路径名）
QlgSetAttr()	设置对象的系统属性（使用启用了 NLS 的路径名）
QlgStat()	获取文件信息（使用启用了 NLS 的路径名）
QlgStat64()	获取文件信息（允许大文件并使用启用了 NLS 的路径名）
QlgStatvfs()	获取文件系统信息（使用启用了 NLS 的路径名）
QlgStatvfs64()	获取文件系统信息（允许大文件并使用启用了 NLS 的路径名）
QlgSymlink()	生成符号链接（使用启用了 NLS 的路径名）
QlgUnlink()	解除文件链接（使用启用了 NLS 的路径名）
QlgUtime()	设置文件访问权和修改次数（使用启用了 NLS 的路径名）
QP0FPTOS()	执行其它文件系统功能
Qp0lCvtPathToSYSObjName()	将集成文件系统路径名解析为 “QSYS 对象名称”
Qp0lFLOP()	对对象执行杂项操作
Qp0lGetAttr()	获取对象的系统属性
Qp0lGetPathFromFileID()	从对象的文件标识获取对象的路径名
Qp0lOpen()	用启动了 NLS 的路径名打开文件

表 3. 集成文件系统 API (续)

函数	描述
Qp0lProcessSubtree()	处理目录树中的目录或对象
Qp0lRenameKeep()	重命名文件或目录，保持*新建*（若它存在）
Qp0lRenameUnlink()	重命名文件或目录，取消链接*新建*（若它存在）
QP0LROR()	检索对象引用
Qp0lSaveStgFree()	保存对象数据并释放其存储器
Qp0lSetAttr()	设置对象的系统属性
Qp0lUnlink()	用启用了 NLS 的路径名解除文件的链接
qsysetegid()	设置有效的组标识
qsyseteuid()	设置有效的用户标识
qsysetgid()	设置组标识
qsysetregid()	设置真实有效的组标识
qsysetreuid()	设置真实有效的用户标识
qsysetuid()	设置用户标识
QZNFRTVE()	检索 NFS 导出信息
read()	读取文件
readdir()	读取目录项
readdir_r()	读取目录项（线程安全）
readlink()	读取符号链接的值
readv()	读取文件（向量）
rename()	重命名文件或目录。可以定义它具有 Qp0lRenameKeep() 或 Qp0lRenameUnlink() 的语义
rewinddir()	复位目录流
rmdir()	除去目录
select()	检查多个文件描述符的 I/O 状态
stat()	获取文件信息
stat64()	获取文件信息（允许大文件）
statvfs()	获取文件系统信息
statvfs64()	获取文件系统信息（允许大文件）
symlink()	生成符号链接
sysconf()	获取系统配置变量
takedescriptor()	从另一个作业获取文件访问权
umask()	设置作业的授权掩码
unlink()	除去到文件的链接
utime()	设置文件访问权和修改次数
write()	写入到文件
writev()	写入到文件（向量）

注: 其中某些函数也用于 OS/400 套接字。有关对特定文件系统使用这些函数的限制，参见第 4 页的『集成文件系统中的文件系统』。有关使用集成文件系统 C 语言函数的示例程序，参见第 87 页的附录 B, 『使用集成文件系统 C 函数的示例程序』。

有关集成文件系统 API 的更多信息，参考下列主题:

- ILE C/400 函数
- API 的大文件支持
- API 的路径名规则
- 文件描述符
- 安全性
- “iSeries 信息中心”中的应用程序编程接口（API）主题

ILE C/400 函数

ILE C/400 提供了由美国国家标准学会（ANSI）定义的标准 C 语言函数。这些函数可以通过数据管理 I/O 支持进行操作，也可以通过集成文件系统流 I/O 支持进行操作，这取决于您在创建 C 语言程序时所指定的内容。编译器使用数据管理 I/O，除非您通知它使用不同的方法。

要通知编译器使用集成文件系统流 I/O，必须在“创建 ILE C/400 模块”（CRTCMOD）或“创建绑定 C 语言程序”（CRTBNDC）命令中对“系统接口选项”（SYSIFCOPT）参数指定 *IFSIO。指定 *IFSIO 时，编译器绑定集成文件系统 I/O 函数，而不是绑定数据管理 I/O 函数。事实上，ILE C/400 的 C 语言函数使用集成文件系统函数来执行 I/O。

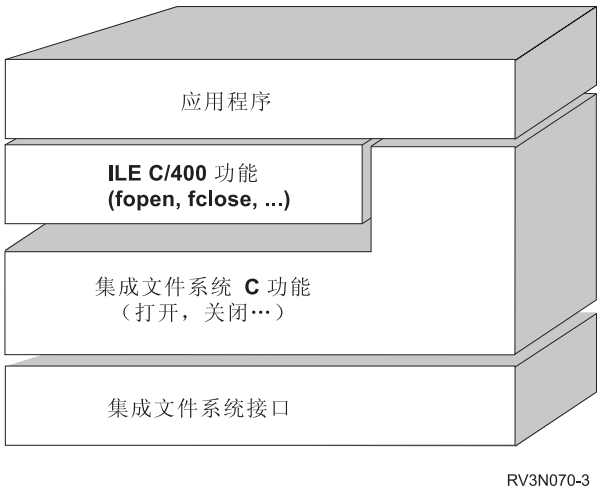


图 10. ILE C/400 函数使用集成文件系统流 I/O 函数。

有关将 ILE C/400 函数用于集成文件系统流 I/O 的更多信息，参见出版物 WebSphere Development Studio: ILE C/C++ Programmers Guide 。有关每个 ILE C/400 C 语言函数的详细信息，参见出版物 WebSphere Development Studio: C/C++ Language Reference 。

API 的大文件支持

增强了集成文件系统 API 以允许应用程序存储和操纵非常大的文件。在“根”（/）、“开放系统文件系统”（QOpenSys）和用户定义的文件系统中，集成文件系统允许多达 256 吉字节的流文件大小。

集成文件系统提供了一组 64 位 UNIX 类型的 API，并允许轻松将现有的 32 位 API 映射到 64 位 API，以便可以通过使用 8 个字节的整数自变量来访问大文件的大小和偏移量。有关每个 64 位 API 的详细信息，参见“iSeries 信息中心”中的集成文件系统 API 主题。

提供下列各项以允许应用程序使用大文件支持:

1. 若编译时定义了宏标号 _LARGE_FILE_API，则应用程序可以访问启用了 64 位的 API 和数据结构。例如，应用程序要想使用 stat64() API 和 stat64 结构，将需要在编译时定义 _LARGE_FILE_API。
2. 若应用程序在编译时定义了宏标号 _LARGE_FILES，则将现有的 API 和数据结构映射到其 64 位版本。例如，如果应用程序在编译时定义了 _LARGE_FILES，则将对 stat() API 的调用映射到 stat64() API，并将 stat() 结构映射到 stat64() 结构。

打算使用大文件支持的应用程序可以在编译时定义 _LARGE_FILE_API 并直接对 64 位 API 编码，或者它们可以在编译时定义 _LARGE_FILES。然后，所有适当的 API 和数据结构自动映射到 64 位版本。

不打算使用大文件支持的应用程序不会受到影响，它们可以继续使用集成文件系统 API 而无需作任何修改。

API 的路径名规则

使用集成文件系统或 ILE C/400 API 对对象进行操作时，通过提供其目录路径来标识该对象。以下是在 API 中指定路径名时应牢记的规则摘要。这些规则中**对象**这个术语表示任何目录、文件、链接或其它对象。

- 按分层次序指定路径名，从目录层次结构的最高级开始。路径中每个部分的名称用斜杠（/）隔开；例如:

  ```
  Dir1/Dir2/Dir3/UsrFile
  ```

  未将反斜杠（[<]dbcs）识别为分隔符。只将它作为名称中的另一个字符进行处理。
- 对象名在一个目录中必须是唯一的。
- 路径名每个部分的最大长度和路径名字符串的最大长度对于每个文件系统各不相同。有关每个文件系统中的限制，参见第 51 页的『文件系统比较』。
- 路径名开始处的 / 字符表示该路径从“根”（/）目录开始；例如:

  ```
  /Dir1/Dir2/Dir3/UsrFile
  ```
- 若路径名不以 / 字符开始，则假定路径从当前目录开始；例如:

  ```
  MyDir/MyFile
  ```

  其中 MyDir 是当前目录的子目录。
- 要避免与 iSeries 服务器特殊值产生混淆，路径名不能以单个星号（*）字符开始。要指定从任何字符数目开始的路径名，使用两个星号（*）；例如:

  ```
  '**.file'
  ```

  注意: 这只适合于其中星号（*）前没有其它字符的相对路径名。
- 当对 QSYS.LIB 文件系统中的对象执行操作时，部分名的格式必须为 *name.object-type*；例如:

  ```
  /QSYS.LIB/PAYROLL.LIB/PAY.FILE
  ```

| 有关更多详细信息，参见第 60 页的『库文件系统（QSYS.LIB）』。

| • 当对“独立 ASP QSYS.LIB”文件系统中的对象执行操作时，部分名的格式必须为 *name.object-type*；例如:

| ```
| '/asp_name/QSYS.LIB/PAYDAVE.LIB/PAY.FILE
| ```

| 有关详细信息，参见第 62 页的『独立 ASP QSYS.LIB』。

- 不要在路径名中使用冒号（:）。冒号在服务器中具有特殊意义。

| • 与集成文件系统命令中的路径名不同（参见第 24 页的『CL 命令和屏幕的路径名规则』），星号（*）、问号（?）、撇号（'）、引号（"）和代字号（[]）没有特殊意义。系统处理它们，就好象它们只是名称中的另一个字符。对此规则唯一例外的 API 是 QjoEndJournal() 和 QjoStartJournal。

文件描述符

使用美国国家标准学会（ANSI）定义的 ILE C/400 流 I/O 函数对文件执行操作时，通过使用指针来标识文件。使用集成文件系统 C 语言函数时，通过指定**文件描述符**来标识文件。文件描述符是一个正整数，在每个作业中它必须是唯一的。作业在对打开的文件执行操作时使用文件描述符来标识该文件。在集成文件系统上进行操作的 C 语言函数中，文件描述符由变量 *fildes* 表示，在套接字上进行操作的 C 语言函数中，文件描述符由变量 *descriptor* 表示。

每个文件描述符表示一个**打开文件描述**，该描述包含诸如文件偏移量、文件状态和文件的访问方式。同一个打开文件描述可以由多个文件描述符表示，但一个文件描述符只能表示一个打开文件描述。

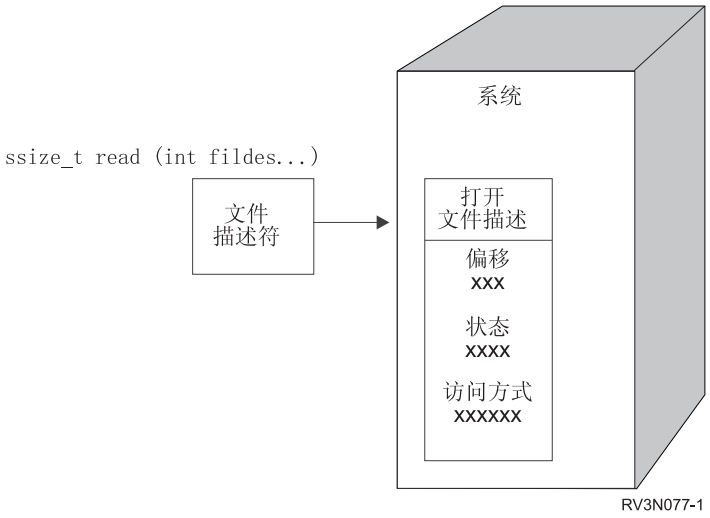


图 11. 文件描述符和打开文件描述

若将 ILE C/400 流 I/O 函数用于集成文件系统，则 ILE C/400 运行时支持将文件指针转换为文件描述符。

使用“根”（/）、QOpenSys 或用户定义文件系统时，可以将对打开文件描述的访问权从一个作业传送给另一个作业，从而允许后者访问文件。通过使用 givedescriptor() 或 takedescriptor() 函数在作业之间传送文件描述符，可以执行此操作。有关这些函数的描述，参见套接字编程或“iSeries 信息中心”中的套接字 API 主题。

安全性

使用集成文件系统 API 时，可以象使用数据管理接口时一样限制对对象的访问。但是，应知道沿用权限是不受支持的。集成文件系统 API 使用作业正在其下运行的用户概要文件的权限。

每个文件系统可能有其自己的特殊权限需求。NFS 服务器作业是此规则的唯一例外。“网络文件系统”服务器请求在一个用户的概要文件之下运行，该用户的用户标识（UID）号在请求时被 NFS 服务器接收到。

服务器上的权限相当于 UNIX 系统上的**许可权**。许可权的类型为读和写（对于文件或目录）以及执行（对于文件）或搜索（对于目录）。许可权由一组许可权位指示，这些许可权位组成了文件或目录的“访问方式”。可以通过使用“更改方式”函数 chmod() 或 fchmod() 更改许可权位。也可以使用 umask() 函数来控制每次作业创建文件时设置哪些文件许可权位。

有关数据安全性和权限的详细信息，参见出版物 Security — Reference 。

套接字支持

若应用程序正在使用“根”、QOpenSys 或用户定义文件系统，则可以利用集成文件系统**本地套接字**支持。本地套接字对象（对象类型 *SOCKET）允许在同一系统上运行的两个作业建立相互之间的通信连接。

一个作业通过使用 bind() C 语言函数来创建本地套接字对象，建立连接点。另一个作业在 connect()、sendto() 或 sendmsg() 函数中指定本地套接字对象的名称。“iSeries 信息中心”中的套接字编程主题中对这些函数和套接字概念进行了一般描述。

在建立连接之后，两个作业可以通过使用诸如 write() 和 read() 的集成文件系统函数互相发送和接收数据。传送的数据实际上都不会通过套接字对象。套接字对象只是一个交汇点，两个作业可以在该点互相找到对方。

当两个作业完成通信时，每个作业使用 close() 函数来关闭套接字连接。本地套接字对象保留在系统中，直到使用 unlink() 函数或除去链接（RMVLNK）命令将该对象除去为止。

不能保存本地套接字对象。

命名和国际支持

“根”（/）和 QOpenSys 文件系统的支持确保对象名中的字符在跨越用于不同本地语言和设备的编码模式时保持不变。将对象名传送到系统时，该名称的每个字符被转换为 16 位格式，在这种格式中，所有字符都有一个标准编码的表示（参见第 20 页的『名称连续性』）。使用该名称时，它被转换为适合于正在使用的代码页的编码格式。

如果该名称正在转换到的代码页不包含名称中使用的字符，则将该名称作为无效名称拒绝。

由于字符在跨代码页时保持不变，所以不应该假定在使用特定代码页时，特定字符将会更改为另一个特定字符。例如，不应该假定数字符号字符将会更改为英镑字符，即使它们可能在不同代码页中具有相同的编码表示。

