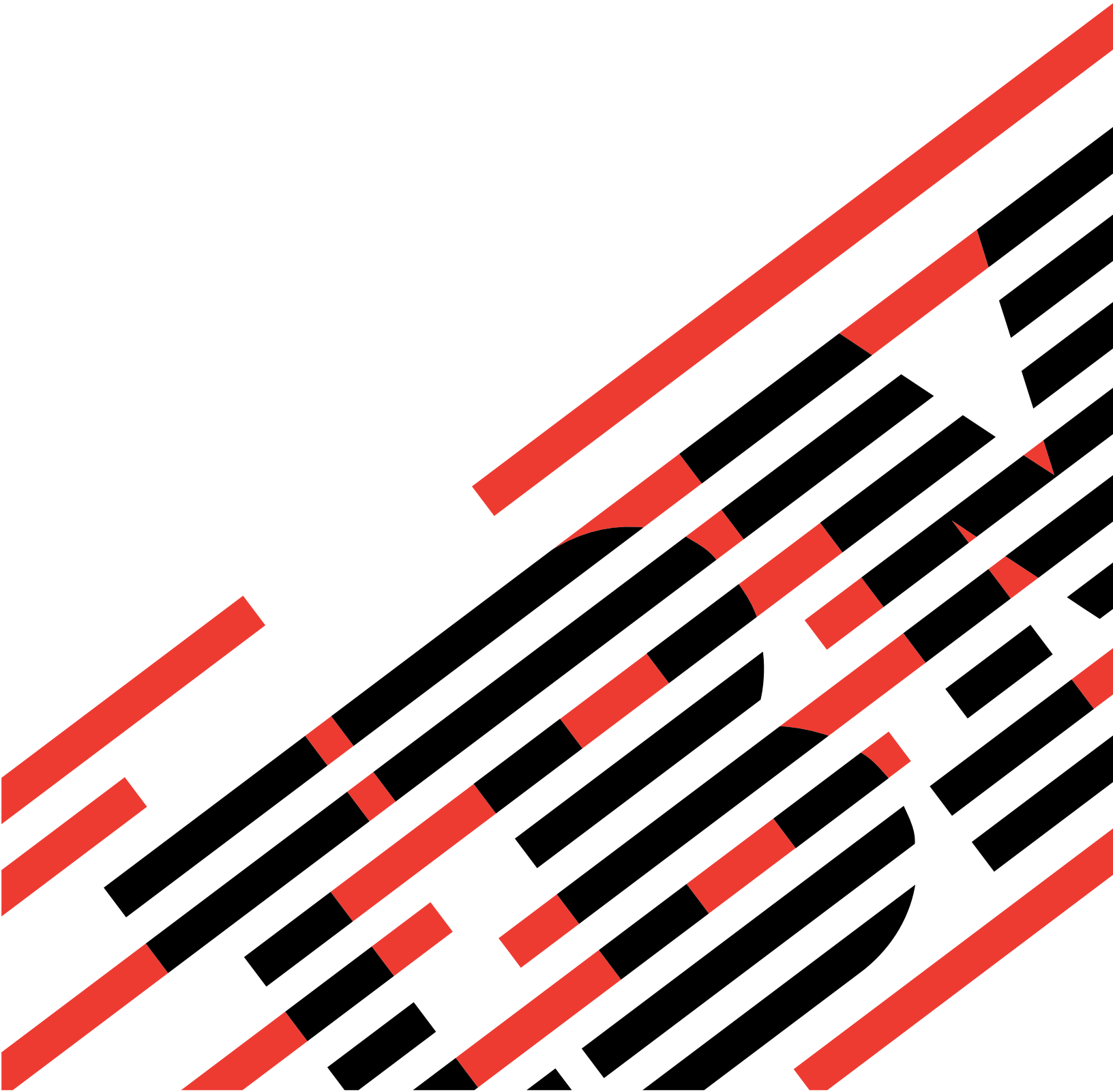
IBM

eserver

iSeries

統合ファイル・システムの紹介

バージョン 5



IBM

@server

iSeries

統合ファイル・システムの紹介

バージョン 5

目次

『統合ファイル・システムの紹介』について vii
『統合ファイル・システムの紹介』の対象読者 vii
コードの特記事項情報 vii

第 1 章 統合ファイル・システムの紹介 1
統合ファイル・システムとは 1
統合ファイル・システム使用の利点 1

第 2 章 統合ファイル・システムの概念 3
ストリーム・ファイル 3
*TYPE1 および *TYPE2 のストリーム・ファイル 4
統合ファイル・システムのファイル・システム 4
ディレクトリー 6
現行のディレクトリーとホーム・ディレクトリー 9
*TYPE2 ディレクトリー 9
OS/400 V5R1 での *TYPE2 ディレクトリーの使用 11
*TYPE2 ディレクトリーへの変換 11
「ルート」、QOpenSys、または UDFS が使用できない場合 12
補助記憶域要件 12
シンボリック・リンクに関する考慮事項 13
独立補助記憶域プール (ASP) 13
保管 / 復元に関する考慮事項 13
*TYPE2 変換のための準備 13
変換処理 15
例: すべてのファイル・システムの変換 (少数のオブジェクト) 16
例: すべてのファイル・システムの変換 (多数のオブジェクト) 17
例: 特定の ASP のみの変換 18
パス名 18
リンク 19
ハード・リンク 20
シンボリック・リンク 21
比較: ハード・リンクとシンボリック・リンク 23
拡張属性 23
名前の継続性 24

第 3 章 従来のシステム・インターフェースを使用した統合ファイル・システムへのアクセス 27
iSeries メニューおよび表示画面を使用した操作の実行 27
CL コマンドを使用した操作の実行 28
CL コマンドおよび表示画面のパス名規則 31
PC を使用した操作の実行 34
FTP を使用したファイルの転送 34
iSeries ネットサーバーを使用したファイルの処理 35
別のファイル・システムへのオブジェクトの移動 36
別のファイル・システムへオブジェクトを移動させる際の考慮事項 37
統合ファイル・システムで提供されるディレクトリー 38

第 4 章 iSeries ナビゲーターを使用した統合ファイル・システムへのアクセス 41
ファイルのチェックイン 41
ファイルのチェックアウト 42

ファイルまたはフォルダーに対する許可のセットアップ 42
ファイル・テキスト変換のセットアップ 42
他のシステムへのファイルまたはフォルダーの送信 43
パッケージ定義のオプションの変更 43
ファイルまたはフォルダーの送信日時のスケジュール 44
フォルダーの作成 44
フォルダーの除去 44
ファイル共用の作成 45
ファイル共用の変更 45
新規のユーザー定義ファイル・システムの作成 45
ユーザー定義ファイル・システムのマウント 46
ユーザー定義ファイル・システムのアンマウント 46
| ジャーナル処理の開始 47
| ジャーナル処理の終了 47

第 5 章 統合ファイル・システムのプログラミング・サポート 49
ストリーム・ファイルとデータベース・ファイルの間でのデータのコピー 49
CL コマンドによるデータのコピー 50
API によるデータのコピー 51
データ転送機能を使用したデータのコピー 51
ストリーム・ファイルと保管ファイルの間でのデータのコピー 54
API を使用した操作の実行 54
ILE C/400 の関数 59
API のラージ・ファイル・サポート 60
API のパス名規則 61
ファイル記述子 62
セキュリティー 62
ソケット・サポート 63
命名および国際サポート 63
データ変換 64

第 6 章 統合ファイル・システムのファイル・システム 65
ファイル・システムの比較 65
「ルート (/)」ファイル・システム 69
「ルート (/)」ファイル・システムの使用 70
オープン・システム・ファイル・システム (QOpenSys) 71
QOpenSys の使用 71
ユーザー定義ファイル・システム (UDFS) 72
UDFS の概念 73
統合ファイル・システム・インターフェースを介した UDFS の使用 74
ライブラリー・ファイル・システム (QSYS.LIB) 77
統合ファイル・システム・インターフェースを介した QSYS.LIB の使用 78
| 独立 ASP QSYS.LIB 80
| 統合ファイル・システム・インターフェースを介した独立 ASP QSYS.LIB の使用 80
文書ライブラリー・サービス・ファイル・システム (QDLS) 83
統合ファイル・システム・インターフェースを介した QDLS の使用 83
光ファイル・システム (QOPT) 85
統合ファイル・システム・インターフェースを介した QOPT の使用 86
NetWare ファイル・システム (QNetWare) 87
NetWare ファイル・システムのマウント 88
QNetWare ディレクトリー構造 88
統合ファイル・システム・インターフェースを介した QNetWare の使用 89

Windows NT Server ファイル・システム (QNTC). 91
統合ファイル・システム・インターフェースを介した QNTC の使用 91
OS/400 ファイル・サーバー・ファイル・システム (QFileSvr.400) 93
統合ファイル・システム・インターフェースを介した QFileSvr.400 の使用 94
ネットワーク・ファイル・システム (NFS) 97
統合ファイル・システム・インターフェースを介した NFS ファイル・システムの使用 97

第 7 章 統合ファイル・システム・オブジェクトのジャーナル処理サポート 101
ジャーナル管理 101
ジャーナル処理するオブジェクト 101
ジャーナル処理される統合ファイル・システム・オブジェクト 102
ジャーナル処理される操作. 103
ジャーナル項目についての特別な考慮事項 104

付録 A. トランスポート独立リモート・プロシージャー・コール 107
ネットワークの選択 107
名前からアドレスへの変換. 107
外部データ表示 (XDR) 108
認証 109
トランスポート独立 RPC (TI-RPC). 109
TI-RPC 単純化 API 110
TI-RPC 最上位 API 110
TI-RPC 中間レベル API. 110
TI-RPC エキスパート・レベル API 110
他の TI-RPC API 111

付録 B. 統合ファイル・システムの C 関数を使用するプログラムの例 113

付録 C. 統合ファイル・システムの RPG コードの例 119

参照文献 121

索引 123

『統合ファイル・システムの紹介』について

本書は、統合ファイル・システムの概要を提供するものです。以下の事項を記載しています。

- 統合ファイル・システムとはなにか
- 使用の目的
- 統合ファイル・システムの概念および用語
- 統合ファイル・システムとの対話に使用するインターフェース
- 統合ファイル・システムと対話するプログラムの作成に使用する API および技法
- 個々のファイル・システムの特性

『統合ファイル・システムの紹介』の対象読者

本書は、iSeries サーバーのユーザー、プログラマー、および管理者が、統合ファイル・システムとその使用方法を理解するためのものです。

コードの特記事項情報

本書には、プログラミング・サンプルが含まれています。

IBM は、お客様に、すべてのプログラミング・コード・サンプルを使用することができる非独占的な使用権を許諾します。お客様は、このサンプル・コードから、お客様独自の特別のニーズに合わせた類似のプログラムを作成することができます。

すべてのサンプル・コードは、例として示す目的でのみ、 IBM により提供されます。 このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。したがって IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

ここに含まれるすべてのプログラムは、現存するままの状態で提供され、いかなる保証条件も適用されません。 第三者の権利の不侵害、商品性、特定目的適合性に関する黙示の保証の適用も一切ありません。

第 1 章 統合ファイル・システムの紹介

以下のトピックでは、iSeries サーバー上での統合ファイル・システムについて説明するとともに、サーバーでの使用方法を紹介します。

統合ファイル・システムとは

統合ファイル・システムとは、OS/400 の一部であり、パーソナル・コンピューターおよび UNIX オペレーティング・システムと同様のストリーム入出力とストレージ管理をサポートするものです。また、サーバーに保管されるすべての情報の統合構造も提供します。

統合ファイル・システムの主な機能は、次のとおりです。

- 長い連続ストリングのデータを入れることができるストリーム・ファイルへの情報の保管。これらのデータのストリングは、文章のテキストや図の画素などです。ストリーム・ファイルのサポートは、クライアント / サーバー・アプリケーションで有効に使用できるように設計されています。
- 木の枝に実がつくようにオブジェクトを編成する階層ディレクトリー構造。オブジェクトへのディレクトリーのパスを指定すると、そのオブジェクトにアクセスできます。
- ユーザーおよびアプリケーションが、ストリーム・ファイル以外にも、データベース・ファイル、文書、およびサーバーに保管されている他のオブジェクトにアクセスできるようにする共通インターフェース。
- ご使用のサーバー、iSeries 統合 xSeries サーバー、またはリモート Windows NT Server にローカルに保管されているストリーム・ファイルの共通ビュー。ストリーム・ファイルを、ローカル・エリア・ネットワーク (LAN) サーバー、 Novell NetWare サーバー、別のリモート iSeries サーバー、またはネットワーク・ファイル・システム・サーバーにリモートに保管することもできます。

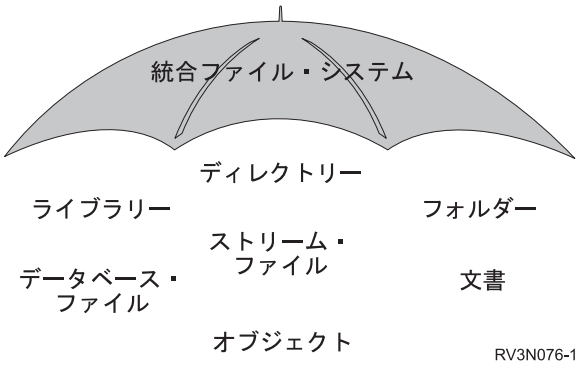


図 1. iSeries サーバーに保管される全情報の構造

統合ファイル・システム使用の利点

統合ファイル・システムは、OS/400 の広範なデータ管理機能をさらに拡張する機能を追加し、新しい情報処理形式 (クライアント / サーバー、オープン・システム、マルチメディアなど) をサポートします。

統合ファイル・システムを使用して、次の事柄を行うことができます。

- OS/400 データに、すばやく、アクセスすることができます (特に、Client Accessなどの OS/400 ファイル・サーバーを使用するアプリケーションの場合)。
- ストリーム・データのタイプ (イメージ、音声、ビデオなど) を、さらに効率よく処理できるようにします。
- UNIX ベースのオープン・システム標準仕様 (Portable Operating System Interface for Computer Environments (POSIX)、 XPG など) をサポートするファイル・システム・ベースおよびディレクトリー・ベースを提供します。このファイル構造とディレクトリー構造は、ディスク・オペレーティング・システム (DOS) や Windows オペレーティング・システムなどの PC オペレーティング・システムのユーザーが使い慣れている環境も提供します。
- 固有の機能をもつファイル・サポート (レコード形式のデータベース・ファイル、 UNIX ベースのストリーム・ファイル、およびファイル提供など) を、すべて共通インターフェースにより管理しながら、別々のファイル・システムとして処理することができます。

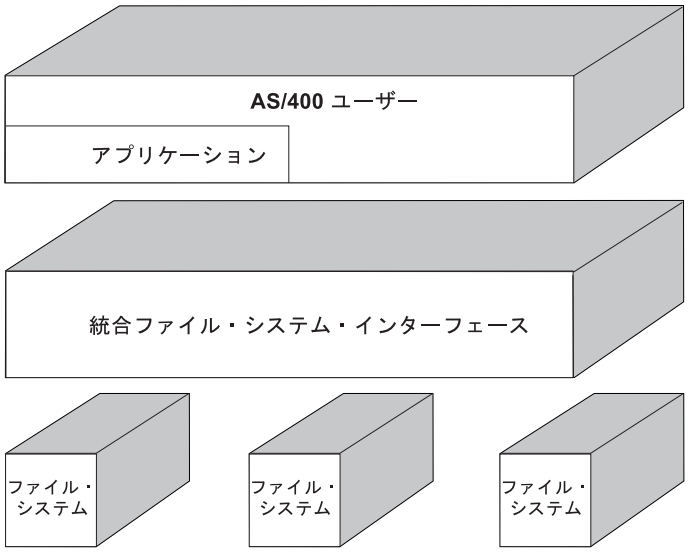


図 2. 別々のファイル・システムへの共通インターフェース

- PC ユーザーが、グラフィカル・ユーザー・インターフェースをさらに有効に活用できるようにします。たとえば、Windows ユーザーは、PC に保管されているファイルを操作するのと同様に、 Windows グラフィカル・ツールを使用して、 iSeries サーバーのストリーム・ファイルや他のオブジェクトを操作することができます。
- 一連のオブジェクト名と関連するオブジェクト情報を、各国語で提供します。たとえば、ある言語のコード・ページから別の言語のコード・ページに切り替えたときでも、それぞれの文字が変わることはありません。

第 2 章 統合ファイル・システムの概念

ストリーム・ファイル

ストリーム・ファイルとは、ランダムにアクセス可能なバイト列で、システムによって構造に制限が課されることはありません。統合ファイル・システムでは、ストリーム・ファイルの形式で情報を保管および操作します。サーバーのフォルダーでは、ストリーム・ファイルで文書が保管されています。 PC ファイルや UNIX システムのファイルも、ストリーム・ファイルの一例です。統合ファイル・システムのストリーム・ファイルは、オブジェクト・タイプが *STMF のシステム・オブジェクトです。

ストリーム・ファイルと iSeries データベース・ファイルを比較すると、ストリーム・ファイルをよりよく理解することができます。データベース・ファイルはレコード単位になっており、長さやデータ・タイプなどの特定の性質をもつ 1 つ以上のフィールドに、事前に区分されています。

図 3. ストリーム・ファイルとレコード単位ファイルの比較

ストリーム・ファイルとレコード単位ファイルの構造は違います。そしてこの構造上の違いにより、それぞれのファイルの使用方法が異なってきます。このような構造は、ファイルにアクセスするために、どのようにアプリケーションを作成するか、それぞれのタイプのファイルが、アプリケーションのどのような分野に適しているかなどに影響します。たとえば、名前、住所、および勘定残高などの顧客統計を保管するには、レコード単位ファイルの方が適しています。レコード単位ファイルを使用すると、サーバーの広範なプログラミング機能を使用して、ファイル内の事前定義フィールドを個々にアクセスしたり操作したりすることができます。一方、ストリーム・ファイルは、さまざまな色を表す一連のビットのストリングで作成された顧客の写真などを保管するのに適しています。ストリーム・ファイルは、文書のテキスト、イメージ、音声、およびビデオなどのデータのストリングを保管するのに特に適しています。

統合ファイル・システム内のストリーム・ファイルについての詳細は、以下を参照してください。

- 49 ページの『ストリーム・ファイルとデータベース・ファイルの間でのデータのコピー』
- 4 ページの『*TYPE1 および *TYPE2 のストリーム・ファイル』