注意：因为对象扩展属性的名称的转换方式与对象名称的转换方式相同，所以它们的注意事项是相同的。

有关代码页的更多信息，参见“iSeries 信息中心”中的全球化主题。

数据转换

当通过集成文件系统访问文件时，可能转换也可能不转换文件中的数据，这取决于打开文件时请求的打开方式。

打开的文件可以具有两种打开方式之一：

二进制 从文件中读取数据并写入到文件时不进行转换。应用程序负责处理数据。

文本 从文件中读取数据和将数据写入到文件时，假定数据的格式是文本。从文件中读取数据时，将数据从文件的编码字符集标识符（CCSID）转换到接收数据的应用程序、作业或系统的 CCSID。将数据写入文件时，将数据从应用程序、作业或系统的 CCSID 转换到文件的 CCSID。对于真正的流文件，只是将任何行格式化字符（如回车符、制表符和文件结束符）从一个 CCSID 转换到另一个 CCSID。

从正在用作流文件的记录文件中进行读取时，行结束字符（回车和换行）被追加到每条记录的数据末尾。当写入到记录文件时：

- 行结束字符被删除。
- 制表字符由到下一个制表位的适当数目的空格替换。

- 用空格（对于源物理文件成员）或空字符（对于数据物理文件成员）填充行，直到记录结束。

在打开请求中，可以指定下列其中一项:

二进制，强制
将数据作为二进制处理，而不管数据的实际内容如何。应用程序对了解如何处理数据负责。

文本，强制
假定数据是文本。将数据从文件的 CCSID 转换到应用程序的 CCSID。

缺省值*二进制，强制*用于集成文件系统 open() 函数。

第 6 章 集成文件系统中的文件系统

集成文件系统中的文件系统有:

- “根”
- QOpenSys
- UDFS
- QSYS.LIB
- 独立 ASP QSYS.LIB
- QDLS
- QOPT
- QNetWare
- QNTC
- QFileSvr.400
- NFS

有关每个文件系统的概述，参见文件系统比较。

文件系统比较

表 4 和第 52 页的表 5 概述了每个文件系统的特征和限制。

表 4. 文件系统摘要（第 1 部分，共两部分）

能力	/（“根”）	QOpenSys	QSYS.LIB[16]	QDLS	QNTC
OS/400 的标准部件	是	是	是	是	是
文件类型	流	流	记录[12]	流	流
与 OfficeVision 集成在一起（例如，可以邮寄文件）	否	否	否	是	否
通过 OS/400 文件服务器访问	是	是	是	是	是
通过 file server I/O processor 直接访问[1]	否	否	否	否	是
打开／关闭的比较速度	中等[2]	中等[2]	低[2]	低[2]	中等[2]
区分大小写的名称搜索	否	是	否[4]	否[5]	否
路径名中每个部件的最大长度	255 个字符	255 个字符	10.6 个字符[6]	8.3 个字符[7]	255 个字符
路径名的最大长度[8]	16MB	16MB	55 – 66 个字符[4]	82 个字符	255 个字符
对象扩展属性的最大长度	2GB	2GB	变化[9]	32KB	64KB
文件系统中目录层次结构的最高级别	无限制[10]	无限制[10]	3	32	127
每个对象的最大链接数[11]	变化[15]	变化[15]	1	1	1
支持符号链接	是	是	否	否	否
对象／文件可以具有所有者	是	是	是	是	否
支持集成文件系统命令	是	是	是	是	是
支持集成文件系统 API	是	是	是	是	是
支持分层文件系统（HFS）API	否	否	否	是	否

表 4. 文件系统摘要（第 1 部分，共两部分） (续)

能力	/（“根”）	QOpenSys	QSYS.LIB[16]	QDLS	QNTC
线程安全[13]	是	是	是	否	是
支持对象日志记录	是	是	是[14]	否	否

注释:

1. file server I/O processor 是由 LAN Server 使用的硬件。
2. 通过 OS/400 文件服务器文件服务器访问时。
3. 通过 LAN Server 客户机 PC 访问时。使用 iSeries API 进行访问的速度比较慢。
4. QSYS.LIB 文件系统具有最大路径名称长度 55 个字符。有关详细信息，参见第 60 页的『库文件系统（QSYS.LIB）』。“独立 ASP QSYS.LIB”文件系统具有最大路径长度 66 个字符。有关详细信息，参见第 62 页的『独立 ASP QSYS.LIB』。
5. 有关详细信息，参见第 64 页的『文档库服务文件系统（QDLS）』。
6. 对象名最多 10 个字符，对象类型最多 6 个字符。有关更多详细信息，参见第 60 页的『库文件系统（QSYS.LIB）』。
7. 名称最多 8 个字符，文件类型扩展名（若有的话）最多 3 个字符。有关详细信息，参见第 64 页的『文档库服务文件系统（QDLS）』。
8. 假定绝对路径名以 / 开始，后跟文件系统名（如 /QDLS...）.
9. QSYS.LIB 文件系统和“独立 ASP QSYS.LIB”文件系统支持三种预定义的扩展属性: .SUBJECT、.CODEPAGE 和 .TYPE。最大长度由这三种扩展属性的组合长度确定。
10. 实际上，目录级限于程序和系统空间限制。
11. 目录除外，它只能有一个到另一目录的链接。
12. QSYS.LIB 文件系统和“独立 ASP QSYS.LIB”文件系统中的用户空间支持流文件输入和输出。
13. 当将操作定向至驻留在线程安全文件系统中的对象时，集成文件系统 API 是线程安全的。对于多个线程在作业中运行时，当 API 正在对不是线程安全的文件系统中的对象执行操作时，则该 API 将失败。
14. QSYS.LIB 文件系统和“独立 ASP QSYS.LIB”文件系统可以支持对不同于根、UDFS 和 QOpenSys 文件系统的对象类型进行日志记录。有关驻留在 QSYS.LIB 或“独立 ASP QSYS.LIB”文件系统中的日志记录对象的更多信息，参见“iSeries 信息中心”的日志管理主题。
15. *TYPE2 目录具有每个对象一百万个链接的限制。*TYPE1 目录具有每个对象 32, 767 个链接的限制。有关更多信息，参见 *TYPE2 目录。
16. 此列中的数据引用 QSYS.LIB 文件系统和“独立 ASP QSYS.LIB”文件系统。

缩写

char = 字符

B = 字节　KB = 千字节　MB = 兆字节　GB = 千兆字节

表 5. 文件系统摘要（第 2 部分，共两部分）

能力	QOPT	QFileSvr.400	UDFS	NFS	QNetWare
OS/400 的标准部件	是	是	是	是	否
文件类型	流	流	流	流	流
与 OfficeVision 集成在一起（例如，可以邮寄文件）	否	否	否	否	否
通过 OS/400 文件服务器访问	是	是	是	是	是
通过集成 PC 服务器（FSIOP）直接访问[1]	否	否	否	否	是

表 5. 文件系统摘要（第 2 部分，共两部分） (续)

能力	QOPT	QFileSvr.400	UDFS	NFS	QNetWare
打开／关闭的比较速度	低	低[2]	中等[2]	中等[2]	高[11]
区分大小写的名称搜索	否	否[2]	是[12]	改变[2]	否
路径名中每个部件的最大长度	改变[4]	改变[2]	255 个字符	改变[2]	255 个字符[13]
路径名的最大长度	294 个字符	无限制[2]	16MB	无限制[2]	255 个字符
对象扩展属性的最大长度	8MB	0[6]	2GB[10]	0[6]	64KB
文件系统中目录层次结构的最高级别	无限制[7]	无限制[2]	无限制[7]	无限制[2]	100
每个对象的最大链接数[7]	1	1	变化[15]	变化[2]	1
支持符号链接	否	否	是	是[2]	否
对象／文件可以具有所有者	否	否[9]	是	是[2]	是
支持集成文件系统命令	是	是	是	是	是
支持集成文件系统 API	是	是	是	是	是
支持分层文件系统（HFS）API	是	否	否	否[2]	否
线程安全[14]	是	否	是	否	否
支持对象日志记录	否	否	是	否	否

注释:

1. file server I/O processor是由 LAN Server 使用的硬件。
2. 取决于正在访问哪个远程文件系统。
3. 通过 OS/400 文件服务器访问时
4. 有关详细信息，参见第 66 页的『光盘文件系统（QOPT）』。
5. 假定绝对路径名以 / 开始，后跟文件系统名。
6. QFileSvr.400 文件系统不返回扩展属性，即使正在被访问的文件系统支持扩展属性。
7. 实际上，目录级限于程序和系统空间限制。
8. 目录除外，它只能有一个到另一目录的链接。
9. 正在被访问的文件系统可能支持对象所有者。
10. UDFS 的扩展属性本身最大长度不能超过 40 个字节。
11. 通过 Novell NetWare 客户机 PC 访问时。使用 iSeries API 进行访问的速度比较慢。
12. 在创建 UDFS 时可以指定区分大小写性。若创建 UDFS 时使用 *MIXED 参数，则该 UDFS 将允许区分大小写的搜索。
13. NetWare “目录服务” 对象的最大值为 255 个字符。文件和目录限于 DOS 8.3 格式。
14. 集成文件系统 API 在支持多线程的进程中被访问时是线程安全的。该文件系统不允许访问不是线程安全的文件系统。
15. *TYPE2 目录具有每个对象一百万个链接的限制。*TYPE1 目录具有每个对象 32, 767 个链接的限制。有关更多信息，参见 *TYPE2 目录。

缩写

char = 字符

B = 字节　KB = 千字节　MB = 兆字节　GB = 千兆字节

“根”（/）文件系统

“根”（/）文件系统充分利用了集成文件系统的流文件支持和分层目录结构。“根”（/）文件系统具有“磁盘操作系统”（DOS）和 OS/2 文件系统的特征。

而且:

- 它最适合流文件的输入／输出。
- 它支持多个硬链接和符号链接。
- 它支持本地套接字。
- 它支持 *OOPOOL 对象。
- 它支持线程安全应用程序接口（API）。
- 它支持 *FIFO 对象。
- 它支持 /dev/null 和 /dev/zero *CHRSF 对象以及其它 *CHRSF 对象。
- 它支持对象更改的日志记录。

“根”（/）文件系统具有对名为 /dev/null 和 /dev/zero 的字符特殊文件（*CHRSF）的支持。字符特殊文件与计算机系统的设备或资源相关联。它们的路径名出现在目录中且具有与正常文件相同的访问保护。/dev/null 或 /dev/zero 字符特殊文件始终是空的，且废弃写入至 /dev/null 或 /dev/zero 的任何数据。文件 /dev/null 和 /dev/zero 具有对象类型 *CHRSF，除了不曾在 /dev/null 文件中读取数据外，可以象正常文件一样使用它，且 /dev/zero 文件始终成功返回，但已将数据清除为零。

有关“根”（/）文件系统的更多信息，参见使用“根”（/）文件系统。

使用“根”（/）文件系统

可以通过集成文件系统接口使用 OS/400 文件服务器或集成文件系统命令、用户屏幕和 C 语言 API 来访问“根”（/）文件系统。

“根”（/）文件系统中的区分大小写性

该文件系统保持输入对象名称时的相同大小写格式，但服务器搜索名称时不区分大写和小写。

“根”（/）文件系统中的路径名

- 路径名具有下列格式:

  ```
  Directory/Directory . . . /Object
  ```

- 共享名之后的路径名的各个部分遵循 Windows NT 路径名规则。组合的目录名和对象名的最大长度为 255 个字符。
- 除程序和服务器空间限制之外，目录层次结构的深度没有限制。
- 存储名称时，将名称中的字符转换为 UCS2 Level 1 格式（对于 *TYPE1 目录）和 UTF-16（对于 *TYPE2 目录）（参见第 20 页的『名称连续性』）。有关目录格式的更多信息，参考 *TYPE2 目录。

“根”（/）文件系统中的链接

“根”（/）文件系统中允许到同一对象的多个硬链接。完全支持符号链接。符号链接可用来从“根”（/）文件系统链接到另一个文件系统（如 QSYS.LIB、“独立 ASP QSYS.LIB”或 QDLS）中的对象。

有关链接的描述，参见第 16 页的『链接』。

在“根”（/）文件系统中使用集成文件系统命令

第 22 页的『使用 CL 命令执行操作』中列出的所有命令和第 21 页的『使用 iSeries 菜单和屏幕执行操作』中描述的所有屏幕都可以对“根”（/）文件系统执行操作。但是，在支持多线程的进程中使用这些命令可能不安全。

在“根”（/）文件系统中使用集成文件系统 API

第 40 页的『使用 API 执行操作』中列示的所有 C 语言 API 都可以对“根”（/）文件系统以线程安全方式执行操作。

将“根”（/）文件系统中的对象更改记入日志

可以将“根”（/）文件系统中的对象记入日志。日志管理的主要目的是使您能够恢复对象自上次保存以来发生的更改。有关“根”（/）文件系统中的对象更改日志记录的更多信息，参见第 79 页的第 7 章，『集成文件系统对象的日志记录支持』。

开放系统文件系统（QOpenSys）

QOpenSys 文件系统与基于 UNIX 的开放系统标准（如 POSIX 和 XPG）兼容。像“根”（/）文件系统一样，此文件系统也利用集成文件系统提供的流文件和目录支持。

而且:

- 可以通过类似于 UNIX 系统的分层目录结构访问它。
- 它最适合流文件的输入／输出。
- 它支持多个硬链接和符号链接。
- 它支持区分大小写的名称。
- 它支持本地套接字。
- 它支持 *OOPOOL 对象。
- 它支持线程安全 API。
- 它支持 *FIFO 对象。
- 它支持对象更改的日志记录。

QOpenSys 文件系统具有与“根”（/）文件系统相同的特征，但它区分大小写，以启用基于 UNIX 的开放系统标准的支持。

有关 QOpenSys 的更多信息，参见使用 QOpenSys。

有关 *TYPE 1 至 *TYPE2 目录转换和 QOpenSys 文件系统限制的信息，参见 *TYPE2 目录。

使用 QOpenSys

可以通过集成文件系统接口使用 OS/400 文件服务器或集成文件系统命令、用户屏幕和 C 语言 API 来访问 QOpenSys。

QOpenSys 文件系统中的区分大小写性

与“根”（/）文件系统不同，QOpenSys 文件系统在搜索对象名时区分大小写。例如，全是大写字符的字符串将不与其中任何一个字符是小写的相同字符串匹配。

区分大小写性允许您使用重复的名称，只要在组成名称的字符大写和小写中有一些差异。例如，可以在 QOpenSys 中的同一目录中具有一个名为 Payroll 的对象、一个名为 PayRoll 的对象和一个名为 PAYROLL 的对象。

QOpenSys 文件系统中的路径名

- 路径名具有下列格式:

```
Directory/Directory/ . . . /Object
```

- 路径名的每个部分的长度最多为 255 个字符。全路径名的长度最多为 16 MB。
- 除程序和服务器空间限制之外，目录层次结构的深度没有限制。
- 存储名称时，将名称中的字符转换为 UCS2 Level 1 格式（对于 *TYPE1 目录）和 UTF-16（对于 *TYPE2 目录）（参见第 20 页的『名称连续性』）。有关目录格式的更多信息，参考 *TYPE2 目录。