*TYPE1 および *TYPE2 のストリーム・ファイル

ファイルは、2 つのフォーマット・オプションのいずれかを持っています。 *TYPE1 ストリーム・ファイルまたは *TYPE2 ストリーム・ファイルです。

*TYPE1 ストリーム・ファイルは、 OS/400 のバージョン 4 リリース 4 より前のリリースで作成されたストリーム・ファイルと同じフォーマットを持っています。このファイルは、OS/400 のバージョン 4 リリース 4 より前のリリースに対して、 *TYPE2 ストリーム・ファイルよりも速く保管されます。このファイルの最小サイズは 4096 バイトです。

*TYPE2 ストリーム・ファイルは、ハイパフォーマンスのファイル・アクセスができ、 OS/400 のバージョン 4 リリース 4 で新たに登場しました。このファイルは、OS/400 のバージョン 4 リリース 4 より前のリリースに対しては、 *TYPE1 ストリーム・ファイルよりも遅く保管されます。 このファイルの最小オブジェクト・サイズは 4096 バイトです。 V4R4 以上のシステムで作成されるすべてのファイルは *TYPE2 ストリーム・ファイルです。

*TYPE2 ストリーム・ファイルは V4R4 以上のシステムでしか作動しませんが、 V4R4 より前のシステム上で復元するために *TYPE2 ストリーム・ファイルを保管することができます。ただし、処理が遅くなります。

統合ファイル・システムのファイル・システム

ファイル・システムは、論理単位として編成された記憶域の特定のセグメントにアクセスできるように、サポートを提供します。サーバーの論理単位とは、ファイル、ディレクトリー、ライブラリー、およびオブジェクトです。

各ファイル・システムには、記憶域の情報にアクセスするための論理構造と規則のセットがあります。これらの構造と規則は、ファイル・システムによって異なります。この構造と規則という観点から見ると、ライブラリーによりデータベース・ファイルや、他のさまざまなオブジェクト・タイプにアクセスするための OS/400 サポートは、 1 つのファイル・システムと見なすことができます。同様に、フォルダー構造により文書 (実際はストリーム・ファイル) にアクセスするための OS/400 サポートは、別のファイル・システムと見なすことができます。

統合ファイル・システムでは、ライブラリー・サポートとフォルダー・サポートは、別々のファイル・システムとして扱われます。異なる機能をもつ他のタイプのファイル管理サポートも、別のファイル・システムとして扱われます。

各ファイル・システムの機能と制限事項の比較については、 65 ページの『ファイル・システムの比較』を参照してください。

統合ファイル・システム内のファイル・システムは、次のとおりです。

「ルート」 (/)
「ルート (/)」ファイル・システム。このファイル・システムは、統合ファイル・システムのストリーム・ファイル・サポートと階層ディレクトリー構造を有効に利用できます。ルート・ファイル・システムには、ディスク・オペレーティング・システム (DOS) と OS/2 ファイル・システムの特性があります。

QOpenSys
オープン・システム・ファイル・システム。このファイル・システムは、 POSIX や XPG などの UNIX ベースのオープン・システム標準と互換性があります。このファイル・システムは、ルー

ト・ファイル・システムと同様に、統合ファイル・システムが提供するストリーム・ファイルおよびディレクトリーのサポートを利用します。また、オブジェクト名の大文字小文字の区別もサポートします。

UDFS ユーザー定義のファイル・システム。このファイル・システムは、ユーザーが選択した補助記憶域プール (ASP)、または独立補助記憶域プール (ASP) にあります。このファイル・システムは、ユーザーが作成して管理します。

QSYS.LIB
ライブラリー・ファイル・システム。このファイル・システムは、サーバーのライブラリー構造をサポートします。このファイル・システムは、データベース・ファイルと、ライブラリー・サポートがシステムおよび基本ユーザー ASP 内で管理する、他のすべての iSeries サーバー・オブジェクト・タイプへのアクセスを提供します。

独立 ASP QSYS.LIB
独立 ASP QSYS.LIB ファイル・システム。このファイル・システムは、ユーザーが作成および定義する独立補助記憶域プール (ASP) 内のサーバー・ライブラリー構造をサポートします。このファイル・システムは、データベース・ファイルと、ライブラリー・サポートが管理する、他のすべての iSeries サーバー・オブジェクト・タイプへのアクセスを提供します。

QDLS 文書ライブラリー・サービス・ファイル・システム。このファイル・システムにより、文書とフォルダーにアクセスできます。

QOPT
光ファイル・システム。このファイル・システムにより、光メディアに保管されているストリーム・データにアクセスできます。

QNetWare
QNetWare ファイル・システム。このファイル・システムにより、 Novell NetWare 4.10 または 4.11 を実行しているサーバーに保管されているローカルまたはリモートのデータやオブジェクトにアクセスできます。また、Novell NetWare 3.12、4.10、4.11、または 5.0 を実行している独立型 PC サーバーにもアクセスできます。 既存のローカル・ファイル・システム上にある NetWare ファイル・システムを動的にマウントできます。

QNTC Windows NT Server ファイル・システム。このファイル・システムにより、 Windows NT 4.0 (またはそれ以降) を実行しているサーバーに保管されているデータやオブジェクトにアクセスできます。このファイル・システムを使用すると、 Windows NT クライアントと同じデータを iSeries サーバー・アプリケーションで使用できます。統合 PC サーバー上で実行している Windows NT Server 上のデータにもアクセスできます。詳細については、AS/400 Windows NT Server 統合機能 (SD88-5056) を参照してください。

QFileSvr.400
このファイル・システムにより、リモートの iSeries サーバーに常駐する他のファイル・システムにアクセスできます。

NFS ネットワーク・ファイル・システム。このファイル・システムにより、リモート NFS サーバーに保管されているデータや、オブジェクトにアクセスできます。 NFS サーバーからネットワーク・ファイル・システムをエクスポートしてから、 NFS クライアントに動的にマウントすることができます。

共通のインターフェースにより、すべてのファイル・システムにアクセスすることができます。このインターフェースは、データ管理インターフェースで提供されるレコード入出力とは対照的に、ストリーム・データの入出力用に最適化されています。備えられているコマンド、メニュー、および表示画面と、アプリケー

ション・プログラム・インターフェース (API) により、この共通インターフェースを介してファイル・システムと対話できます。

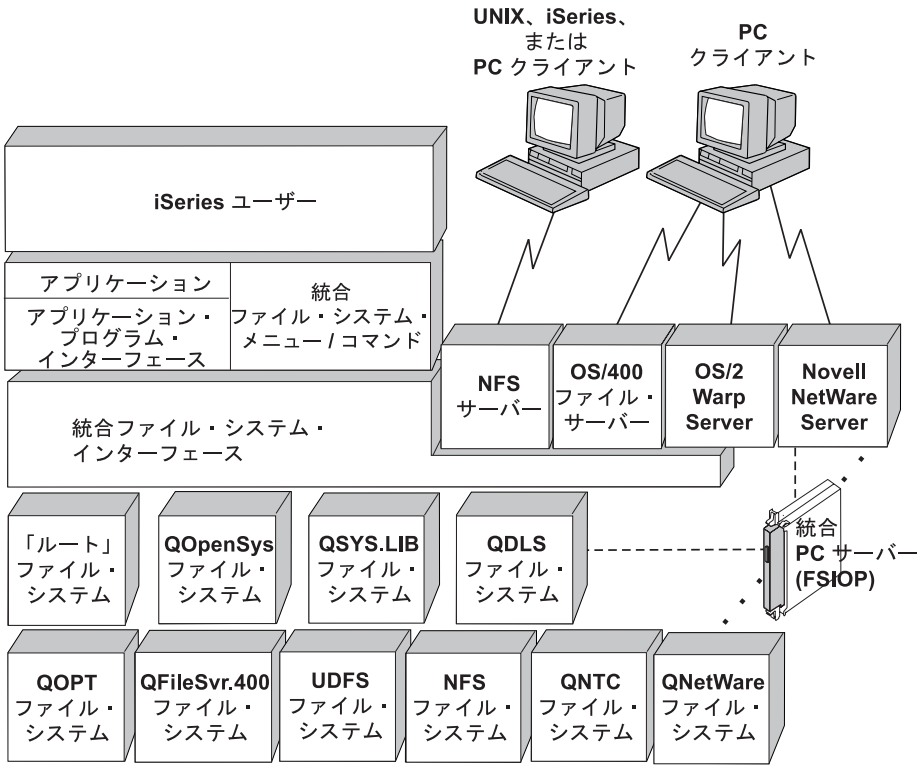


図 4. ファイル・システム、ファイル・サーバー、および統合ファイル・システム・インターフェース

詳細については、次のトピックおよび資料を参照してください。

- オプティカル・サポート
- OS/400 ネットワーク・ファイル・サポート

ディレクトリー

ディレクトリーとは、指定された名前でオブジェクトを探すために使用する、特殊なオブジェクトです。各ディレクトリーには、そこに属しているオブジェクトのリストが入っています。そのリストに、他のディレクトリーが含まれている場合もあります。

統合ファイル・システムは、階層ディレクトリー構造を提供し、サーバーのすべてのオブジェクトにアクセスできるようにします。このディレクトリー構造は、逆さになった木 (根が上にあり、枝が下にある) のようになっています。枝は、ディレクトリー階層の中のディレクトリーを表しています。これらのディレクトリーの枝から分かれている枝を、サブディレクトリーと呼びます。いくつかのディレクトリーおよびサブディレクトリーの枝には、ファイルなどのオブジェクトが付いています。オブジェクトを検索するには、オブジェクトが入っているサブディレクトリーに通じるディレクトリーへのパスを指定する必要があります。特定のディレクトリーに付いているオブジェクトは、そのディレクトリーに 『入っている』 と表現することもあります。

1 つのディレクトリーの枝は、そこから枝分かれしている枝 (サブディレクトリー) と、それらの枝に付加されているすべてのオブジェクトを含めて、**サブツリー**と呼ばれます。各ファイル・システムは、統合ファ

| イル・システムのディレクトリー構造の中の主要なサブツリーです。 QSYS.LIB および独立 ASP
| QSYS.LIB ファイル・システムのサブツリーでは、ライブラリーがサブディレクトリーと同様に扱われま
| す。ライブラリー内のオブジェクトは、サブディレクトリー内のオブジェクトと同様に扱われます。データ
| ベース・ファイルにはオブジェクト (データベース・ファイル・メンバー) が入っているため、データベー
| ス・ファイルはオブジェクトではなくサブディレクトリーとして扱われます。文書ライブラリー・サービ
| ス・ファイル・システム (QDLS サブツリー) では、フォルダーはサブディレクトリーと同様に扱われ、フ
| ォルダー内の文書はサブディレクトリー内のオブジェクトと同様に扱われます。

ファイル・システムが異なるために、ディレクトリー階層の中の 1 つのサブツリーで実行できる操作が、別のサブツリーでは実行できない場合もあります。

統合ファイル・システムのディレクトリー・サポートは、 DOS ファイル・システムで提供されるディレクトリー・サポートと同様のものです。その他に、ファイルを一度だけ保管し、リンクを使用して複数のパスでアクセスするなど、 UNIX システム特有の機能も提供します。

| 統合ファイル・システムのディレクトリーについての詳細は、以下のトピックを参照してください。

- 9 ページの『現行のディレクトリーとホーム・ディレクトリー』
- 38 ページの『統合ファイル・システムで提供されるディレクトリー』
- 9 ページの『*TYPE2 ディレクトリー』

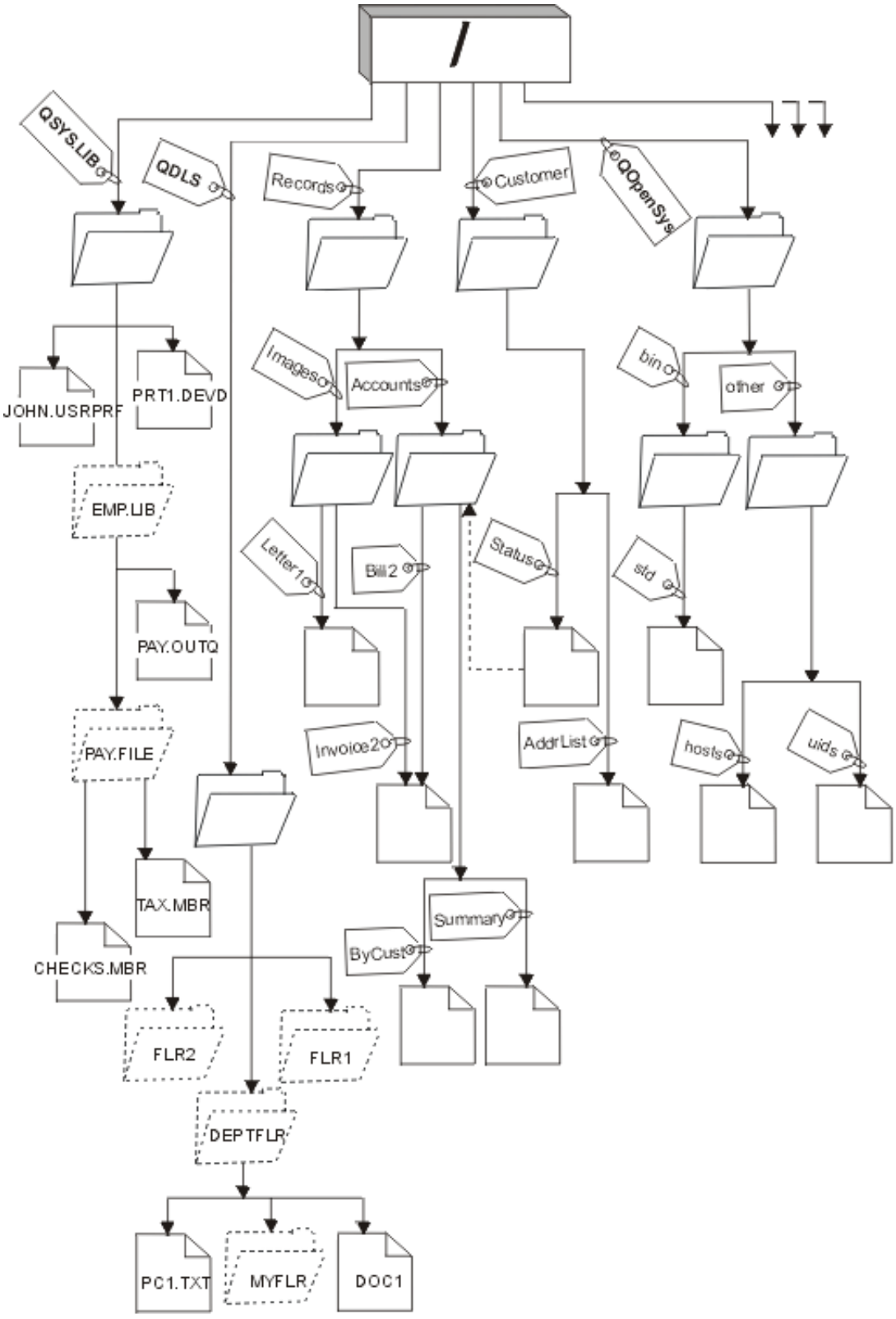


図 5. ファイル・システムおよびオブジェクトは、統合ファイル・システムのディレクトリー・ツリーの枝である

現行のディレクトリーとホーム・ディレクトリー

オペレーティング・システムが、まず最初にプログラムやファイルを検索したり一時ファイルや出力を保管したりするディレクトリーは、**現行ディレクトリー**です。ファイルなどのオブジェクトでの操作を要求するときに、現行ディレクトリー以外のディレクトリー・パスを指定しなければ、システムは現行ディレクトリーで、そのオブジェクトを検索します。現行ディレクトリーは、現行ライブラリーと同様の概念です。**現行作業ディレクトリー**、または**作業ディレクトリー**とも呼びます。

システムにサインオンすると、**ホーム・ディレクトリー**が現行ディレクトリーとして使用されます。ホーム・ディレクトリーの名前は、ユーザー・プロファイルに指定されています。ジョブを開始すると、システムはユーザー・プロファイルの中で、ホーム・ディレクトリー名を探します。その名前のディレクトリーがシステムに存在しなければ、ホーム・ディレクトリーは「ルート (/)」ディレクトリーに変更されます。

一般には、ユーザー・プロファイルを作成するシステム管理者が、ユーザーのホーム・ディレクトリーも作成します。個々のユーザーのホーム・ディレクトリーを /home ディレクトリーの下に作成することをお勧めします。 /home ディレクトリーは、「ルート (/)」ディレクトリーの下のサブディレクトリーです。システム・デフォルトでは、ユーザーのホーム・ディレクトリーの名前がユーザー・プロファイルの名前と同じであると予期されます。

たとえば、コマンド CRTUSRPRF USRPRF(John) HOMEDIR(*USRPRF) は、 John 氏のホーム・ディレクトリーを /home/JOHN に割り当てます。ディレクトリー /home/JOHN がない場合は、ルート (/) ディレクトリーが John 氏のホーム・ディレクトリーになります。

| サインオンした後には、現行ディレクトリーの変更 (CHGCURDIR) CL コマンド、 chdir( ) API、または
| fchdir() API を使用して、いつでもホーム・ディレクトリー以外に現行ディレクトリーを指定することができます。

デフォルトでは、プロセスを開始した時点で選択したホーム・ディレクトリーは、個々のスレッドのホーム・ディレクトリーにもなります。このことは、開始後にスレッドのアクティブ・ユーザー・プロファイルを変更しても、変わりありません。しかし、ジョブの変更 (QWTCHGJB) API によるサポートを使用して、スレッド用に使用されるホーム・ディレクトリーを、そのスレッドの現行のユーザー・プロファイルのホーム・ディレクトリー (ホーム・ディレクトリーがない場合は「ルート (/)」ディレクトリー) に変更できます。 2 次スレッドは、その 2 次スレッドを作成したスレッドのホーム・ディレクトリーを常に継承します。 QWTCHGJB を使用してスレッドのホーム・ディレクトリーを変更しても、プロセスの現行ディレクトリーは変更されないことに注意してください。現行ディレクトリーはプロセス・レベルで扱われ、ホーム・ディレクトリーはスレッド・レベルで扱われます。いずれかのスレッド中の現行作業ディレクトリーを変更すると、そのプロセス全体で現行作業ディレクトリーが変更されます。スレッドのホーム・ディレクトリーを変更しても、その現行作業ディレクトリーは変更されません。