QOpenSys 文件系统中的链接

QOpenSys 文件系统中允许到同一对象的多个硬链接。完全支持符号链接。符号链接可用来从 QOpenSys 文件系统链接到另一个文件系统中的对象。

有关链接的描述，参见第 16 页的『链接』。

在 QOpenSys 文件系统中使用集成文件系统命令和屏幕

第 22 页的『使用 CL 命令执行操作』中列出的所有命令和第 21 页的『使用 iSeries 菜单和屏幕执行操作』中描述的所有屏幕都可以对 QOpenSys 文件系统执行操作。但是，在支持多线程的进程中使用这些命令可能不安全。

在 QOpenSys 文件系统中使用集成文件系统 API

第 40 页的『使用 API 执行操作』中列出的所有 C 语言函数可以对 QOpenSys 文件系统以线程安全方式执行操作。

将 QOpenSys 文件系统中的对象更改记入日志

可以将 QOpenSys 文件系统中的对象记入日志。日志管理的主要目的是使您能够恢复对象自上次保存以来发生的更改。有关 QOpenSys 文件系统中的对象更改日志记录的更多信息，参见第 79 页的第 7 章，『集成文件系统对象的日志记录支持』。

用户定义文件系统（UDFS）

UDFS 文件系统驻留在您选择的辅助存储池（ASP）或独立辅助存储池（ASP）上。您应创建和管理这些文件系统。

而且:

- 它们提供类似于 PC 操作系统（如 DOS 和 OS/2）的分层目录结构。
- 它们最适合流文件的输入／输出。
- 它们支持多个硬链接和符号链接。
- 它们支持本地套接字。
- 它们支持线程安全 API。
- 它们支持 *FIFO 对象。
- 它们支持对象更改的日志记录。

可以通过给每个 UDFS 指定一个唯一名称来创建多个 UDFS。可以在创建 UDFS 时为它指定其它属性，包括:

- 位于 UDFS 中的对象存储在其中的 ASP 号或独立 ASP 名称。
- 位于 UDFS 中的对象名的区分大小写性。

  UDFS 的区分大小写性确定在 UDFS 中搜索对象名时，大写和小写字符将是否匹配。

有关用户定义的文件系统的更多信息，参见下列主题:

- UDFS 概念
- 通过集成文件系统接口使用 UDFS

UDFS 概念

在 UDFS 中，与在“根”（/）和 QOpenSys 文件系统中一样，可以创建目录、流文件、符号链接、本地套接字和 SOM 对象。

单个块特殊文件对象（*BLKSF）表示一个 UDFS。创建 UDFS 时，也自动创建了块特殊文件。用户只能通过集成文件系统类属命令、API 和 QFileSvr.400 接口来访问块特殊文件。

UDFS 仅存在于两种状态: **安装**和**卸装**。安装 UDFS 时，该 UDFS 中的对象是可访问的。卸装 UDFS 时，该 UDFS 中的对象变成不可访问的。

为了访问 UDFS 中的对象，必须在目录（例如 /home/JON）中安装该 UDFS。在目录中安装 UDFS 时，该目录中原来的内容（包括对象和子目录）变成不可访问的。安装 UDFS 时，UDFS 的内容可以通过安装该 UDFS 的目录路径进行访问。例如，/home/JON 目录包含文件 /home/JON/payroll。UDFS 包含三个目录: mail、action 和 outgoing。在 /home/JON 上安装 UDFS 之后，/home/JON/payroll 文件是不可访问的，而三个 UDFS 目录变成可访问的，它们是 /home/JON/mail、/home/JON/action 和 /home/JON/outgoing。在卸装该 UDFS 之后，/home/JON/payroll 文件再次变成可访问的，而 UDFS 中的三个目录变成不可访问的。

| **注**: 不能安装独立 ASP 上的 UDFS。

要了解关于安装文件系统的更多信息，参见 OS/400 网络文件系统支持  。

通过集成文件系统接口使用 UDFS

可以使用 OS/400 文件服务器或集成文件系统命令、用户屏幕和 API 通过集成文件系统接口来访问 UDFS。在使用集成文件系统接口时，应知道下列注意事项和限制。

集成文件系统 UDFS 中的区分大小写性

可以指定在 UDFS 中创建对象名时，该名称是否区分大小写。

当选择区分大小写性时，则在搜索对象名时区分大写字符和小写字符。例如，全是大写字符的名称将不与其中任何一个字符是小写的相同字符串匹配。因此，/home/MURPH/ 和 /home/murph/ 被认为是不同目录。要创建区分大小写的 UDFS，可以在使用 CRTUDFS 命令时对 CASE 参数指定 *MIXED。

当选择不区分大小写性时，服务器在搜索名称时不区分大写和小写字符。因此，服务器将认为 /home/CAYCE 和 /HOME/cayce 是同一目录，而不是两个独立的目录。要创建不区分大小写的 UDFS，可以在使用 CRTUDFS 命令时对 CASE 参数指定 *MONO。

在这两种情况下，文件系统都保存用户输入对象名时的相同大小写格式。区分大小写性选项只适用于用户在服务器中如何搜索名称。

集成文件系统 UDFS 中的路径名

| 当需要操纵整个 UDFS 和其中的所有对象时，一个块特殊文件（*BLKSF）表示一个 UDFS。如果您的 UDFS
| 驻留在系统 ASP 中或基本用户 ASP 中，则块特殊文件名必须具有以下格式:

```
| /dev/QASPXX/udfs_name.udfs
```

| 其中，XX 是存储 UDFS 的 ASP 号，udfs_name 是 UDFS 在该 ASP 中的唯一名称。注意: UDFS 名必须以 .udfs 扩展名结束。

| 如果您的 UDFS 驻留在独立 ASP 中，则块特殊文件名必须具有以下格式:

| /dev/asp_name/udfs_name.udfs

| 其中，asp_name 是存储 UDFS 的独立 ASP 的名称，udfs_name 是 UDFS 在该独立 ASP 中的唯一名称。注意: UDFS 名必须以 .udfs 扩展名结束。

UDFS 中对象的路径名以您在其中安装 UDFS 的目录为基准。例如，若在 /home/dennis 目录上安装 UDFS /dev/qasp01/wysocki.udfs，则该 UDFS 中的所有对象的路径名将以 /home/dennis 开始。

| 附加路径名规则:

- 路径名的每个部分的长度最多为 255 个字符。全路径名的长度最多为 16 MB。
- 除程序和服务器空间限制之外，目录层次结构的深度没有限制。
- 存储名称时，将名称中的字符转换为 UCS2 Level 1 格式（对于 *TYPE1 目录）和 UTF-16（对于 *TYPE2 目录）（参见第 20 页的『名称连续性』）。有关目录格式的更多信息，参考 *TYPE2 目录。

集成文件系统 UDFS 中的链接

UDFS 中的对象允许到同一对象的多个硬链接并完全支持符号链接。符号链接可以从 UDFS 创建到另一个文件系统中对象的链接。

有关链接的描述，参见第 16 页的『链接』。

在 UDFS 中使用集成文件系统命令

第 22 页的『使用 CL 命令执行操作』中列出的所有命令和第 21 页的『使用 iSeries 菜单和屏幕执行操作』中描述的所有屏幕都可以对用户定义文件系统执行操作。有些 CL 命令是特定于用户定义文件系统和其它安装的一般文件系统。下表描述了这些命令。

表 6. 用户定义文件系统 CL 命令

命令	描述
ADDMFS	“添加安装的文件系统”将导出的远程服务器文件系统安装在本地客户机目录。
CRTUDFS	“创建 UDFS”创建用户定义文件系统。
DLTUDFS	“删除 UDFS”删除用户定义文件系统。
DSPMFSINF	“显示安装的文件系统信息”显示关于安装的文件系统的信息。
DSPUDFS	“显示 UDFS”显示关于用户定义文件系统的信息。
MOUNT	“安装文件系统”将导出的远程服务器文件系统安装在本地客户机目录。此命令是 ADDMFS 命令的别名。
RMVMFS	“除去安装的文件系统”将导出的远程服务器文件系统从本地客户机名称空间中除去。
UNMOUNT	“卸装文件系统”将导出的远程服务器文件系统从本地客户机名称空间中除去。此命令是 RMVMFS 命令的别名。

注: 在任何集成文件系统命令可以对存储在 UDFS 中的对象进行操作之前，必须安装该 UDFS。

在 UDFS 中使用集成文件系统 API

第 40 页的『使用 API 执行操作』中列出的所有 C 语言函数可以对用户定义文件系统执行操作。

注: 在任何集成文件系统命令可以对存储在 UDFS 中的对象进行操作之前，必须安装该 UDFS。

UDFS 的图形用户界面

“iSeries 导航器”（PC 上的图形用户界面）提供了访问 UDFS 的简便方法。此界面允许您从 Windows 客户机创建、删除、显示、安装和卸装 UDFS。

可以通过 “iSeries 导航器” 对 UDFS 执行操作。基本任务包括:

- 第 35 页的『创建新的用户定义的文件系统』。
- 第 35 页的『安装用户定义的文件系统』。
- 第 36 页的『卸装用户定义的文件系统』。

创建集成文件系统 UDFS

“创建用户定义的文件系统” 命令（CRTUDFS）创建一个文件系统，可以通过集成文件系统名称空间、API 和 CL 命令看到该文件系统。ADDMFS 或 MOUNT 命令将 UDFS 置于已经存在的本地目录的 “顶部”。可以在您选择的 ASP 或独立 ASP 中创建 UDFS。还可以指定区分大小写性。

删除集成文件系统 UDFS

“删除用户定义的文件系统” 命令（DLTUDFS）删除一个现有并卸装的 UDFS 以及其中的所有对象。若您已安装该 UDFS，则该命令将失败。删除 UDFS 将导致删除该 UDFS 中的所有对象。若没有适当的权限来删除 UDFS 中的所有对象，则将不会删除任何对象。

显示集成文件系统 UDFS

“显示用户定义的文件系统”（DSPUDFS）命令显示现有 UDFS 的属性，无论该 UDFS 是安装的还是卸装的。“显示安装的文件系统信息”（DSPMFSINF）命令也将显示关于安装的 UDFS 以及任何安装的文件系统的信息。

安装集成文件系统 UDFS

“添加安装的文件系统”（ADDMFS）和 MOUNT 命令使集成文件系统名称空间可以访问文件系统中的对象。要安装 UDFS，需要在 ADDMFS 命令中对 TYPE 参数指定 *UDFS。

| **注:** 不能安装独立 ASP 上的 UDFS。

卸装集成文件系统 UDFS

| 卸装命令使集成文件系统接口不能访问 UDFS 的内容。一旦卸装 UDFS，则将不能单独访问该 UDFS 中的对
| 象。“除去安装的文件系统”（RMVMFS）或 UNMOUNT 命令将使集成文件系统名称空间不能访问安装的文
| 件系统。若在使用该命令时文件系统中的任何对象正在使用中（例如，打开文件），则将接收到一条错误消
| 息。该 UDFS 将保持为安装状态。若已安装该 UDFS 的任何部分，则在显示该 UDFS 之前，不能卸装它。

例如，在集成文件系统名称空间中的 /home/judy 上安装 UDFS /dev/qasp02/jenn.udfs。若然后在 /home/judy 上安装另一个文件系统 /pubs，则 jenn.udfs 将变成不可访问的。而且，在从 /home/judy 中卸装第二个文件系统之前，不能卸装 jenn.udfs。

| **注:** 不能安装独立 ASP 上的 UDFS。

保存和恢复集成文件系统 UDFS

您有能力保存和恢复所有 UDFS 对象及其相关联的权限。“保存”命令（SAV）允许您保存 UDFS 中的对象，而“恢复”命令（RST）允许您恢复 UDFS 对象。无论 UDFS 是安装的还是卸装的，这两个命令都将起作用。但是，要正确地保存 UDFS 属性，而不仅是 UDFS 中的对象，则应该卸装 UDFS。

将 UDFS 文件系统中的对象更改记入日志

可以将用户定义的文件系统中的对象记入日志。日志管理的主要目的是使您能够恢复对象自上次保存以来发生的更改。有关 UDFS 文件系统中的对象更改日志记录的更多信息，参见第 79 页的第 7 章，『集成文件系统对象的日志记录支持』。

库文件系统（QSYS.LIB）

QSYS.LIB 文件系统支持 iSeries 服务器库结构。此文件系统允许您访问系统 ASP 和基本用户 ASP 中库支持所管理的数据库文件和所有其它 iSeries 服务器对象类型。

而且:

- 它支持对 iSeries 服务器库及这些库中的对象执行操作的所有用户界面和编程接口
- 它支持对数据库文件执行操作的所有编程语言和设施
- 它为管理 iSeries 服务器对象提供了广泛的管理支持
- 它支持对物理文件成员、用户空间和保存文件的流 I/O 操作

在 OS/400 的版本 3 之前，QSYS.LIB 文件系统可能已经被称为 iSeries 服务器文件系统。使用诸如 RPG 或 COBOL 的语言和诸如 DDS 的设施来开发应用程序的程序员使用的是 QSYS.LIB 文件系统。使用命令、菜单和屏幕来操纵输出队列的系统操作员使用的是 QSYS.LIB 文件系统，就象创建和更改用户概要文件的系统管理员一样。

所有这些设施和基于这些设施的应用程序与引进集成文件系统之前一样起作用。但是，这些设施不能通过集成文件系统接口访问 QSYS.LIB。

有关 QSYS.LIB 的更多信息，参见通过集成文件系统接口使用 QSYS.LIB。

通过集成文件系统接口使用 QSYS.LIB

可以通过集成文件系统接口使用 OS/400 文件服务器或集成文件系统命令、用户屏幕和 C 语言 API 来访问 QSYS.LIB 文件系统。在使用集成文件系统接口时，应知道下列注意事项和限制。

QSYS.LIB 文件系统中的 QPWFSERVER 授权列表

QPWFSERVER 是一个权限列表（对象类型 *AUTL），它对 QSYS.LIB 文件系统中正在通过远程客户机访问的所有对象提供了附加的访问需求。此授权列表中指定的权限适用于 QSYS.LIB 文件系统中的所有对象。

此对象的缺省权限是 PUBLIC *USE 权限。管理员可以使用 EDTAUTL（编辑授权列表）或 WRKAUTL（使用授权列表）命令来更改此权限的值。管理员可以将 PUBLIC *EXCLUDE 权限分配给权限列表，以便一般公众不能从远程客户机访问 QSYS.LIB 对象。

QSYS.LIB 文件系统中的文件处理限制

- 不支持逻辑文件。
- 文本方式访问支持的物理文件是包含单个字段的程序描述的物理文件和包含单个文本字段的源物理文件。二进制方式访问支持的物理文件包括外部描述的物理文件以及文本方式访问支持的那些文件。
- 不支持字节范围锁定。（有关字节范围锁定的更多信息，参见“iSeries 信息中心”中的 fcntl() 主题。）

- 若任何作业打开了一个数据库文件成员，则任何时候只授予一个作业对该文件成员的写访问权。只允许其它请求具有读访问权。

QSYS.LIB 文件系统中的用户空间支持

QSYS.LIB 支持对用户空间对象的流输入／输出操作。例如，程序可以将流数据写入到用户空间和从用户空间中读取数据。用户空间最大为 16 776 704 个字节。