QWTCHGJB API についての詳細は、 『Application programming interfaces (APIs)』のトピックを参照してください。

| *TYPE2 ディレクトリー

| 統合ファイル・システム内の「ルート (/)」、QOpenSys、およびユーザー定義ファイル・システム (UDFS) は、 *TYPE2 ディレクトリー・フォーマットをサポートします。 *TYPE2 ディレクトリー・フォーマットは、オリジナルの *TYPE1 ディレクトリー・フォーマットを拡張したものです。 *TYPE2 ディレクトリーは、*TYPE1 ディレクトリーとは異なる内部構造を持っており、インプリメンテーションも異なります。

| *TYPE2 ディレクトリーの利点は、以下のとおりです。

- パフォーマンスの向上
- 信頼性の向上
- 機能性の追加
- より少ない補助記憶域スペース (多くの場合)

*TYPE2 ディレクトリーは、*TYPE1 ディレクトリーよりも、特にディレクトリーの作成および削除時に、ファイル・システム・パフォーマンスが優れています。

*TYPE2 ディレクトリーは、*TYPE1 ディレクトリーよりも信頼性があります。システムが異常終了した後、補助記憶域障害がなければ、*TYPE2 ディレクトリーは完全に回復されます。 *TYPE1 ディレクトリーは、完全に回復するために記憶域の再利用 (RCLSTG) コマンドを使用する必要があります。

*TYPE2 ディレクトリーは、以下の追加機能を提供します。

1. *TYPE2 ディレクトリーは、上段専用ファイル・システムで名前の大文字小文字の名前変更 (たとえば、A から a への名前変更) をサポートします。
2. *TYPE2 ディレクトリー内のオブジェクトは、*TYPE1 ディレクトリーの 32,767 リンクに比べ、最高 1,000,000 個のリンクまで持つことができます。これは、ストリーム・ファイルへのハード・リンクを最高 1,000,000 個まで持つことができ、 *TYPE2 ディレクトリーに最高 1,000,000 個までのサブディレクトリーを含められることを意味しています。
3. iSeries ナビゲーターを使用すると、エントリーのリストは、 *TYPE2 フォーマットを持つディレクトリーをオープンするときに、自動的に 2 進数の順序でソートされます。

通常、350 個より少ないオブジェクトを持つ *TYPE2 ディレクトリーは、同じ数のオブジェクトを持つ *TYPE1 ディレクトリーよりも少ない補助記憶域を必要とします。 350 個より多くのオブジェクトを持つ *TYPE2 ディレクトリーは、 *TYPE1 ディレクトリーよりも 10 % (平均) 大きくなります。

ご使用のシステム上で *TYPE2 ディレクトリーを得るいくつかの方法があります。

- OS/400 V5R2 でインストールされたシステムではじめて独立補助記憶域プール (ASP) をオンに変更するとき、独立 ASP 内のユーザー定義ファイル・システム (UDFS) は *TYPE2 フォーマットに変換されます。
- 独立 ASP 上の UDFS を除き、サポートされているその他すべてのファイル・システムは、 ディレクトリーの変換 (CVTDIR) コマンドを使用して *TYPE2 に変換されなければなりません。
- OS/400 V5R2 に事前ロードされている新規 iSeries サーバーは *TYPE2 ディレクトリーを持っています。 ASP 1〜32 内の「ルート (/)」、QOpenSys、および UDFS の変換は必要ありません。
- iSeries サーバー上の OS/400 V5R2 のスクラッチ・インストールは *TYPE2 ディレクトリーを持っています。 ASP 1〜32 内の「ルート (/)」、QOpenSys、および UDFS の変換は必要ありません。

ご使用のサーバー上のファイル・システムのディレクトリー・フォーマットを判別するには、 CVTDIR (ディレクトリーの変換) コマンドを使用してください。

```
CVTDIR OPTION(*CHECK)
```

注: *TYPE2 ディレクトリーは OS/400 V5R1 でサポートされていますが、通常の *TYPE2 ディレクトリーのサポートとはいくつかの点が異なっています。詳細については、 『OS/400 V5R1 での *TYPE2 ディレクトリーの使用』を参照してください。

*TYPE2 ディレクトリーについての詳細は、以下のトピックを参照してください。

- *TYPE2 ディレクトリーへの変換

- 「ルート」、QOpenSys、または UDFS が使用できない場合
- 補助記憶域要件
- シンボリック・リンクに関する考慮事項
- 独立補助記憶域プール (ASP)
- 保管 / 復元に関する考慮事項
- *TYPE2 変換のための準備
- 変換処理
- 例: すべてのファイル・システムの変換 (少数のオブジェクト)
- 例: すべてのファイル・システムの変換 (多数のオブジェクト)
- 例: 特定の ASP のみの変換

OS/400 V5R1 での *TYPE2 ディレクトリーの使用

統合ファイル・システム内の「ルート (/)」、QOpenSys、およびユーザー定義ファイル・システム (UDFS) は、 OS/400 V5R1 で *TYPE2 ディレクトリーをサポートします。 *TYPE2 ディレクトリー・フォーマットは、オリジナルの *TYPE1 ディレクトリー・フォーマットを拡張したものです。 *TYPE2 ディレクトリーは、*TYPE1 ディレクトリーとは異なる内部構造を持っており、改善されたパフォーマンスおよび信頼性を提供します。

V5R1 をお持ちの場合、V5R1 ディレクトリーを *TYPE2 ディレクトリー・フォーマットに変換することができます。 OS/400 の新しいリリースをインストールする**前に**、 *TYPE2 ディレクトリー・フォーマットに変換することをお勧めします。これは、インストール中に自動的にディレクトリー変換が行われる場合があるので必要です。インストール中の自動変換の影響は、インストールに必要な時間が**著しく**増えることです。

注: OS/400 V5R1 または V5R2 にアップグレードする場合には、 *TYPE2 ディレクトリー・フォーマットへの自動変換は行われ**ません**。これらのインストールの前にディレクトリーを変換する必要はありません。

V5R1 での *TYPE2 ディレクトリーのサポートは修正 (PTF) によって使用可能になります。変換ユーティリティーは V5R2 バージョンとはわずかに異なっています。 V5R1 での *TYPE2 ディレクトリーの完全な資料については、情報 APAR II13161 を参照してください。以下の方式のいずれかを使用して APAR にアクセスしてください。

1. 情報 APAR をご使用の iSeries サーバーにダウンロードし、表示します。以下のコマンドを使用します。

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

2. http://www-912.ibm.com Web サイトに進んで、情報 APAR をオンライン表示します。**「許可プログラム分析報告書 (Authorized Program Analysis Reports (APARs))」-->「V5R1 APAR」-->「APAR 番号 II13161 (APAR number II13161)」**を選択します。

*TYPE2 ディレクトリーへの変換

CVTDIR コマンドは、 *TYPE1 ディレクトリーから *TYPE2 ディレクトリーへの変換を実行します。さらに、このコマンドは、*TYPE2 ディレクトリー・フォーマットにファイル・システムを変換する方法に関する情報を提供します。 CVTDIR は、以下のことを行います。

- *TYPE2 ディレクトリーをサポートする既存のファイル・システムの現行ディレクトリー・フォーマットをリストします。
- 変換を行うためにかかる時間を見積もります。
- 変換のための補助記憶域要件を見積もります。
- ファイル・システムを *TYPE2 フォーマットに変換します。既存のディレクトリーは *TYPE2 に変換され、変換後に作成される新規ディレクトリーは *TYPE2 です。

いずれかのファイル・システムにあるディレクトリーが変換されるいくつかの方法があります。

- CVTDIR コマンドを使用することにより、手動で変換
- OS/400 V5R2 がインストールされているシステムで、はじめて独立 ASP をオンに変更するときに、自動的に変換
- システムが異常終了したときにファイル・システムの変換が進行中だったことをシステムが判別する場合、IPL 中に変換
- *TYPE2 フォーマットに変換されたファイル・システムの一部である、脱落した *TYPE1 ディレクトリーが見つかった場合、 記憶域の再利用 (RCLSTG SELECT(*ALL)) の際に変換

「ルート」、QOpenSys、または UDFS が使用できない場合

「ルート (/)」または QOpenSys ファイル・システムの変換は、システムが制限状態にあるときに行われなければなりません。 UDFS を変換するとき、システムは制限状態にある必要はありません。ただし、その ASP 内の UDFS は変換中には使用できません。変換を実行するために必要な時間の長さは、ファイル・システムのサイズによって異なります。したがって、変換を実行するための最善の時をスケジュールするために計画が必要です。 CVTDIR コマンドの *ESTIMATE オプションは、指定されたファイル・システムを変換するために必要な時間の長さを見積もります。見積もった時間の長さは最高見積値です。これは、単一スレッドを持つジョブで実行される変換に基づいて時間の長さを見積もります。実際の変換では複数のスレッドを使用し、見積時間よりも短い時間で済みます。通常、40,000 個を超えるリンクを持つファイル・システムは、見積時間の 30 % ～ 50 % で変換できます。ただし、実際の時間は、ハードウェアおよびサーバーの構成によって異なります。

CVTDIR コマンドが実行している間にシステムが異常終了する場合、後続の IPL 中に、変換機能は SCPF ジョブ内で実行します。 SCPF ジョブは、複数のスレッドがアクティブになることを許可しません。したがって、ファイル・システムの変換が IPL 中に完了されなければならないとき、変換は単一スレッドを使用して実行します。変換機能は、IPL 中に SRC C900 2A85 が表示され、変換の進行を示す状況メッセージ CPIA089 が表示されるときに実行します。

すでに *TYPE2 に変換されたファイル・システムからの脱落 *TYPE1 ディレクトリーがある場合、RCLSTG 中に変換機能が実行されます。変換機能は、RCLSTG コマンドを発行するジョブ内で実行されます。変換を必要とする脱落ディレクトリーが見つかった場合、システム制限のため、変換は単一スレッドで実行します。

補助記憶域要件

ファイル・システム内のディレクトリーを *TYPE2 フォーマットに変換する前に、補助記憶域要件を考慮する必要があります。補助記憶域要件に関するいくつかの考慮点があります。

- *TYPE2 フォーマットに変換された後のディレクトリーの最終サイズ
- 変換機能が実行している間に必要な追加記憶域

多くの場合、*TYPE2 ディレクトリーの最終サイズは *TYPE1 ディレクトリーよりも小さくなります。通常、350 個より少ないオブジェクトを持つ *TYPE2 ディレクトリーは、同じ数のオブジェクトを持つ

*TYPE1 ディレクトリーよりも少ない補助記憶域を必要とします。 350 個より多くのオブジェクトを持つ *TYPE2 ディレクトリーは、 *TYPE1 ディレクトリーよりも 10 % (平均) 大きくなります。

変換機能が実行している間、追加記憶域が必要です。変換機能では、かなりの数のディレクトリー内に *TYPE1 バージョンと *TYPE2 バージョンの両方が同時に存在している必要があります。この数は、iSeries サーバー構成および変換されているファイル・システムのディレクトリー構造によって異なります。

CVTDIR コマンド上の *ESTIMATE オプションは、変換中に必要な補助記憶域の見積量を示す情報を提供します。

シンボリック・リンクに関する考慮事項

シンボリック・リンクは、別のオブジェクトへのパスを含む統合ファイル・システム内のオブジェクトです。変換中にオブジェクトの名前が変更される、いくつかの場合があります。シンボリック・リンク内のパスのエレメントの 1 つが変換中に名前変更される場合、シンボリック・リンクの内容はもはやそのオブジェクトを指しません。オブジェクトの名前変更についての詳細は、『名前変更されるオブジェクト』を参照してください。

独立補助記憶域プール (ASP)

OS/400 V5R2 でインストールされたシステムではじめて独立 ASP をオンに変更するとき、ディレクトリーは *TYPE2 に変換されます。計画の目的のために、変換の実行時間の長さに関する情報を提供するための見積もり機能が OS/400 V5R1 で提供されます。 V5R2 サーバーへの独立 ASP をオンに変更する前に、独立 ASP (ASP_NAME という名前) がオンに変更され、アクティブになるとき、 V5R1 システム上で以下の API を実行してください。

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

注: この機能を呼び出す前に、 V5R1 システム上の独立 ASP 上で RCLSTG を実行することをお勧めします。

保管 / 復元に関する考慮事項

*TYPE1 として存在するディレクトリーは、 *TYPE2 に変換されたファイル・システム内に保管および復元できます。同様に、ディレクトリーが *TYPE2 ディレクトリーとして存在するときに *TYPE1 制限を超えない場合、 TYPE2 として存在するディレクトリーは *TYPE1 フォーマットのファイル・システム内に保管および復元できます。

*TYPE2 変換のための準備

*TYPE2 ディレクトリーに変換する前に推奨されるいくつかの CL コマンドおよびパラメーターがあります。

- 記憶域の再利用 (RCLSTG)

  ファイル・システムを変換する前に RCLSTG SELECT(*ALL) コマンドを使用すると、ディレクトリーをクリーンアップし、ディレクトリーを良い状態に保ちます。これは、ディレクトリー変換中に遭遇する可能性のあるすべての問題を除去するわけではありませんが、確実にファイル・システム内のディレクトリーを読み取ることができるようにします。

  このコマンドは、CVTDIR コマンドのいずれかオプションを使用する前に 1 度だけ実行する必要があります。

- システムの保管 (SAVSYS)

iSeries サーバーのシステム全体の保管は、RCLSTG を実行した後、 CVTDIR コマンドの *CONVERT オプションを使用する前に行う必要があります。システムをバックアップするには、iSeries の「保管」メニューを使用してください。「保管」メニューに進むには、コマンド行で GO SAVE と入力し、オプション 21 を選択してください。詳細については、バックアップおよび回復の手引き  を参照してください。

このコマンドは、CVTDIR コマンドのいずれかオプションを使用する前に 1 度だけ実行する必要があります。

- CVTDIR コマンドの *ESTIMATE オプション

  CVTDIR コマンドの *ESTIMATE オプションを使用して、ディレクトリーを変換するために必要な時間を判別します。

  時間および補助記憶域の見積もりを提供することに加え、 *ESTIMATE オプションにはさらに利点があります。このオプションは、*TYPE1 ディレクトリーと関連した 2 次オブジェクトを構築し、変換をより速く実行できるようにします (それらを作成する必要がなくなるため)。ファイル・システムを *TYPE2 フォーマットに変換するために *CONVERT オプションが使用されるまで、これらの 2 次オブジェクトは存在したままです。 *ESTIMATE オプションは、ファイル・システム内のすべてのディレクトリーを読み通し、暗黙的にディレクトリーを検査します。 *ESTIMATE オプションは、実際の変換中に発生する可能性のあるすべてのエラーを見つけるという保証はありませんが、そうする助けになります。

  *ESTIMATE オプションを実行した後、ジョブ・ログ内のエラーを調べ、ファイル・システムを変換する前に、推奨される回復アクションを実行してください。推奨される回復アクションを実行した後に再び見積もりを実行することは必須ではありませんが、その他の問題がないかを検査するために再び見積もりを実行するようお勧めします。

- 補助記憶域に関する考慮事項

  変換されているファイル・システムを含む ASP 用の使用可能な補助記憶域を確認してください。 CVTDIR *ESTIMATE オプションは、メッセージ CPIA090 を表示します。このメッセージは、ASP 用の補助記憶域の使用可能量を示しています。さらに、このメッセージは、変換中に必要と予期される補助記憶域の量を表示します。メッセージ CPIA091 も表示されます。このメッセージは、変換後のファイル・システム内の *TYPE2 ディレクトリーの合計サイズが既存の *TYPE1 ディレクトリーよりも大きくまたは小さく見積もられているかどうかを示しています。 ASP 内の使用可能な記憶域は、使用されていない補助記憶域 (メッセージ CPIA090 内に表示されている) と、 *TYPE1 ディレクトリー・サイズと *TYPE2 ディレクトリー・サイズとの間の差 (メッセージ CPIA091 内に表示されている) の合計であるはずです。

  また、使用可能な補助記憶域を検出するには、システム・サービス・ツールの開始 (STRSST) コマンドを使用し、 **ディスク装置の処理**オプションを選択します。

  注: 1 つの ASP のみがシステム上に定義されている場合は、システム状況の処理 (WRKSYSSTS) コマンドを使用するだけで、使用可能な補助記憶域情報を表示できます。

  CVTDIR コマンドでいずれかのオプションを使用する前に、一般的なシステム・クリーンアップを実行することをお勧めします。必要でなくなったディレクトリーまたはファイルがある場合には、 CVTDIR コマンドのいずれかのオプションを使用する前にそれらを除去してください。そのように、補助記憶域スペースを解放することにより、使用可能な補助記憶域スペースをより正確に見積もることができ、処理が必要なオブジェクトがより少なくなるので、より短い時間で変換を完了することができます。

- CVTDIR コマンドを発行しているジョブについて、ジョブ・メッセージ・キューの全アクションを *PRTWRAP へ変更することを考慮してください。それによって、以下のようになります。

  1. ジョブ・ログがいっぱいになった場合にジョブが異常終了しないようにします。

2. ジョブ・ログが折り返す場合、スプール・ファイルにオーバーレイ・メッセージをプリントします。したがって、重要なメッセージは失われません。

- 独立 ASP を持つシステムの場合: OS/400 V5R2 を実行しているシステムで独立 ASP をオンに変更する前に、すべての独立 ASP 上で V5R1 *ESTIMATE 機能を使用します。これは、インストール後に最初に独立 ASP をオンに変更するまでにかかる時間の見積もりを提供します。詳細については、独立補助記憶域プール (ASP) を参照してください。

変換処理

CVTDIR コマンドは、*TYPE1 ディレクトリーを *TYPE2 ディレクトリーに変換します。変換処理の際に考慮すべきいくつかの点があります。

- 「ルート (/)」または QOpenSys
- ユーザー定義ファイル・システムの変換
- ユーザー・プロファイルの作成
- 名前変更されるオブジェクト
- ユーザー・プロファイルに関する考慮事項

「ルート」または QOpenSys の変換

「ルート」または QOpenSys ファイル・システムを変換するとき、システムは制限状態**でなければなりません**。変換中にファイル・システムのいずれかを使用することはできません。すべての UDFS および NFS ファイル・システムは、 CVTDIR コマンドによってアンマウントされ、変換が完了するときに再マウントされません。 UDFS または NFS ファイル・システムを再マウントするために、マウント・コマンド (MOUNT) を使用できます。