应知道用户空间未用 CCSID（编码字符集标识符）进行标记。因此，返回的 CCSID 是作业的缺省 CCSID。

QSYS.LIB 文件系统中对保存文件的支持

QSYS.LIB 文件系统支持对保存文件对象的流 I/O 操作。例如，现有保存文件包含可以读出或复制到另一个文件的数据，直到必须将该数据放到另一个现有的空保存文件对象为止。当打开保存文件进行写入时，不允许文件的其它打开实例。保存文件**确实**允许多个进行读取的打开实例，只要没有任何作业具有文件的多个进行读取的打开实例。不能打开保存文件进行读／写访问。当在一个作业中运行多个线程时，不允许保存文件数据的流 I/O 操作。

当通过“网络文件系统”服务器导出保存文件或其目录时，不支持对保存文件的流 I/O 操作。但是，可以从 PC 客户机或通过 QFileSvr.400 文件系统访问它们。

QSYS.LIB 文件系统中的区分大小写性

一般来说，QSYS.LIB 文件系统不区分对象名中的大小写。无论对象名中的字符是大写还是小写，对该名称的搜索结果是相同的。

但是，若将名称用引号括起来，则保持名称中每个字符的大小写。因此，涉及加引号名称的搜索区分加引号名称中字符的大小写。

QSYS.LIB 文件系统中的路径名

- 路径名的每个部分都必须包含后跟对象的对象类型的对象名。例如:

  ```
  /QSYS.LIB/QGPL.LIB/PRT1.OUTQ

  /QSYS.LIB/EMP.LIB/PAY.FILE/TAX.MBR
  ```

  对象名和对象类型用句点（.）隔开。若库中的对象有不同的对象类型，则它们可以具有相同的名称，所以必须指定对象类型，以唯一标识该对象。
- 每个部分中对象名的长度最多为 10 个字符，对象名类型的长度最多为 6 个字符。
- QSYS.LIB 中目录层次结构的深度可以为两级或三级（路径名中有两个或三个部分），这取决于正在访问的对象的类型。若对象是数据库文件，则层次结构可以包含三级（库、文件和成员）；否则，只能有两级（库和对象）。每个部分名的长度和目录级数的组合确定路径名的最大长度。

  若包括“根”（/）和 QSYS.LIB 作为头两级，则 QSYS.LIB 的目录层次结构的深度最多可以为五级。
- 存储名称时，将名称中的字符转换为 CCSID 37。但是，使用作业的 CCSID 存储加引号的名称。

  有关 CCSID 的更多信息，参见“iSeries 信息中心”中的全球化主题。

QSYS.LIB 文件系统中的链接

不能在 QSYS.LIB 文件系统中创建或存储符号链接。

库和库中对象之间的关系与库和库中每个对象之间的一个硬链接是等价的。集成文件系统将库 — 对象关系作为链接来进行处理。因此，有可能从支持符号链接的文件系统链接到 QSYS.LIB 文件系统中的对象。

有关链接的描述，参见第 16 页的『链接』。

在 QSYS.LIB 文件系统中使用集成文件系统命令和屏幕

第 22 页的『使用 CL 命令执行操作』中列出的命令可以对 QSYS.LIB 文件系统执行操作，但下列情况除外:

- ADDLNK 命令只能用来创建*到* QSYS.LIB 中的对象的符号链接。
- 只能对程序描述的物理文件和源物理文件执行文件操作。
- 不能对数据库物理文件使用 STRJRN 和 ENDJRN 命令。

相同的限制适用于第 21 页的『使用 iSeries 菜单和屏幕执行操作』中描述的用户屏幕。

在 QSYS.LIB 文件系统中使用集成文件系统 API

第 40 页的『使用 API 执行操作』中列示的 C 语言函数可以对 QSYS.LIB 文件系统执行操作，但下列函数除外:

- 只能对程序描述的物理文件和源物理文件执行文件操作。
- symlink() 函数只能用来从支持符号链接的另一个文件系统链接到 QSYS.LIB 中的对象。
- 对数据库物理文件不能使用 QjoStartJournal() 和 QjoEndJournal() API。

独立 ASP QSYS.LIB

“独立 ASP QSYS.LIB”文件系统支持您创建和定义的独立辅助存储池（ASP）中的 iSeries 服务器库结构。此文件系统允许您访问独立 ASP 中库支持所管理的数据库文件和所有其它 iSeries 服务器对象类型。

而且:

- 它支持对独立 ASP 中的 iSeries 服务器库及那些库中的对象执行操作的所有用户界面和编程接口。
- 它支持对数据库文件执行操作的所有编程语言和设施
- 它为管理 iSeries 服务器对象提供了广泛的管理支持
- 它支持对物理文件成员、用户空间和保存文件的流 I/O 操作

有关“独立 ASP QSYS.LIB”文件系统的更多信息，参见通过集成文件系统接口使用“独立 ASP QSYS.LIB”。

通过集成文件系统接口使用“独立 ASP QSYS.LIB”

可以使用 OS/400 文件服务器或集成文件系统命令、用户屏幕和 C 语言 API 通过集成文件系统接口来访问“独立 ASP QSYS.LIB”文件系统。在使用集成文件系统接口时，应知道下列注意事项和限制。

“独立 ASP QSYS.LIB”文件系统中的 QPWFSERVER 权限列表

QPWFSERVER 是一个权限列表（对象类型为 *AUTL），它对“独立 ASP QSYS.LIB”文件系统中正在通过远程客户机访问的所有对象提供了附加的访问需求。此权限列表中指定的权限适用于“独立 ASP QSYS.LIB”文件系统中的所有对象。

此对象的缺省权限是 PUBLIC *USE 权限。管理员可以使用 EDTAUTL（编辑授权列表）或 WRKAUTL（使用授权列表）命令来更改此权限的值。管理员可以将 PUBLIC *EXCLUDE 权限分配给权限列表，以便一般公众不能从远程客户机访问“独立 ASP QSYS.LIB”对象。

“独立 ASP QSYS.LIB”文件系统中的文件处理限制

- 不支持逻辑文件。
- 文本方式访问支持的物理文件是包含单个字段的程序描述的物理文件和包含单个文本字段的源物理文件。二进制方式访问支持的物理文件包括外部描述的物理文件以及文本方式访问支持的那些文件。
- 不支持字节范围锁定。（有关字节范围锁定的更多信息，参见“iSeries 信息中心”中的 fcntl() 主题。）

- 若任何作业打开了一个数据库文件成员，则任何时候只授予一个作业对该文件成员的写访问权。只允许其它请求具有读访问权。

“独立 ASP QSYS.LIB”文件系统中的用户空间支持

“独立 ASP QSYS.LIB”支持对用户空间对象的流输入／输出操作。例如，程序可以将流数据写入到用户空间和从用户空间中读取数据。用户空间最大为 16 776 704 个字节。

应知道用户空间未用 CCSID（编码字符集标识符）进行标记。因此，返回的 CCSID 是作业的缺省 CCSID。

“独立 ASP QSYS.LIB”文件系统中的保存文件支持

“独立 ASP QSYS.LIB”支持对保存文件对象的流 I/O 操作。例如，现有保存文件包含可以读出或复制到另一个文件的数据，直到必须将该数据放到另一个现有的空保存文件对象为止。当打开保存文件进行写入时，不允许文件的其它打开实例。保存文件**确实**允许多个进行读取的打开实例，只要没有任何作业具有文件的多个进行读取的打开实例。不能打开保存文件进行读／写访问。当在一个作业中运行多个线程时，不允许保存文件数据的流 I/O 操作。

当通过“网络文件系统”服务器导出保存文件或其目录时，不支持对保存文件的流 I/O 操作。但是，可以从 PC 客户机或通过 QFileSvr.400 文件系统访问它们。

“独立 ASP QSYS.LIB”文件系统中的区分大小写性

一般来说，“独立 ASP QSYS.LIB”文件系统不区分对象名称中的大小写。无论对象名中的字符是大写还是小写，对该名称的搜索结果是相同的。

但是，若将名称用引号括起来，则保持名称中每个字符的大小写。因此，涉及加引号名称的搜索区分加引号名称中字符的大小写。

“独立 ASP QSYS.LIB”文件系统中的路径名称

- 路径名的每个部分都必须包含后跟对象的对象类型的对象名。例如:

```
/asp_name/QSYS.LIB/QGPL.LIB/PRT1.OUTQ

/asp_name/QSYS.LIB/EMP.LIB/PAY.FILE/TAX.MBR
```

  其中，asp_name 是独立 ASP 的名称。对象名和对象类型用句点（.）隔开。若库中的对象有不同的对象类型，则它们可以具有相同的名称，所以必须指定对象类型，以唯一标识该对象。
- 每个部分中对象名的长度最多为 10 个字符，对象名类型的长度最多为 6 个字符。
- “独立 ASP QSYS.LIB”中目录层次结构的深度可以为两级或三级（路径名中有两个或三个部分），这取决于正在访问的对象的类型。若对象是数据库文件，则层次结构可以包含三级（库、文件和成员）；否则，只能有两级（库和对象）。每个部分名的长度和目录级数的组合确定路径名的最大长度。

  如果包括 /、asp_name 和 QSYS.LIB 作为头三级，则“独立 ASP QSYS.LIB”文件系统的目录层次结构的深度最多可以为六级。
- 存储名称时，将名称中的字符转换为 CCSID 37。但是，使用作业的 CCSID 存储加引号的名称。

  有关 CCSID 的更多信息，参见“iSeries 信息中心”中的全球化主题。

“独立 ASP QSYS.LIB”文件系统中的链接

不能在“独立 ASP QSYS.LIB”文件系统中创建或存储符号链接。

库和库中对象之间的关系与库和库中每个对象之间的一个硬链接是等价的。集成文件系统将库 — 对象关系作为链接来进行处理。因此，有可能从支持符号链接的文件系统链接到“独立 ASP QSYS.LIB”文件系统中的对象。

| 有关链接的描述，参见第 16 页的『链接』。

在“独立 ASP QSYS.LIB”文件系统中使用集成文件系统命令和屏幕

第 22 页的『使用 CL 命令执行操作』中列示的命令可以对“独立 ASP QSYS.LIB”文件系统执行操作，但下列命令除外:

- ADDLNK 命令只能用来创建*到*“独立 ASP QSYS.LIB”中的对象的符号链接。
- 只能对程序描述的物理文件和源物理文件执行文件操作。
- 不能对数据库物理文件使用 STRJRN 和 ENDJRN 命令。
- 不能使用 MOV 命令将“独立 ASP QSYS.LIB”文件系统中的库移动到基本辅助存储池（ASP）。但是，可以将“独立 ASP QSYS.LIB”中的库移动到系统 ASP 或其它独立 ASP。
- 如果使用 SAV 或 RST 来保存或恢复独立 ASP 中的库对象，则该独立 ASP 必须与执行 SAV 或 RST 的作业相关联，或必须在 ASPDEV 参数中指定独立 ASP。路径名命名约定 /asp_name/QSYS.LIB/object.type 在 SAV 或 RST 上不受支持。

相同的限制适用于第 21 页的『使用 iSeries 菜单和屏幕执行操作』中描述的用户屏幕。

在“独立 ASP QSYS.LIB”文件系统中使用集成文件系统 API

第 40 页的『使用 API 执行操作』中列示的 C 语言函数可以对“独立 ASP QSYS.LIB”文件系统执行操作，但下列函数除外:

- 只能对程序描述的物理文件和源物理文件执行文件操作。
- symlink() 函数只能用来从支持符号链接的另一个文件系统链接到“独立 ASP QSYS.LIB”中的对象。
- 对数据库物理文件不能使用 QjoStartJournal() 和 QjoEndJournal() API。

文档库服务文件系统（QDLS）

QDLS 文件系统支持文件夹结构。它提供对文档和文件夹的访问。

而且:

- 它支持 iSeries 服务器文件夹和文档库对象（DLO）。
- 它支持存储在流文件中的数据。

有关 QDLS 的更多信息，参见通过集成文件系统接口使用 QDLS。

通过集成文件系统接口使用 QDLS

可以通过集成文件系统接口使用 OS/400 文件服务器或集成文件系统命令、用户屏幕和 C 语言 API 来访问 QDLS 文件系统。在使用集成文件系统接口时，应知道下列注意事项和限制。

QDLS 文件系统中的集成文件系统和 HFS

不仅可以通过“文档库对象”（DLO）CL 命令，而且还可以通过称为 HFS 的分层文件系统提供的集成文件系统接口或 API 对 QDLS 文件系统中的对象执行操作。集成文件系统基于集成语言环境（ILE）程序模型，而 HFS 基于原来的 iSeries 服务器程序模型。

HFS API 允许您执行集成文件系统不支持的一些附加操作。特别是可以使用 HFS API 来访问和更改目录扩展属性（也称为*目录项属性*）。应知道使用 HFS API 的命名规则与使用集成文件系统接口时 API 的命名规则是不同的。

有关 HFS 的更多信息，参见“iSeries 信息中心”中的分层文件系统 API 主题。

QDLS 文件系统中的用户注册

使用 QDLS 中的对象时，必须注册到系统分发目录。

QDLS 文件系统中的区分大小写性

QDLS 将用于对象名中的小写英文字母 **a** 到 **z** 转换为大写。因此，只使用这些字符的对象名搜索不区分大小写。

QDLS 中，所有其它字符是区分大小写的。

有关更多详细信息，参见 “iSeries 信息中心” 中的文件夹和文档名称主题。

QDLS 文件系统中的路径名

- 路径名的每个部分只能由一个名称组成，如:

  ```
  /QDLS/FLR1/DOC1
  ```

  或一个名称加一个扩展名（如 DOS 扩展名）组成，如:

  ```
  /QDLS/FLR1/DOC1.TXT
  ```

- 每个部分中名称的长度最多为 8 个字符，扩展名（如果有的话）的长度最多为 3 个字符。路径名的最大长度为 82 个字符，假定绝对路径名以 /QDLS 开始。
- QDLS 中的目录层次结构的深度最多可以为 32 级。若包括 / 和 QDLS 作为头两级，则目录层次结构的深度最多可以为 34 级。
- 存储名称时，名称中的字符被转换到作业的代码页，除非已经在 QUSRSYS 库中创建了数据区 Q0DEC500。若此数据区存在，则存储名称时，名称中的字符被转换到代码页 500。此功能提供了与先前发行版的 QDLS 文件系统的行为的兼容性。若不能将名称转换到适当的代码页，则可能拒绝该名称。

  有关代码页的更多信息，参见 “iSeries 信息中心” 中的全球化主题。

QDLS 文件系统中的链接

不能在 QDLS 文件系统中创建或存储符号链接。

集成文件系统处理文件夹和文件夹中文档库对象之间的关系的方法与处理文件夹和文件夹中每个对象之间的一个链接的方法是等价的。因此，有可能从支持符号链接的文件系统链接到 QDLS 文件系统中的对象。

有关链接的描述，参见第 16 页的『链接』。

在 QDLS 文件系统中使用集成文件系统命令和屏幕

第 22 页的『使用 CL 命令执行操作』中列出的命令可以对 QDLS 文件系统执行操作，但下列情况除外:

- ADDLNK 命令只能用来从支持符号链接的文件系统链接*到* QDLS 中的对象。
- CHKIN 和 CHKOUT 命令受文件的支持，但不受目录的支持。
- APYJRNCHG、ENDJRN、SNDJRNE 和 STRJRN 命令不受支持。

相同的限制适用于第 21 页的『使用 iSeries 菜单和屏幕执行操作』中描述的用户屏幕。

在 QDLS 文件系统中使用集成文件系统 API

第 40 页的『使用 API 执行操作』中列示的 C 语言函数可以对 QDLS 文件系统执行操作，但下列函数除外:

- symlink() 函数只能用来从支持符号链接的文件系统链接到 QDLS 中的对象。
- 不支持下列函数:

  givedescriptor()

ioctl()

link()

QjoEndJournal()

QjoRetrieveJournalInformation()

QJORJIDI()

QJOSJRNE()

QjoStartJournal()

Qp0lGetPathFromFileID()

readlink()

takedescriptor()

光盘文件系统（QOPT）

QOPT 文件系统提供对存储在光盘介质上的流数据的访问。

而且:

- 它提供类似于 PC 操作系统（如 DOS 和 OS/2）的分层目录结构。
- 它最适合流文件的输入／输出。
- 它支持存储在流文件中的数据。

有关 QOPT 的更多信息，参见通过集成文件系统接口使用 QOPT。

通过集成文件系统接口使用 QOPT

可以通过集成文件系统接口使用 OS/400 文件服务器或集成文件系统命令、用户屏幕和 API 来访问 QOPT 文件系统。在使用集成文件系统接口时，应知道下列注意事项和限制。

有关更多详细信息，参见出版物 Optical Support  。

QOPT 文件系统中的集成文件系统和 HFS

可以通过称为 HFS 的分层文件系统提供的集成文件系统接口或 API 对 QOPT 文件系统中的对象执行操作。集成文件系统基于集成语言环境（ILE）程序模型，而 HFS 基于原来的 iSeries 服务器程序模型。

HFS API 允许您执行集成文件系统不支持的一些附加操作。特别是可以使用 HFS API 来访问和更改目录扩展属性（也称为*目录项属性*）或使用保持的光盘文件。应知道使用 HFS API 的命名规则与使用集成文件系统接口时 API 的命名规则是不同的。

有关 HFS API 的更多信息，参见 “iSeries 信息中心” 中的分层文件系统 API 主题或出版物 Optical Support  。

QOPT 文件系统中的区分大小写性

取决于光盘介质的格式，当在 QOPT 中创建文件或目录时，可以保持或不保持大小写。但是，无论光盘介质的格式如何，文件和目录搜索不区分大小写。

QOPT 文件系统中的路径名

- 路径名必须以斜杠（/）开始。路径由文件系统名、卷名、目录和子目录名以及文件名组成。例如:

  ```
  /QOPT/VOLUMENAME/DIRECTORYNAME/SUBDIRECTORYNAME/FILENAME
  ```

- 文件系统名 QOPT 是必需的。
- 卷和路径名长度随光盘介质格式而变化。
- 可以在路径名中只指定 /QOPT，或在路径名中包含一个或多个目录或子目录。目录和文件名允许除 X'00' 至 X'3F' 和 X'FF' 以外的任何字符。根据光盘介质格式，可能存在附加限制。
- 文件名是路径名中的最后一个元素。文件名长度受路径中目录名长度的限制。

有关 QOPT 文件系统中路径名规则的更多详细信息，参见出版物 Optical Support  中的 “Path Name Rules” 讨论。

QOPT 文件系统中的链接

QOPT 文件系统只支持到对象的一个链接。不能在 QOPT 中创建或存储符号链接。但是，可以通过使用符号链接从 “根”（/）或 QOpenSys 文件系统访问 QOPT 中的文件。

有关链接的描述，参见第 16 页的『链接』。

在 QOPT 文件系统中使用集成文件系统命令和屏幕

第 22 页的『使用 CL 命令执行操作』中列示的大多数命令可以对 QOPT 文件系统执行操作。但是，在 QOPT 文件系统中有几种例外情况。记住，在支持多线程的进程中使用这些 CL 命令可能不安全；取决于光盘介质格式，可能存在某些限制。相同的限制适用于第 21 页的『使用 iSeries 菜单和屏幕执行操作』中描述的用户屏幕。

QOPT 文件系统不支持下列集成文件系统命令:

- ADDLNK
- APYJRNCHG
- CHKIN
- CHKOUT
- ENDJRN
- SNDJRNE
- STRJRN
- WRKOBJOWN
- WRKOBJPGP

在 QOPT 文件系统中使用集成文件系统 API

第 40 页的『使用 API 执行操作』中列示的所有 C 语言 API 都可以对 “根”（/）文件系统以线程安全方式执行操作，但下列 API 除外:

- QjoEndJournal()
- QjoRetrieveJournalInformation()
- QJORJIDI()
- QJOSJRNE()
- QjoStartJournal()

NetWare 文件系统（QNetWare）

QNetWare 文件系统提供对运行 Novell NetWare 4.10 或 4.11 的本地或远程 Integrated xSeries Server for iSeries 上数据的访问，或对运行 Novell NetWare 3.12、4.10、4.11 或 5.0 的独立 PC 服务器的访问。

而且:

- 它提供对 NetWare 目录服务（NDS）对象的访问。
- 它支持存储在流文件中的数据。
- 它提供将 Netware 文件系统动态安装到本地名称空间的能力

注: 仅当系统上安装了 NetWare Enhanced Integration for iSeries 400（BOSS 选项 25）时，QNetWare 文件系统才可用。在安装之后的下一次 IPL 后，/QNetWare 目录及其子目录表现为集成文件系统目录结构的一部分。

有关 QNetWare 文件系统的更多信息，参见下列主题:

- 安装 NetWare 文件系统
- QNetWare 目录结构
- 通过集成文件系统接口使用 QNetWare

安装 NetWare 文件系统

可以将位于 Novell NetWare 服务器上的 NetWare 文件系统安装在“根”（/）、QOpenSys 和其它文件系统上，这将比安装在 /QNetWare 目录下更易访问和更好操作。安装 NetWare 文件系统也可以用来利用“添加安装的文件系统”（ADDMFS）命令上的选项，如将读写文件系统安装成只读文件系统。

可以通过使用 NDS 路径或通过以格式 SERVER/VOLUME:directory/directory 指定 NetWare 路径来安装 NetWare 文件系统。例如，要安装位于服务器 Dreyfuss 上卷 Nest 中的目录 doorway，将使用下列语法:

```
DREYFUSS/NEST:doorway
```

此路径语法与 NetWare MAP 命令语法非常类似。NDS 路径可用来指定 NetWare 卷的路径，但它们本身不能被安装。

QNetWare 目录结构

/QNetWare 目录结构表示多个相异的文件系统:

- 该结构以下列格式表示网络中的 Novell NetWare 服务器和卷:

  ```
  /QNetWare/SERVER.SVR/VOLUME
  ```

  扩展名 .SVR 用来表示 Novell NetWare 服务器。
- 当通过集成文件系统菜单、命令和 API 访问服务器下的卷时，自动将 NetWare 卷的根目录安装在 /QNetWare 之下的 VOLUME 目录上。
- QNetWare 以下列格式表示网络上的 NDS 树:

  ```
  /QNetWare/CORP_TREE.TRE/USA.C/ORG.O/ORG_UNIT.OU/SVR1_VOL.CN
  ```

  扩展名 .TRE、.C、.O、.OU 和 .CN 分别用来表示 NDS 树、国家、组织、组织单元和公共名称。若通过卷对象的 NDS 路径或卷对象的别名访问 Novell NetWare 卷，则还自动将该卷的根目录安装在 NDS 对象上。

通过集成文件系统接口使用 QNetWare

可以通过集成文件系统接口使用 OS/400 文件服务器或集成文件系统命令、用户屏幕和 API 来访问 QNetWare 文件系统。应知道下列注意事项、限制和相关性。

QNetWare 文件系统中的权限和所有权

Novell NetWare 服务器存储和管理 QNetWare 中的文件和目录。当使用命令和 API 来检索或设置所有者或用户的权限时，QNetWare 根据用户的名称将 NetWare 用户映射为 iSeries 服务器用户。若 NetWare 名称超过十个字符或对应的 iSeries 服务器用户不存在，则不映射权限。不能被映射的所有者自动映射到用户概要文件 QDFTOWN。可以使用 WRKAUT 和 CHGAUT 命令来显示和更改用户的权限。当将权限传送到服务器和从服务器传送回权限时，将这些权限映射为 iSeries 服务器权限。

QNetWare 文件系统中的审计

虽然 Novell NetWare 支持文件和目录的审计，但 QNetWare 文件系统不能更改这些对象的审计值。因此，CHGAUD 命令不受支持。

QNetWare 文件系统中的文件和目录

QNetWare 文件系统不保持在命令或 API 中输入文件或目录时的大小写。所有名称在传输到 NetWare 服务器时设置为大写。Novell NetWare 还支持多平台（如 DOS、OS/2、Apple Macintosh 和 NFS）的名称空间。QNetWare 文件系统只支持 DOS 名称空间。由于在所有 Novell NetWare 卷上需要 DOS 名称空间，所以，所有文件和目录将出现在 QNetWare 文件系统中。

QNetWare 文件系统中的 NDS 对象

QNetWare 文件系统支持以大写和小写显示 NDS 名。

QNetWare 文件系统中的链接

QNetWare 文件系统只支持到对象的一个链接。不能在 QNetWare 中创建或存储符号链接。但是，可以在“根”（/）目录或 QOpenSys 目录中创建指向 QNetWare 文件或目录的符号链接。

在 QNetWare 文件系统中使用集成文件系统命令和屏幕

第 22 页的『使用 CL 命令执行操作』中列示的命令可以对 QNetWare 文件系统执行操作，但下列命令除外:

ADDLINK
APYJRNCHG
CHGAUD
CHGPGP
CHKIN
CHKOUT
ENDJRN
SNDJRN
STRJRN
WRKOBJOWN
WRKOBJPGP

除了先前的命令之外，也不能对 NDS 对象、服务器或卷使用下列命令:

CHGOWN
CPYFRMSTMF
CPYTOSTMF

CRTDIR

在 QNetWare 文件系统中使用集成文件系统 API

第 40 页的『使用 API 执行操作』中列出的 C 语言函数可以对 QNetWare 文件系统执行操作，但下列 API 除外:

givedescriptor()

link()

| QjoEndJournal()

| QjoRetrieveJournalInformation()

| QJORJIDI()

| QJOSJRNE()

| QjoStartJournal()

readlink()

symlink()

takedescriptor()

除了先前的 API 之外，也不能对 NDS 对象、服务器或卷使用下列 API:

chmod()

chown()

create()

fchmod()

fchown()

fcntl()

ftruncate()

lseek()

mkdir()

read()

readv()

unmask()

write()

writev()

Windows NT 服务器文件系统（QNTC）

QNTC 文件系统提供对一些数据和对象的访问，这些数据和对象存储在运行 Windows NT 4.0 服务器或更高版本的本地或远程 Integrated xSeries Server for iSeries 或独立服务器上。它允许 iSeries 服务器应用程序使用与 Windows NT 客户机相同的数据。它将数据存储在流文件中。

QNTC 文件系统是基本 OS/400 操作系统的一部分。要使用 QNTC 文件系统，必须已安装 TCP/IP Connectivity Utilities iSeries 版（部件号: 5769-TC1 ）。不需要安装 iSeries 400 Integration with Windows NT Server（操作系统的选项 29）也能访问 /QNTC。

有关 QNTC 的更多信息，参见通过集成文件系统接口使用 QNTC。

通过集成文件系统接口使用 QNTC

通过使用 OS/400 文件服务器或集成文件系统命令、用户屏幕和 API，可以通过集成文件系统接口访问 QNTC 文件系统。应知道下列注意事项和限制。

QNTC 文件系统中的权限和所有权

QNTC 文件系统不支持文件或目录的所有权概念。尝试使用命令或 API 来更改存储在 QNTC 中的文件的所有权将失败。系统用户概要文件 QDFTOWN 拥有 QNTC 中的所有文件和目录。

从 Windows NT 服务器管理对 NT 服务器文件和目录的权限。QNTC 不支持 WRKAUT 和 CHGAUT 命令。

QNTC 文件系统中的区分大小写性

QNTC 文件系统保持输入对象名时的相同大小写形式，但不区分名称中的大小写。无论对象名中的字符是大写还是小写，对该名称的搜索结果是相同的。

QNTC 文件系统中的路径名

- 路径名必须以斜杠（/）开始并且长度不能超过 255 个字符。
- 路径名区分大小写。
- 路径由文件系统名、Windows NT 服务器名、共享名、目录和子目录名以及对象名组成。路径名具有下列格式:

  ```
  /QNTC/Servername/Sharename/Directory/ . . . /Object
  ```

  （QNTC 是路径名的必需部分。）
- 服务器名的长度最多为 15 个字符。它必须是路径的一部分。
- 共享名的长度最多为 12 个字符。
- 共享名之后路径名的每个部分的长度最多为 255 个字符。
- 在 QNTC 中，一般可使用 130 级的层次结构。若将路径名的所有部分作为层次结构级，则目录层次结构的深度可以为 132 级。
- 以 Unicode CCSID 的格式存储名称。
- 本地子网中每个可运行的 Windows NT 服务器将自动表现为 /QNTC 下的目录。使用“生成目录”（MKDIR）命令（参见第 22 页的表 2）或 mkdir() API（参见第 40 页的『使用 API 执行操作』）来添加本地子网之外的 Windows NT 服务器。

QNTC 文件系统中的链接

QNTC 文件系统只支持到对象的一个链接。不能在 QNTC 中创建或存储符号链接。可以从“根”（/）或 QOpenSys 文件系统使用符号链接来访问 QNTC 中的文件。

有关链接的描述，参见第 16 页的『链接』。

在 QNTC 文件系统中使用集成文件系统命令和屏幕

| 第 22 页的『使用 CL 命令执行操作』中列示的命令可以对 QNTC 文件系统执行操作，但下列命令除外:

ADDLNK

| APYJRNCHG

CHGOWN

CHGAUT

CHGPGP

CHKIN

CHKOUT

DSPAUT
ENDJRN
RST
SAV
SNDJRNE
STRJRN
WRKAUT
WRKOBJOWN
WRKOBJPGP

相同的限制适用于第 21 页的『使用 iSeries 菜单和屏幕执行操作』中描述的用户屏幕。

在 QNTC 文件系统中使用 MKDIR 命令

使用“生成目录”（MKDIR）命令将服务器目录添加到 /QNTC 目录。自动创建本地子网中所有可运行的 Windows NT 服务器。必须使用 MKDIR 命令或 mkdir() API 添加本地子网之外的那些 Windows NT 服务器。例如:

```
MKDIR '/QNTC/NTSRV1'
```

将把 NTSRV1 服务器添加到 QNTC 文件系统目录结构中，以允许访问该服务器上的文件和目录。

也可以通过使用 TCP/IP 地址将新的服务器添加到目录结构中。例如:

```
MKDIR '/QNTC/9.130.67.24'
```

将把该服务器添加到 QNTC 文件系统目录结构中。

注: 如果使用 mkdir() API 或 MKDIR CL 命令将目录添加到目录结构，则这些目录在经过 IPL 后将不再可见。必须在每次系统 IPL 之后重新发出 MKDIR 命令或 mkdir() API。