ユーザー定義ファイル・システムの変換

ASP 1〜32 内の UDFS を変換するとき、システムを使用しているユーザーがそれらを使用することはできません。これらの各 ASP ごとに、/dev ディレクトリー内に QASPxx ディレクトリーがあります。CVTDIR コマンドが実行している間、それはネーム・スペースから QASPxx ディレクトリーを除去し、ユーザーが ASP 内の UDFS にアクセスできないようにします。 CVTDIR コマンドが、QASPxx ディレクトリーを含め、すべてのオブジェクトの処理を終了するとき、オブジェクトはネーム・スペースに戻り、システム上のユーザーが使用できるようになります。 ASP 用の UDFS は、CVTDIR コマンドによってアンマウントされ、変換が完了するときに再マウントされません。 UDFS を再マウントするために、マウント・コマンド (MOUNT) を使用できます。

注: QASP01 ディレクトリーはすべてのシステム上に存在します。

ユーザー・プロファイルの作成

変換機能は、変換機能が実行している間に使用されるユーザー・プロファイルを作成します。これらのユーザー・プロファイルには QP0FCVxxxx という名前があります。ここで、xxxx は 0001 などの番号です。ユーザー・プロファイルは、元の所有者が自分のディレクトリーを所有することができない場合に、変換されるファイル・システム内のディレクトリーを所有するために、変換機能によって使用されます。

これらのユーザー・プロファイルは、可能であれば、変換が完了するときに削除されます。ディレクトリーの所有権がこれらのユーザー・プロファイルのいずれかに付与される場合、メッセージ CPIA08B が送信されます。

名前変更されるオブジェクト

*TYPE2 ディレクトリーでは、リンク名が有効な UTF-16 の名前である必要があります。これは、UCS2 レベル 1 の名前を持っている *TYPE1 ディレクトリーとは異なっています。そのため、ディレクトリー変

| 換中に無効な名前や重複している名前が見つかる場合があります。無効な名前や重複している名前が見つかるとき、名前は固有で、有効な UTF-16 の名前に変更され、元の名前と新規名をリストしているメッセージ CPIA08A がジョブ・ログに送信されます。名前の中に結合文字または無効なサロゲート文字の対が含まれていると、オブジェクトが名前変更される場合があります。

| UTF-16 についての詳細は、Unicode ホーム・ページ (http://www.unicode.org ) を参照してください。

| ***結合文字:*** 文字の中には複数の Unicode 文字で成り立っているものもあります。たとえば、アクセントやウムラウトを持つ文字があります。これらの文字は、すべてのオブジェクトが固有名を持つようにするため、ディレクトリー内に保管される前に、共通フォーマットに変更もしくは正規化される必要があります。結合文字の正規化は、文字が既知で予測可能なフォーマットに入れられるプロセスです。 *TYPE2 ディレクトリー用に選択されたフォーマットは、正規の合成形式です。同じ結合文字を含む *TYPE1 ディレクトリー内の 2 つのオブジェクトがある場合、同じ名前に正規化されます。これにより、一方のオブジェクトに合成結合文字が含まれており、他方のオブジェクトに分解結合文字が含まれている場合でさえ、衝突が起こります。したがって、*TYPE2 ディレクトリー内にリンクされる前に、それらのオブジェクトのうちの一方がその名前を変更します。

| ***サロゲート文字:*** 文字の中には Unicode 内に有効な表記を持っていないものもあります。これらの文字は、いくつかの特殊値を持っており、最初の Unicode 文字は 1 つ目の範囲 (たとえば、0xD800〜0xD8FF) 内にあり、 2 番目の Unicode 文字は 2 番目の範囲 (たとえば、0xDC00〜0xDCFF) 内にあるといった、 2 つの特定の範囲内の 2 つの Unicode 文字から成り立っています。これは、サロゲート対と呼ばれます。Unicode 文字の 1 つが脱落しているか、または規定外の場合 (部分文字のみ)、無効な名前になります。このタイプの名前は、*TYPE1 ディレクトリー内では許可されますが、*TYPE2 ディレクトリー内では許可されません。変換機能を継続するために、これらの無効な名前のいずれかを含む名前が見つかる場合、オブジェクトが *TYPE2 ディレクトリーにリンクされる前に名前は変更されます。

| ユーザー・プロファイルに関する考慮事項

| 変換が実行している間、*TYPE1 ディレクトリーを所有する同じユーザー・プロファイルが、対応する *TYPE2 ディレクトリーを引き続き所有するようにするために、すべての試みが行われます。 *TYPE1 および *TYPE2 ディレクトリーが瞬間的に同時に存在するので、これはユーザー・プロファイルが所有するストレージの量、およびユーザー・プロファイル内のエントリーの数に影響を与えます。

| ***ユーザー・プロファイル用の最大ストレージの変更:*** ディレクトリー変換の処理中に、瞬間的に同時に両方のフォーマットで存在し、同じユーザー・プロファイルによって所有される多数のディレクトリーがあります。ユーザー・プロファイル用の最大ストレージ制限に到達しているために *TYPE2 ディレクトリーを作成できない場合、ユーザー・プロファイル用の最大ストレージ制限は増やされます。メッセージ CPIA08C がジョブ・ログに送信され、変換は継続します。

| ***ディレクトリーの所有者の変更:*** *TYPE1 ディレクトリーを所有するユーザー・プロファイルが、作成される *TYPE2 ディレクトリーを所有することができない場合、*TYPE2 ディレクトリーの所有者は、 『ユーザー・プロファイルの作成』で説明されている代替ユーザー・プロファイルのいずれかに設定されます。メッセージ CPIA08B がジョブ・ログに送信され、変換は継続します。

| 例: すべてのファイル・システムの変換 (少数のオブジェクト)

| システム A は 5 つの補助記憶域プール (ASP) で構成されています。すなわち、1 (システム ASP)、3、5、11、および 25 です。システム上で *TYPE1 ディレクトリーから *TYPE2 ディレクトリーに変換されたファイル・システムはありません。すべてのファイル・システムを変換したいとします。ファイル・システムは、多数のオブジェクトを含んでいません。したがって、1 日にすべてのステップを実行することを計画します。

| 少数のオブジェクトを持つすべてのファイル・システム内のディレクトリーを変換するには、以下のようにします。

1. システムを制限状態に置きます。
2. コマンド行上に RCLSTG SELECT(*ALL) と入力します。
3. 「保管」メニューを使用してシステムを保管します。コマンド行上に GO SAVE と入力し、オプション 21 を選択します。
4. コマンド行上に CVTDIR OPTION(*ESTIMATE) FILESYS(*ALL) FORMAT(*TYPE2) と入力します。
5. *ESTIMATE 機能からエラー・メッセージを調べます。
6. すべての ASP が使用可能な十分の補助記憶域スペースを持っていることを確認します。
7. コマンド行上に CVTDIR OPTION(*CONVERT) FILESYS(*ALL) FORMAT(*TYPE2) と入力します。

   注: すべてのファイル・システムを変換するときは (*ALL)、メッセージ CPAA084 が表示され、リストされたファイル・システムを変換してよいか確認を求められます。
8. *CONVERT 機能からエラー・メッセージを調べます。
9. システムを制限状態から解放します。

例: すべてのファイル・システムの変換 (多数のオブジェクト)

システム B も 5 つの補助記憶域プール (ASP) で構成されています。すなわち、1 (システム ASP)、3、5、11、および 25 です。 システム上で *TYPE1 ディレクトリーから *TYPE2 ディレクトリーに変換されたファイル・システムはありません。 すべてのファイル・システムを変換したいとします。 ファイル・システムは多数のオブジェクトを含んでいます。したがって、2 つの別の週末に変換ステップを実行することを計画します。

週末 1:

1. システムを制限状態に置きます。
2. コマンド行上に RCLSTG SELECT(*ALL) と入力します。
3. 「保管」メニューを使用してシステムを保管します。コマンド行上に GO SAVE と入力し、オプション 21 を選択します。
4. システムを制限状態から解放します。

平日:

5. コマンド行上に CVTDIR OPTION(*ESTIMATE) FILESYS(*ALL) FORMAT(*TYPE2) と入力します。
6. *ESTIMATE 機能からエラー・メッセージを調べます。
7. すべての ASP が使用可能な十分の補助記憶域スペースを持っていることを確認します。

週末 2:

8. システムを制限状態に置きます。
9. コマンド行上に CVTDIR OPTION(*CONVERT) FILESYS(*ALL) FORMAT(*TYPE2) と入力します。

   注: すべてのファイル・システムを変換するときは (*ALL)、メッセージ CPAA084 が表示され、リストされたファイル・システムを変換してよいか確認を求められます。
10. *CONVERT 機能からエラー・メッセージを調べます。
11. システムを制限状態から解放します。

例: 特定の ASP のみの変換

システム C は 6 つの補助記憶域プール (ASP) で構成されています。すなわち、1 (システム ASP)、2、4、8、10、および 30 です。システム上で変換されたファイル・システムはありません。 ASP 4、10、および 30 のみで UDFS を変換したいとします。