在 QNTC 文件系统中使用集成文件系统 API

第 40 页的『使用 API 执行操作』中列出的 C 语言函数可以对 QNTC 文件系统执行操作，但下列情况除外:

- chmod()、fchmod()、utime() 和 umask() 函数将对 QNTC 中的对象无效，但尝试使用它们不会导致错误。
- QNTC 文件系统不支持下列函数:
  - chown()
  - fchown()
  - givedescriptor()
  - link()
  - QjoEndJournal()
  - QjoRetrieveJournalInformation()
  - QJORJIDI()
  - QJOSJRNE()
  - QjoStartJournal()
  - Qp0lGetPathFromFileID()
  - readlink()
  - symlink()
  - takedescriptor()

OS/400 文件服务器文件系统（QFileSvr.400）

OS/400 文件服务器文件系统提供对驻留在远程 iSeries 服务器上的其它文件系统的透明访问。可以通过分层目录结构来访问它。

可以认为 QFileSvr.400 文件系统是一个为用户执行文件请求的客户机。QFileSvr.400 与目标系统上的 OS/400 文件服务器进行交互来执行实际的文件操作。

有关 QFileSvr.400 的更多信息，参见通过集成文件系统接口使用 QFileSvr.400。

通过集成文件系统接口使用 QFileSvr.400

可以通过集成文件系统接口使用 OS/400 文件服务器或集成文件系统命令、用户屏幕和 API 来访问 QFileSvr.400 文件系统。在使用集成文件系统接口时，应知道下列注意事项和限制。

注: QFileSvr.400 文件系统的特征由目标服务器上正在被访问的文件系统的特征决定。

OS/400 文件服务器文件系统中的区分大小写性

对于第一级目录（它实际上表示目标系统的“根”（/）目录），QFileSvr.400 文件系统保持输入对象名时的相同大小写形式。但是，当 QFileSvr.400 搜索名称时，大写和小写之间没有区别。

对于所有其它目录，区分大小写性取决于正在被访问的特定文件系统。将文件请求发送到 OS/400 文件服务器时，QFileSvr.400 保持输入对象名时的相同大小写形式。

OS/400 文件服务器文件系统中的路径名

- 路径名具有下列格式:

  ```
  /QFileSvr.400/RemoteLocationName/Directory/Directory . . . /Object
  ```

  第一级目录（即上述示例中的 RemoteLocationName）表示下列两项:
  - 将用来建立通信连接的目标服务器的名称。目标服务器名称可以为下列名称之一:
    - TCP/IP 主机名（例如 beowulf.newyork.corp.com）
    - SNA LU 6.2 名（例如 appn.newyork）。
  - 目标服务器的“根”（/）目录

  因此，当使用集成文件系统接口创建第一级目录时，忽略任何指定的属性。

  注: 第一级目录在经过 IPL 后不会保持。即在每次 IPL 后必须再次创建第一级目录。
- 路径名的每个部分的长度最多为 255 个字符。全路径名的长度最多为 16 MB。

  注: 对象所在的文件系统可能将部分长度和路径名长度局限于小于 QFileSvr.400 所允许的最大值。
- 除程序和系统空间限制以及正在被访问的文件系统所强加的任何限制之外，目录层次结构的深度没有限制。
- 存储名称时名称中的字符被转换为 UCS2 Level 1 格式（参见第 20 页的『名称连续性』）。

OS/400 文件服务器文件系统中的通信

- 仅当目标服务器上的 QSERVER 子系统是活动的时候，才能与目标服务器上的文件服务器建立 TCP 连接。
- 仅当有未在使用的本地控制的会话（例如为了供 LU 6.2 连接使用而专门建立的会话）时，才尝试 SNA LU 6.2 连接。QFileSvr.400 文件系统在建立 LU 6.2 连接时，使用 BLANK 方式。在目标系统上，名为

QPWFSERV 的作业被提交到 QSERVER 子系统。此作业的用户概要文件由 BLANK 方式的通信项定义。

有关 LU 6.2 通信的更多信息，参见出版物 APPC Programming  。

- 使用 TCP 作为通信协议的文件服务器请求在正在发出该请求的作业的上下文中执行。使用 SNA 作为通信协议的文件服务器请求由 OS/400 系统作业 Q400FILSVR 执行。
- 若尚未建立与目标服务器的连接，则 QFileSvr.400 文件系统假定第一级目录表示 TCP/IP 主机名。QFileSvr.400 文件系统通过下列步骤来建立与目标服务器的连接:
  1. 将远程位置名转换为 IP 地址。
  2. 使用转换后的 IP 地址连接到已知端口 449 上主机服务器的服务器映射程序。然后向服务器映射程序发送一个查询，以获取服务名 “as-file”。由于查询的结果，发生下列情况之一:
     - 若 “as-file” 在目标服务器上的服务表中，则服务器映射器返回 OS/400 文件服务器守护程序收听所在的端口。
     - 若服务器映射器在目标服务器上是不活动的，则使用 “as-file” 的缺省端口号（8473）。

     然后，QFileSvr.400 文件服务器尝试与目标服务器上的 OS/400 文件服务器守护程序建立 TCP 连接。建立连接时，QFileSvr.400 与文件服务器交换请求和答复。在 QSERVER 子系统中，QPWFSERVSO 预启动请求以控制连接。每个预启动作业在自己的用户概要文件下运行。
  3. 若未将远程位置名转换为 IP 地址，则假定第一级目录是一个 SNA LU 6.2 名。因此，会尝试与 OS/400 文件服务器建立 APPC 连接。
- QFileSvr.400 文件系统定期（每 2 个小时）进行检查，以确定是否有任何未在使用的连接（例如，没有任何打开的文件与连接相关联）以及那些连接是否在两小时内没有活动。若找到这样的连接，则连接结束。
- QFileSvr.400 文件系统不能检测循环。下列路径名是一个循环示例:

  ```
  /QFileSvr.400/Remote2/QFileSvr.400/Remote1/QFileSvr.400/Remote2/...
  ```

  其中 Remote1 是本地系统。当指定了包含循环的路径名时，QFileSvr.400 文件系统会在短暂时间段之后返回一个错误。该错误指示发生了超时。

  QFileSvr.400 文件系统在基于 SNA 进行通信时将使用现有的空闲会话。必须启动该方式并建立一个会话，QFileSvr.400 才能成功地连接到远程通信系统。

OS/400 文件服务器文件系统中的安全性和对象权限

| 如果两个系统都已配置 Kerberos，且用户已对 Kerberos 认证，则 Kerberos 可以用来认证驻留在目标 iSeries 服务器上的文件系统。如果 Kerberos 认证失败，则用户标识和密码可以用来验证访问权。

| **注**: 如果在目标服务器已验证您的访问权之后凭单授权凭单或服务器凭单到期，则在与目标服务器的连接结束之前，到期将不生效。有关 Kerberos 的更多信息，参见 “iSeries 信息中心” 中的网络认证服务主题。

- 要访问驻留在目标 iSeries 服务器上的文件系统，如果不使用 Kerberos 进行认证，则您必须在目标服务器上具有与本地服务器上的用户标识和密码匹配的用户标识和密码。

  注: 若在目标服务器验证您的访问权之后更改您在本地或目标服务器上的密码，则在与目标服务器的连接结束之前该更改不会生效。但是，若删除您在本地服务器上的用户概要文件，并用相同的用户标识创建另一个用户概要文件，则不会有任何延迟。在这种情况下，QFileSvr.400 文件系统验证您是否具有对目标服务器的访问权。
- 对象权限基于驻留在目标服务器上的用户概要文件。即，仅当您在目标服务器上的用户概要文件对目标服务器上文件系统中的对象具有适当的权限时，才允许您访问该对象。

OS/400 文件服务器文件系统中的链接

QFileSvr.400 文件系统只支持到对象的一个链接。不能在 QFileSvr.400 中创建或存储符号链接。但是，可以通过使用符号链接从“根”（/）、QOpenSys 或用户定义文件系统访问 QFileSvr.400 中的文件。

有关链接的描述，参见第 16 页的『链接』。

在 OS/400 文件服务器 文件系统中使用集成文件系统命令和屏幕

第 22 页的『使用 CL 命令执行操作』中列出的命令可以对 QFileSvr.400 文件系统执行操作，但下列命令除外:

ADDLNK
APYJRNCHG
CHGAUT
CHGOWN
DSPAUT
ENDJRN
RST
SAV
SNDJRNE
STRJRN
WRKOBJOWN
WRKOBJPGP

相同的限制适用于第 21 页的『使用 iSeries 菜单和屏幕执行操作』中描述的用户屏幕。

在 OS/400 文件服务器 文件系统中使用集成文件系统 API

第 40 页的『使用 API 执行操作』中列示的 C 语言函数可以对 QFileSvr.400 文件系统执行操作，但下列函数除外:

chown()
fchown()
givedescriptor()
link()
QjoEndJournal()
QjoRetrieveJournalInformation()
QJORJIDI()
QJOSJRNE
QjoStartJournal
Qp0lGetPathFromFileID()
symlink()
takedescriptor()

网络文件系统（NFS）

NFS 文件系统允许用户访问存储在远程 NFS 服务器上的数据和对象。NFS 服务器可以导出一个网络文件系统，然后 NFS 客户机将动态安装该网络文件系统。

而且，通过“网络文件系统”安装在本地的任何文件系统还将具有从远程服务器安装的目录或文件系统的功能、特征、限制和相关性。对安装的文件系统的操作将不在本地执行。请求通过连接流向服务器，并且必须遵守服务器上文件系统类型的需求和限制。

有关 NFS 的更多信息，参见通过集成文件系统接口使用 NFS 文件系统。

通过集成文件系统接口使用 NFS 文件系统

“网络文件系统”可以通过集成文件系统接口进行访问，它具有下列注意事项和限制。

网络文件系统的特征

通过 NFS 安装的任何文件系统的特征取决于从服务器安装的文件系统的类型。必须认识到，对好像是本地目录或文件系统执行的请求实际上是通过 NFS 连接正在对服务器进行操作。

此客户机／服务器关系可能令人产生混淆。例如，考虑在客户机的“根”（/）目录分支的顶部从服务器安装了 QDLS 文件系统。虽然安装的文件系统好像是本地目录的扩展，但实际上起 QDLS 文件系统的作用。

认识到通过 NFS 安装的文件系统的此关系，对于在本地通过服务器连接来处理请求是很重要的。因为命令在本地级别上正确处理并不意味着它将在从服务器安装的目录中起作用。安装在客户机上的每个目录将具有服务器文件系统的特性和特征。

网络文件系统中客户机和服务器的变化

客户机／服务器连接有三种主要的可能性，它们可以影响“网络文件系统”将如何起作用以及它将具有哪些特征:

1. 用户在客户机上从 iSeries 服务器安装文件系统。
2. 用户在客户机上从 UNIX 服务器安装文件系统。
3. 用户在客户机上从非 iSeries 和非 UNIX 服务器安装文件系统。

在第一种方案中，安装的文件系统在客户机上的行为将与它在 iSeries 服务器上的行为相似。但是，需要考虑“网络文件系统”的特征和将安装的文件系统。例如，如果从服务器将 QDLS 文件系统安装到客户机，则它将具有 QDLS 文件系统的特征和限制。例如，在 QDLS 文件系统中，将路径名部分限制为 8 个字符加 3 个字符扩展名。但是，安装的文件系统也将具有 NFS 特征和限制。例如，不能使用 CHGAUD 命令来更改 NFS 对象的审计值。

在第二种方案中，必须认识到，从 UNIX 服务器安装的任何文件系统的行为将基本上与 iSeries QOpenSys 文件系统类似。要更详细了解 QOpenSys 文件系统，请参见第 55 页的『开放系统文件系统（QOpenSys）』。

在第三种方案中，您将需要检查与服务器操作系统相关联的文件系统的文档。

网络文件系统中的链接

一般来说，“网络文件系统”中允许到同一对象的多个硬链接。完全支持符号链接。符号链接可用来从“网络文件系统”链接到另一个文件系统中的对象。 多个硬链接和符号链接的能力完全取决于正在通过 NFS 安装的文件系统。

有关链接的描述，参见第 16 页的『链接』。

在网络文件系统中使用集成文件系统命令

第 22 页的『使用 CL 命令执行操作』中列示的所有命令和第 21 页的『使用 iSeries 菜单和屏幕执行操作』中描述的所有屏幕都可以对“网络文件系统”执行操作，但下列命令和屏幕除外:

- APYJRNCHG

- CHGAUD
- CHGATR
- CHGAUT
- CHGOWN
- CHGPGP
- CHKIN
- CHKOUT
- ENDJRN
- SNDJRNE
- STRJRN

一般来说，有些 CL 命令是特定于“网络文件系统”和其它安装的文件系统。但是，在支持多线程的进程中使用这些命令可能不安全。下表描述了这些命令。有关具体与“网络文件系统”相关的命令和屏幕的完整描述，参见 OS/400 网络文件系统支持  。

表 7. 网络文件系统 CL 命令

命令	描述
ADDMFS	“添加安装的文件系统”将导出的远程服务器文件系统安装在本地客户机目录。
CHGNFSEXP	“更改网络文件系统导出”对导出到“网络文件系统”客户机的文件系统的导出表添加或除去目录树。
DSPMFSINF	“显示安装的文件系统信息”显示关于安装的文件系统的信息。
ENDNFSSVR	“结束网络文件系统服务器”结束服务器上一个或所有“网络文件系统”守护程序。
EXPORTFS	“导出网络文件系统”对导出到“网络文件系统”客户机的文件系统的导出表添加或除去目录树。
MOUNT	“安装文件系统”将导出的远程服务器文件系统安装在本地客户机目录。此命令是 ADDMFS 命令的别名。
RLSIFSLCK	“释放集成文件系统锁定”释放客户机或对象上保持的所有“网络文件系统”字节范围锁定。
RMVMFS	“除去安装的文件系统”将导出的远程服务器文件系统从本地客户机名称空间中除去。
STRNFSSVR	“启动网络文件系统服务器”启动服务器上一个或所有“网络文件系统”守护程序。
UNMOUNT	“卸装文件系统”将导出的远程服务器文件系统从本地客户机名称空间中除去。此命令是 RMVMFS 命令的别名。

注: 在可以在“网络文件系统”上使用任何命令之前，必须安装该“网络文件系统”。

在网络文件系统中使用集成文件系统 API

第 40 页的『使用 API 执行操作』中列示的所有 C 语言函数可以对网络文件系统执行操作，但下列函数除外:

- QjoEndJournal()
- QjoRetrieveJournalInformation()
- QJORJIDI()
- QJOSJRNE()
- QjoStartJournal()

有关具体与“网络文件系统”相关的 C 语言函数的完整描述，参见 OS/400 网络文件系统支持  。

注: 在可以在“网络文件系统”上使用任何 API 之前，必须安装该“网络文件系统”。

第 7 章 集成文件系统对象的日志记录支持

日志记录的主要目的是使您能够恢复对象自上次保存以来发生的更改。

此信息将提供对日志管理的简要概述，并提供集成文件系统对象日志记录的注意事项和集成文件系统对象的日志记录支持的描述。

下列主题介绍集成文件系统对象的日志记录支持:

- 『日志管理』
- 『应该记入日志的对象』
- 第 80 页的『已记入日志的集成文件系统对象』
- 第 80 页的『已记入日志的操作』
- 第 81 页的『日志项的特殊注意事项』

| 有关集成文件系统对象日志记录的详细信息，参考“iSeries 信息中心”中的日志管理主题。

日志管理

日志管理的主要目的是使您能够恢复对象自上次保存以来发生的更改。也可以将日志管理用于:

- 对系统上的对象发生的活动审计跟踪
- 记录对对象发生的活动，您不能将这些对象记入日志
- 当从“活动时保存”介质恢复时可更快恢复
- 测试应用程序程序时进行辅助

可以使用日志来定义要用日志管理保护的对象。有关对象日志记录的更多注意事项，参见『应该记入日志的对象』。在该集成文件系统中，可以将流文件、目录和符号链接记入日志。仅支持“根”（/）、QOpenSys 和 UDFS 文件系统中的对象。

应该记入日志的对象

当决定是否应该将集成文件系统对象记入日志时，考虑下列问题:

- 对象更改的程度？在保存操作之间具有较大更改的对象是日志记录的好的候选者。
- 重新构造对对象所做的更改有多困难？对对象进行了很多更改而没有已写的记录吗？例如，重新构造用于电话订单项的对象的比重新构造用于以订单格式在邮件中送达的订单的对象更难。
- 对象中的信息有多关键？如果必须将对象恢复回上次保存操作的状态，重新构造更改中的延迟会对商业产生什么影响？
- 对象与服务器上其它对象的关系如何？尽管特定对象中的数据可能不会经常更改，但该对象的数据对服务器上的其它更动态的对象可能是关键的。例如，许多对象取决于客户主文件。如果要重新构造订单，客户主文件必须包括新客户或者信任限制自上次保存以来已发生的更改。

已记入日志的集成文件系统对象

可以使用 OS/400 日志记录支持来将某些集成文件系统对象类型记入日志。受支持的对象类型是流文件、目录和符号链接。“根”（/）、QOpenSys 和 UDFS 是支持这些对象类型的日志记录的唯一文件系统。通过使用传统系统界面（CL 命令或 API）或使用“iSeries 导航器”，可以将集成文件系统对象记入日志。可以通过“iSeries 导航器”启动日志记录和结束日志记录，并显示日志记录信息。

注: 不能将内存映射流文件和由 Integrated xSeries Server for iSeries（IXS）用于虚拟驱动器存储空间的流文件记入日志。

以下是概述集成文件系统中的日志记录支持的列表:

- 可以使用一般命令和 API 对受支持的对象类型执行日志操作。这些界面通常接受格式为路径名或 / 和文件标识的对象标识。
- 可以对集成文件系统对象的整个子树上执行某些日志操作命令，包括“启动日志记录”、“结束日志记录”和“应用记入日志的更改”。您可以选择使用可以对对象名称使用通配符模式的包含和排除列表。例如，可以使用“启动日志记录”命令来指定在树“/MyCompany”中与模式“*.data”匹配的所有对象上开始，但排除与模式“A*.data”和“B*.data”匹配的任何对象。
- 目录的日志记录支持包括目录操作，如添加链接，除去链接、创建对象、重命名对象和在目录内移动对象。

  记入日志的目录支持这样一种属性，可以设置该属性导致子树中的新对象继承目录的当前日志记录状态。当对记入日志的目录打开此属性时，创建或链接到目录的所有流文件、目录和符号链接（通过添加硬链接或通过重命名或除去对象）将自动由系统启动日志记录。

  注: 如果结束一个对象的日志记录，然后在它当前驻留的相同目录中重命名该对象，则不会对该对象启动日志记录，即使目录已将继承当前日志记录状态属性打开。
- 对象名称和完整的路径名称包含在集成文件系统对象的几个日志项内。对象名称和路径名称启用了“本地语言支持”（NLS）。
- 如果系统异常结束，为记入日志的集成文件系统对象提供了系统初始程序装入（IPL）恢复。
- 由 write() 和 writev() API 支持的最大写限制是 2 G 字节-1。如果指定了 RCVSIZOPT (*MAXOPT2)，则最大日志项大小是 4,000,000,000。否则，最大日志项大小是 15, 761,440 字节。如果您将流文件记入日志，并具有超过 15,761,440 字节的任何写入，您将要使用 *MAXOPT2 支持来防止发生任何错误。

有关集成文件系统对象日志记录的更详细的信息，参考“iSeries 信息中心”中的日志管理。

有关各种日志项布局的更多信息，参见在成员 QSYSINC/H（QP0LJRNL）中交付的 C 语言包含文件 qp0ljrnl.h，它包含集成文件系统日志项特定的数据内容和格式的详细信息。

要获取为集成文件系统对象存入的所有日志项的完整列表，参考“iSeries 信息中心”中的日志代码查找程序。

已记入日志的操作

仅当操作所使用的对象或链接的类型也是可记入日志的类型时，才将以下操作记入日志:

- 创建对象。
- 将链接添加到现有对象。
- 解链链接。
- 重命名链接。
- 重命名文件标识符。

- 将链接移进或移出目录。

以下记入日志的操作是特定于流文件的:

- 数据写
- 文件截断 / 扩展
- 强制的文件数据
- 保存并释放存储器

下列记入日志的操作适用于所有记入日志的对象类型:

- 属性更改（包括安全性更改，例如，权限和所有权）
- 打开
- 关闭
- 启动日志记录
- 结束日志记录
- 启动“应用记入日志的更改”（APYJRNCHG）命令
- 结束“应用记入日志的更改”（APYJRNCHG）命令
- 保存
- 恢复

有关集成文件系统对象日志记录的详细信息，参考“iSeries 信息中心”中的日志管理主题。要获取为集成文件系统对象存入的所有日志项的完整列表，参考**系统管理**主题中的日志代码查找程序。

日志项的特殊注意事项

| 许多记入日志的集成文件系统操作在内部使用提交控制来从操作期间执行的多个功能形成单个事务。除非提交
| 控制周期具有“提交”日志项（日志代码为 C，类型为 CM），否则完全不应该考虑这些记入日志的操作。在
| 提交控制周期中包含“回滚”日志项（日志代码为 C，类型为 RB）的记入日志的操作是失败的操作，不应该
| 回放或复制它们中的日志项。

以此方式使用提交控制的记入日志的集成文件系统项（日志代码为 B）包括:

- AA — 更改审计值
- B0 — 开始创建
- B1 — 创建摘要
- B2 — 添加链接
- B3 — 重命名 / 移动
- B4 — 解除链接（父目录）
- B5 — 解除链接（链接）
- FA — 属性更改
- JT — 启动日志（仅当因为“继承日志记录”属性为“是”的目录中的操作而启动日志时）
- OA — 权限更改
- OG — 对象主组更改
- OO — 对象所有者更改

几个集成文件系统日志项具有一个特定数据字段，指示该项是否是摘要项。发送摘要项类型的操作将发送两个相同项类型到日志。第一个项包含项特定数据的子集。第二个项包含完整的项特定数据，并将指示它是否是摘要项。正在复制对象和／或回放操作的程序通常将仅对摘要项有兴趣。

对于记入日志的目录中的创建操作，B1 日志项（创建摘要）被认为是摘要项。

某些记入日志的操作需要发送与该操作有逆向关系的日志项。例如，包含 B4 日志项（解除链接）的提交控制周期也可能包含 B2 日志项（添加链接）。仅在导致“回滚”日志项（C — RB）的操作中，才将发生此类型情况。

因为以下两种原因而可能发生此情况:

1. 操作即将失败，而在内部需要项以清除错误路径。
2. 操作由于系统停机而中断，在随后的 IPL 期间，执行了发送项所需要的恢复来回滚中断的操作。

附录 A. 独立于传送的远程过程调用

由 Sun Microsystems 开发的“远程过程调用”（RPC）便于将客户机应用程序从服务器机制中分离出来并进行分发。它包括一个数据表示标准，称为“外部数据表示”或 XDR 以允许多种类型的机器访问传送数据。“独立于传送的 RPC”（TI-RPC）是 RPC 的最新版本。它提供了一种隔离在网络层使用的基础协议的方法，从而允许更加无缝地从一种协议转换到另一种协议。iSeries 服务器上当前唯一可用的协议是 TCP 和 UDP。

在使用 RPC 时，在网络上开发分布式应用程序是一个无缝任务。主要目标是更倾向于分发用户界面或数据检索的应用程序。

网络选择

下列 API 提供了选择应用程序应运行的传送的一些方式。

这些 API 要求 *STMF /etc/netconfig 文件在系统上存在。若 /etc 目录中没有 netconfig 文件，则用户必须从 /QIBM/ProdData/OS400/RPC 目录复制该文件。netconfig 文件总是在 /QIBM/ProdData/OS400/RPC 目录中。

API	描述
endnetconfig()	释放指向存储在 netconfig 文件中的记录的指针
freenetconfigent()	释放从 getnetconfigent() 函数调用返回的 netconfig 结构。
getnetconfig()	返回指向 netconfig 文件中的当前记录的指针，并将它的指针加 1 以指向下一条记录。
getnetconfigent()	返回指向与输入网络标识对应的 netconfig 结构的指针。
setnetconfig()	初始化指向 netconfig 文件中的第一项的记录指针。在第一次使用 getnetconfig() 函数之前，必须使用 setnetconfig() 函数。setnetconfig() 函数返回一个唯一的句柄（指向存储在 netconfig 文件中的记录的指针）以供 getnetconfig() 函数使用。

名称到地址的转换

下列 API 允许应用程序以一种独立于传送的方式获得服务或指定主机的地址。

API	描述
netdir_free()	释放由名称到地址转换 API 所分配的结构
netdir_getbyaddr()	将地址映射到主机名和服务名
netdir_getbyname()	将服务参数中指定的主机名和服务名映射到一组与 netconfig 结构中标识的传送一致的地址
netdir_options()	提供与特定于传送的能力（如广播地址及 TCP 和 UDP 的保留端口设施）的接口
netdir_sperror()	发出一条信息性消息，说明一个名称到地址转换 API 失败的原因
taddr2uaddr()	将特定于传送（本地）的地址转换为独立于传送（通用）的地址
uaddr2taddr()	将独立于传送（通用）的地址转换为特定于传送（本地）的地址（netbuf 结构）

外部数据表示（XDR）

下列 API 允许“远程过程调用”（RPC）应用程序处理任意的数据结构，而不管它们的主机字节次序或结构布局约定是否相同。

API	描述
xdr_array()	一个过滤器原语，在可变长度数组和其对应的外部表示之间进行转换。调用此函数对数组的每个元素进行编码或解码。
xdr_bool()	一个过滤器原语，在布尔数（C 语言整数）和其外部表示之间进行转换。在对数据进行编码时，此过滤器生成值 1 或 0。
xdr_bytes()	一个过滤器原语，在计数字节数组和其外部表示之间进行转换。此函数处理其中数组元素大小已知为 1 并且每个元素的外部描述是内部的类属数组的子集。字节序列的长度显式地放在一个无符号整数中。字节序列不以空字符结束。字节的外部表示与其内部表示相同。
xdr_char()	一个过滤器原语，在 C 语言字符和其外部表示之间进行转换。
xdr_double()	一个过滤器原语，在 C 语言双精度数和其外部表示之间进行转换。
xdr_double_char()	一个过滤器原语，在 C 语言双字节字符和其外部表示之间进行转换。
xdr_enum()	一个过滤器原语，在 C 语言枚举（enum）和其外部表示之间进行转换。
xdr_free()	递归释放传入的指针所指向的对象
xdr_float()	一个过滤器原语，在 C 语言浮点数（规格化单精度浮点数）和其外部表示之间进行转换。
xdr_int()	一个过滤器原语，在 C 语言整数和其外部表示之间进行转换。
xdr_long()	一个过滤器原语，在 C 语言长整数和其外部表示之间进行转换。
xdr_netobj()	一个过滤器原语，在可变长度不透明数据和其外部表示之间进行转换。
xdr_opaque()	一个过滤器原语，在固定大小不透明数据和其外部表示之间进行转换。
xdr_pointer()	提供在结构内部进行跟踪的指针并序列化空指针。可以表示递归数据结构，如二进制树或链接的列表。
xdr_reference()	一个服务器原语，它提供在结构内部进行跟踪的指针。此原语允许序列化、串并转换和释放一种结构中由另一种结构引用的任何指针。 xdr_reference() 函数不会在序列化期间将特殊意义附加到空指针上，传送空指针的地址可能引起内存错误。因此，程序员必须用一个两面判别的联合来描述数据。指针有效时使用一面；指针为空时使用另一面。
xdr_short()	一个过滤器原语，在 C 语言短整数和其外部表示之间进行转换。
xdr_string()	一个过滤器原语，在 C 语言字符串和其对应的外部表示之间进行转换。
xdr_u_char()	一个过滤器原语，在无符号的 C 语言字符和其外部表示之间进行转换。
xdr_u_int()	一个过滤器原语，在 C 语言无符号整数和其外部表示之间进行转换。
xdr_u_long()	一个过滤器原语，在 C 语言无符号长整数和其外部表示之间进行转换。
xdr_u_short()	一个过滤器原语，在 C 语言无符号短整数和其外部表示之间进行转换。
xdr_union()	一个过滤器原语，在判别型 C 语言联合和其对应的外部表示之间进行转换。
xdr_vector()	一个过滤器原语，在固定长度数组和其对应的外部表示之间进行转换。
xdr_void()	没有参数。它被传送到其它需要参数的 RPC 函数，但不传送数据。
xdr_wrapstring()	一个调用 xdr_string(xdr, sp, maxuint) API 的原语，其中 maxuint 是无符号整数的最大值。xdr_wrapstring() 是有用的，因为 RPC 软件包将两个 XDR 函数的最大值作为参数传送，而 xdr_string() 函数需要三个参数。

认证

下列 API 提供了对“独立于传送的远程过程调用”（TI-RPC）应用程序的认证。

API	描述
auth_destroy()	消除 auth 参数所指向的认证信息结构
authnone_create()	创建并返回一个缺省 RPC 认证句柄，该句柄通过每个远程过程调用来传送空认证信息。
authsys_create()	创建并返回一个包含认证信息的 RPC 认证句柄

独立于传送的 RPC（TI-RPC）

下列 API 通过将应用程序与任何特定的传送功能部件隔离来提供一个分布式应用程序开发环境。这使得传送功能便于使用。

TI-RPC 简化 API

下列简化 API 指定要使用的传送类型。使用此级别 API 的应用程序不必显式创建句柄。

API	描述
rpc_call()	调用指定系统上的远程过程
rpc_reg()	向 RPC 服务包注册过程

TI-RPC 顶级 API

下列 API 允许应用程序指定传送类型。

API	描述
clnt_call()	调用与客户机相关联的远程过程
clnt_control()	更改关于客户机对象的信息
clnt_create()	创建类属客户机句柄 clnt_destroy() 来消除客户机的 RPC 句柄
clnt_destroy()	消除客户机的 RPC 句柄
svc_create()	创建服务器句柄
svc_destroy()	消除 RPC 服务传送句柄

TI-RPC 中级 API

下列 API 与顶级 API 类似，但用户应用程序使用网络选择 API 来选择特定于传送的信息:

API	描述
clnt_tp_create()	创建客户机句柄
svc_tp_create()	创建服务器句柄

TI-RPC 专家级 API

下列 API 允许应用程序选择要使用的传送。它们还对 CLIENT 和 SVCXPRT 句柄的详细信息提供增强的控制级别。这些 API 与中级 API 类似，但通过使用名称到地址转换 API 提供了附加控制。

通过使用名称到地址转换 API 提供的附加控制。

API	描述
clnt_tli_create()	创建客户机句柄
rpcb_getaddr()	查找服务的通用地址
rpcb_set()	向 RPCbind 注册服务器地址
rpcb_unset()	由服务器用来取消注册其地址
svc_reg()	使程序和版本与调遣相关联
svc_tli_create()	创建服务器句柄
svc_unreg()	删除由 svc_reg() 设置的关联

其它 TI-RPC API

下列 API 允许各种应用程序协调使用简化、顶级、中级和专家级 API。

API	描述
clnt_freeres()	释放由 RPC 或 XDR 系统分配的数据
clnt_geterr()	从客户机句柄获取错误结构
svc_freeargs()	释放由 RPC 或 XDR 系统分配的数据
svc_getargs()	解码 RPC 请求的自变量
svc_getrpccaller()	获取调用者的网络地址
svc_run()	等待 RPC 请求到达
svc_sendreply()	将过程调用结果发送到远程客户机。
svcerr_decode()	将信息发送到客户机以获取解码错误
svcerr_noproc()	将信息发送到客户机以获取过程号错误
svcerr_systemerr()	将信息发送到客户机以获取系统错误

附录 B. 使用集成文件系统 C 函数的示例程序

这个简单的 C 语言程序说明了如何使用几个集成文件系统函数。该程序执行下列操作:

1. 使用 **getuid()** 函数来确定实际的用户标识（UID）。
2. 使用 **getcwd()** 函数来确定当前目录。
3. 使用 **open()** 函数来创建文件。它为所有者（创建文件的人）建立对文件的读、写和执行权限。
4. 使用 **write()** 函数将字节串写入到文件。 打开操作（ 3 ）中提供的文件描述符标识该文件。
5. 使用 **close()** 函数来关闭文件。
6. 使用 **mkdir()** 函数在当前目录中创建新的子目录。所有者获得对该子目录的读、写和执行访问权。
7. 使用 **chdir()** 函数将新的子目录更改为当前目录。
8. 使用 **link()** 函数来创建到先前创建的文件（ 3 ）的链接。
9. 使用 **open()** 函数来打开文件，以便只读取该文件。在（ 8 ）中创建的链接允许访问该文件。
10. 使用 **read()** 函数从文件中读取字节串。打开操作（ 9 ）中提供的文件描述符标识该文件。
11. 使用 **close()** 函数来关闭文件。
12. 使用 **unlink()** 函数来除去到文件的链接。
13. 使用 **chdir()** 函数将当前目录更改回在其中创建了该新子目录的父目录。
14. 使用 **rmdir()** 函数来除去先前创建的子目录（ 6 ）。
15. 使用 **unlink()** 函数来除去先前创建的文件（ 3 ）。

注: 此样本程序将在一些系统上正常运行，在这些系统中运行的作业的 CCSID 是 37。集成文件系统 API 必须以作业的 CCSID 格式将对象和路径名进行编码；但是，C 编译器以 CCSID 37 格式存储字符常量。为了完全兼容，在将 API 转到作业的 CCSID 之前，转换诸如对象和路径名的字符常量。

此不保证声明信息与代码示例有关。

```
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>


#define BUFFER_SIZE       2048
#define NEW_DIRECTORY     "testdir"
#define TEST_FILE         "test.file"
#define TEST_DATA         "Hello World!"
#define USER_ID           "user_id_"
#define PARENT_DIRECTORY  ".."


char  InitialFile[BUFFER_SIZE];
char  LinkName[BUFFER_SIZE];
char  InitialDirectory[BUFFER_SIZE] = ".";
char  Buffer[32];
int   FilDes = -1;
int   BytesRead;
int   BytesWritten;
uid_t UserID;
```

```
void CleanUpOnError(int level)
{
   printf("Error encountered, cleaning up.\n");
   switch ( level )
      {
       case 1:
           printf("Could not get current working directory.\n");
           break;
       case 2:
           printf("Could not create file %s.\n",TEST_FILE);
           break;
       case 3:
           printf("Could not write to file %s.\n",TEST_FILE);
           close(FilDes);
           unlink(TEST_FILE);
           break;
       case 4:
           printf("Could not close file %s.\n",TEST_FILE);
           close(FilDes);
           unlink(TEST_FILE);
           break;
       case 5:
           printf("Could not make directory %s.\n",NEW_DIRECTORY);
           unlink(TEST_FILE);
           break;
       case 6:
           printf("Could not change to directory %s.\n",NEW_DIRECTORY);
           rmdir(NEW_DIRECTORY);
           unlink(TEST_FILE);
           break;
       case 7:
           printf("Could not create link %s to %s.\n",LinkName,InitialFile);
           chdir(PARENT_DIRECTORY);
           rmdir(NEW_DIRECTORY);
           unlink(TEST_FILE);
           break;
       case 8:
           printf("Could not open link %s.\n",LinkName);
           unlink(LinkName);
           chdir(PARENT_DIRECTORY);
           rmdir(NEW_DIRECTORY);
           unlink(TEST_FILE);
           break;
       case 9:
           printf("Could not read link %s.\n",LinkName);
           close(FilDes);
           unlink(LinkName);
           chdir(PARENT_DIRECTORY);
           rmdir(NEW_DIRECTORY);
           unlink(TEST_FILE);
           break;
       case 10:
           printf("Could not close link %s.\n",LinkName);
           close(FilDes);
           unlink(LinkName);
           chdir(PARENT_DIRECTORY);
           rmdir(NEW_DIRECTORY);
           unlink(TEST_FILE);
           break;
       case 11:
           printf("Could not unlink link %s.\n",LinkName);
           unlink(LinkName);
           chdir(PARENT_DIRECTORY);
           rmdir(NEW_DIRECTORY);
           unlink(TEST_FILE);
           break;
```

```
        case 12:
            printf("Could not change to directory %s.\n",PARENT_DIRECTORY);
            chdir(PARENT_DIRECTORY);
            rmdir(NEW_DIRECTORY);
            unlink(TEST_FILE);
            break;
        case 13:
            printf("Could not remove directory %s.\n",NEW_DIRECTORY);
            rmdir(NEW_DIRECTORY);
            unlink(TEST_FILE);
            break;
        case 14:
            printf("Could not unlink file %s.\n",TEST_FILE);
            unlink(TEST_FILE);
            break;
        default:
            break;
       }
   printf("Program ended with Error.\n"\
          "All test files and directories may not have been removed.\n");
}

int main ()
{
 1
/* Get and print the real user id with the getuid() function. */
   UserID = getuid();
   printf("The real user id is %u. \n",UserID);

 2
/* Get the current working directory and store it in InitialDirectory. */
   if ( NULL == getcwd(InitialDirectory,BUFFER_SIZE) )
      {
      perror("getcwd Error");
      CleanUpOnError(1);
   return 0;
      }
   printf("The current working directory is %s. \n",InitialDirectory);

 3
/* Create the file TEST_FILE for writing, if it does not exist.
   Give the owner authority to read, write, and execute. */
   FilDes = open(TEST_FILE, O_WRONLY | O_CREAT | O_EXCL, S_IRWXU);
   if ( -1 == FilDes )
      {
      perror("open Error");
      CleanUpOnError(2);
   return 0;
      }
   printf("Created %s in directory %s.\n",TEST_FILE,InitialDirectory);

 4
/* Write TEST_DATA to TEST_FILE via FilDes */
   BytesWritten = write(FilDes,TEST_DATA,strlen(TEST_DATA));
   if ( -1 == BytesWritten )
      {
      perror("write Error");
      CleanUpOnError(3);
   return 0;
      }
   printf("Wrote %s to file %s.\n",TEST_DATA,TEST_FILE);

 5
/* Close TEST_FILE via FilDes */
   if ( -1 == close(FilDes) )
      {
      perror("close Error");
```

```
      CleanUpOnError(4);
   return 0;
      }
   FilDes = -1;
   printf("File %s closed.\n",TEST_FILE);

 6
/* Make a new directory in the current working directory and
   grant the owner read, write and execute authority */
   if ( -1 == mkdir(NEW_DIRECTORY, S_IRWXU) )
      {
      perror("mkdir Error");
      CleanUpOnError(5);
   return 0;
      }
   printf("Created directory %s in directory %s.\n",NEW_DIRECTORY,InitialDirectory);

 7
/* Change the current working directory to the
   directory NEW_DIRECTORY just created. */
   if ( -1 == chdir(NEW_DIRECTORY) )
      {
      perror("chdir Error");
      CleanUpOnError(6);
   return 0;
      }
   printf("Changed to directory %s/%s.\n",InitialDirectory,NEW_DIRECTORY);

/* Copy PARENT_DIRECTORY to InitialFile and
   append  "/" and TEST_FILE to InitialFile. */
   strcpy(InitialFile,PARENT_DIRECTORY);
   strcat(InitialFile,"/");
   strcat(InitialFile,TEST_FILE);

/* Copy USER_ID to LinkName then append the
   UserID as a string to LinkName. */
   strcpy(LinkName, USER_ID);
   sprintf(Buffer, "%d\0", (int)UserID);
   strcat(LinkName, Buffer);

 8
/* Create a link to the InitialFile name with the LinkName. */
   if ( -1 == link(InitialFile,LinkName) )
      {
      perror("link Error");
      CleanUpOnError(7);
   return 0;
      }
   printf("Created a link %s to %s.\n",LinkName,InitialFile);

 9
/* Open the LinkName file for reading only. */
   if ( -1 == (FilDes = open(LinkName,O_RDONLY)) )
      {
      perror("open Error");
      CleanUpOnError(8);
   return 0;
      }
   printf("Opened %s for reading.\n",LinkName);

 10
/* Read from the LinkName file, via FilDes, into Buffer. */
   BytesRead = read(FilDes,Buffer,sizeof(Buffer));
   if ( -1 == BytesRead )
      {
      perror("read Error");
      CleanUpOnError(9);
```

```
   return 0;
      }
   printf("Read %s from %s.\n",Buffer,LinkName);
   if ( BytesRead != BytesWritten )
      {
      printf("WARNING: the number of bytes read is "\
             "not equal to the number of bytes written.\n");
      }

 11
/* Close the LinkName file via FilDes. */
   if ( -1 == close(FilDes) )
      {
      perror("close Error");
      CleanUpOnError(10);
   return 0;
      }
   FilDes = -1;
   printf("Closed %s.\n",LinkName);

 12
/* Unlink the LinkName link to InitialFile. */
   if ( -1 == unlink(LinkName) )
      {
      perror("unlink Error");
      CleanUpOnError(11);
   return 0;
      }
   printf("%s is unlinked.\n",LinkName);

 13
/* Change the current working directory
   back to the starting directory. */
   if ( -1 == chdir(PARENT_DIRECTORY) )
      {
      perror("chdir Error");
      CleanUpOnError(12);
   return 0;
      }
   printf("changing directory to %s.\n",InitialDirectory);

 14
/* Remove the directory NEW_DIRECTORY */
   if ( -1 == rmdir(NEW_DIRECTORY) )
      {
      perror("rmdir Error");
      CleanUpOnError(13);
   return 0;
      }
   printf("Removing directory %s.\n",NEW_DIRECTORY);

 15
/* Unlink the file TEST_FILE */
   if ( -1 == unlink(TEST_FILE) )
      {
      perror("unlink Error");
      CleanUpOnError(14);
   return 0;
      }
   printf("Unlinking file %s.\n",TEST_FILE);


   printf("Program completed successfully.\n");
   return 0;
}
```

附录 C. 集成文件系统 RPG 代码示例

代码片断  包含一个集成文件系统 RPG 代码示例。要查看此示例，执行下列步骤:

1. 从“搜索”类别的下拉列表中选择 **ILE RPG 源**。
2. 单击**搜索**。
3. 向下滚动列表，直至看到**从 RPG 使用 IFS** 为止。
4. 单击**用于从 RPG 使用 IFS 的代码**。

此不保证声明信息与代码示例有关。

文献目录

本文献书目列示了 iSeries 服务器信息，它包含本书中所讨论信息的背景信息或更多详细信息。

- “iSeries 信息中心”的**编程**类别中的控制语言主题提供了 iSeries 服务器控制语言（CL）及其命令的描述。每个命令描述都包括一个语法图、参数、缺省值、关键字和示例。
- “iSeries 信息中心”的全球化主题解释了诸如字符集和代码页的本地语言支持（NLS）概念，并提供了评估、计划和使用 iSeries 服务器 NLS 和多种语言能力所需要的信息。
- “iSeries 信息中心”的**编程**类别中的 API 主题提供了每个 OS/400 API 的描述，包括集成文件系统 API。
- “iSeries 信息中心”的**系统管理**类别中的日志管理主题提供了关于如何设置、管理和故障诊断 iSeries 服务器上的管理系统的访问路径保护（SMAPP）、本地日志和远程日志的信息。
- “iSeries 信息中心”类别的**数据库**类别中的提交控制主题解释了如何以逻辑工作单元的形式定义和处理对资源（如数据库文件或集成文件系统文件）的一组更改。
- OS/400 Network File System Support 此书通过一系列现实生活中的应用程序描述了“网络文件系统”。包括关于导出、安装、文件锁定和安全性注意事项的信息。可以从此书学习如何使用 NFS 来构造和开发安全网络名称空间。
- Optical Support 此书用作 OS/400 上“IBM 光盘支持”的用户指南和参考。此书中的信息可以帮助用户理解光盘库数据服务器的概念，规划光盘库，管理和运行光盘库数据服务器，以及解决光盘数据服务器问题。
- WebSphere Development Studio: ILE C/C++ Programmers Guide 此书提供了在 iSeries 服务器上设计、编辑、编译、运行和调试 ILE C/400 程序所需要的信息。
- WebSphere Development Studio: C/C++ Language Reference 此书提供了关于 ILE C/400 程序结构的信息，并包含关于库函数和包含（头）文件的详细信息。
- Security — Reference 此书提供了关于 OS/400 安全性的详细技术信息。
- APPC Programming 此书描述了 iSeries 服务器的高级程序间通信（APPC）支持。它可以在开发使用 APPC 的应用程序和定义 APPC 的通信环境方面进行指导。
- Backup and Recovery 此书提供了关于 IBM iSeries 服务器的恢复和可用性选项的一般信息。

文献书目

索引

[K]

[L]

[M]

[P]

[Q]

[R]

[Z]

A

C

F

I

N

O

P

Q

U

W

中国印刷