ある特定の ASP 上の UDFS 内のディレクトリーを変換するには、以下のようにします。

1. ファイル・システムのディレクトリー・フォーマットを確認します。これを行うには、コマンド行上に CVTDIR OPTION(*CHECK) と入力します。
2. システムを制限状態に置きます。
3. コマンド行上に RCLSTG SELECT(*ALL) と入力します。
4. 「保管」メニューを使用してシステムを保管します。コマンド行上に GO SAVE と入力し、オプション 21 を選択します。
5. システムを制限状態から解放します。
6. コマンド行上に CVTDIR OPTION(*ESTIMATE) FILESYS(*UDFS) ASP(4) FORMAT(*TYPE2) と入力します。
7. *ESTIMATE 機能からエラー・メッセージを調べます。
8. コマンド行上に CVTDIR OPTION(*ESTIMATE) FILESYS(*UDFS) ASP(10) FORMAT(*TYPE2) と入力します。
9. *ESTIMATE 機能からエラー・メッセージを調べます。
10. コマンド行上に CVTDIR OPTION(*ESTIMATE) FILESYS(*UDFS) ASP(30) FORMAT(*TYPE2) と入力します。
11. *ESTIMATE 機能からエラー・メッセージを調べます。
12. すべての ASP が使用可能な十分の補助記憶域スペースを持っていることを確認します。
13. コマンド行上に CVTDIR OPTION(*CONVERT) FILESYS(*UDFS) ASP(4) FORMAT(*TYPE2) と入力します。
14. *CONVERT 機能からエラー・メッセージを調べます。
15. コマンド行上に CVTDIR OPTION(*CONVERT) FILESYS(*UDFS) ASP(10) FORMAT(*TYPE2) と入力します。
16. *CONVERT 機能からエラー・メッセージを調べます。
17. コマンド行上に CVTDIR OPTION(*CONVERT) FILESYS(*UDFS) ASP(30) FORMAT(*TYPE2) と入力します。
18. *CONVERT 機能からエラー・メッセージを調べます。

パス名

パス名 (一部のシステムでは **pathname** と呼ばれる) は、サーバーにオブジェクトを見つける方法を指示するものです。パス名は、ディレクトリー名のあとにオブジェクト名が続く形式になっています。ディレクトリー名とオブジェクト名は、それぞれスラッシュ (/) で区切ります。たとえば、次のようになります。

```
directory1/directory2/file
```

ユーザーの便宜のために、統合ファイル・システムでは、スラッシュでなく円記号 (¥) が使用できます。

パス名の識別方法には、次の 2 つがあります。

- **絶対パス名**は、最高レベルまたは「ルート」ディレクトリー (/ で示される) から始まります。たとえば、/ ディレクトリーから Smith という名前のファイルへのパスを考えてみてください。

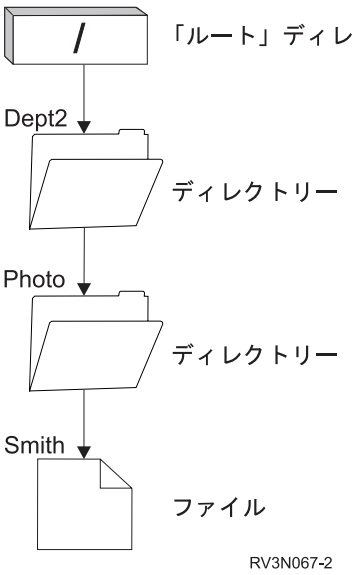


図 6. パス名の構成要素

Smith ファイルの絶対パス名は、次のようになります。

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

絶対パス名は、**完全パス名**とも呼ばれます。

- パス名が / で始まっていなければ、システムは、現行ディレクトリーからパスが始まるものと見なします。このようなパス名は、**相対パス名**と呼ばれます。たとえば、現行ディレクトリーが Dept2 で、Smith ファイルを含む Photo という名前のサブディレクトリーがある場合、ファイルへの相対パス名は、次のようになります。

  ```
  Photo/Smith
  ```

  パス名には、現行ディレクトリー名は含まれません。名前の最初の項目は、現行ディレクトリーの *1 レベル下* のディレクトリーまたはオブジェクトです。

リンク

リンクとは、ディレクトリーとオブジェクトの間の名前付きの結合です。ユーザーまたはプログラムは、オブジェクトとのリンクをサーバーに指定して、そのオブジェクトの所在を示します。リンクは、パス名またはその一部として使用することができます。

ディレクトリー・ベースのファイル・システムのユーザーは、オブジェクトを、サーバーで識別できる名前をもつファイルのようなものと考えてください。オブジェクトは、そのオブジェクトのディレクトリー・パスで識別されます。オブジェクトの 『名前』 を指定するだけで、オブジェクトにアクセスできる場合もあります。これを行うことができるのは、該当するパスのディレクトリー部分を一定の条件に想定するように、システムが設計されているためです。リンクという観念は、オブジェクトを識別するのはディレクトリー・パスであるという事実を利用したものです。名前は、オブジェクトではなく、リンクに付けられるものです。

オブジェクトではなく、リンクが名前をもつという観念に慣れると、以前には考えられなかった多くの可能性が見えてきます。 1 つのオブジェクトに、複数のリンクを設定することができます。たとえば、2 人のユーザーがそれぞれのホーム・ディレクトリーから同じファイルにリンクして、 1 つのファイルを共用す

ることができます (9 ページの『現行のディレクトリーとホーム・ディレクトリー』を参照)。リンクの中には、複数のファイル・システムにわたったり、オブジェクトが存在しなくてもリンクだけ存在できるものもあります。

リンクには、**ハード・リンク**と**シンボリック・リンク**の 2 種類があります。

リンクについての詳細は、以下のトピックを参照してください。

- ハード・リンク
- シンボリック・リンク
- 比較: ハード・リンクとシンボリック・リンク

ハード・リンク

ハード・リンクは、単にリンクと呼ばれることもあり、実際のオブジェクトにリンクしていなければなりません。ディレクトリーにオブジェクトが (ディレクトリーにファイルをコピーするなどの方法で) 作成されると、ディレクトリーとオブジェクトの間に最初のハード・リンクが設定されます。ユーザーおよびアプリケーション・プログラムが、別のハード・リンクを追加することもできます。それぞれのハード・リンクは、ディレクトリー内の別々のディレクトリー項目で識別されます。同じディレクトリーからのリンクは同じ名前にできませんが、異なるディレクトリーからのリンクは同じ名前でもかまいません。

ファイル・システムでサポートされていれば、 1 つのオブジェクトが複数のハード・リンク (同じディレクトリーからでも、異なるディレクトリーからでもかまいません) をもつことができます。ただし、オブジェクトが別のディレクトリーである場合は例外です。ディレクトリーとディレクトリーとの間には、1 つしかハード・リンクを設定することができません。

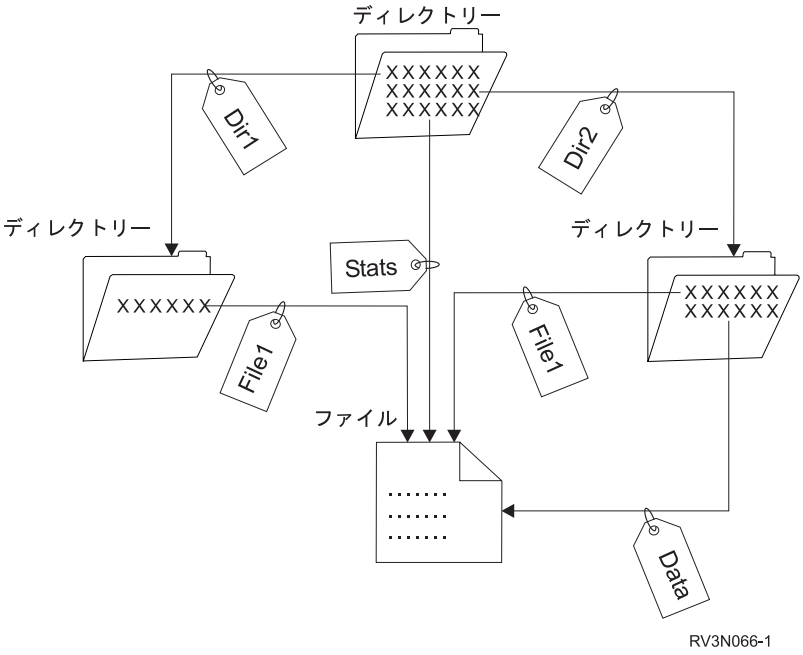


図 7. 各ハード・リンクを定義するディレクトリー項目

ハード・リンクは、そのオブジェクトに対するハード・リンクが少なくとも 1 つ残っていれば、オブジェクトに影響を与えることなく除去することができます。最後のハード・リンクが除去されたときに、そのオブジェクトがアプリケーションでオープンされていなければ、オブジェクトはサーバーから削除されます。オブジェクトをオープンしているアプリケーションでは、各アプリケーションが該当のオブジェクトをクロ

ーズするまで使用することができます。すべてのアプリケーションでオブジェクトの使用が終了すると、そのオブジェクトはサーバーから削除されます。すべてのハード・リンクを除去したあとに、そのオブジェクトをオープンすることはできません。

ハード・リンクの概念は、 QSYS.LIB または独立 ASP QSYS.LIB ファイル・システムおよび文書ライブラリー・サービス (QDLS) ファイル・システムにも適用できますが、制限があります。実際に、ライブラリーとその中の各オブジェクトの間には、1 つずつハード・リンクが設定されています。同様に、フォルダーとその中の各文書の間には、1 つずつハード・リンクが設定されています。ただし、QSYS.LIB、独立 ASP QSYS.LIB、または QDLS では、*同じオブジェクト* に複数のハード・リンクを設定することはできません。

ハード・リンクは、複数のファイル・システムにわたることはできません。たとえば、QOpenSys ファイル・システムのディレクトリーでは、 QSYS.LIB または独立 ASP QSYS.LIB ファイル・システム内のオブジェクトへのハード・リンクや、 QDLS ファイル・システムの文書へのハード・リンクを設定することはできません。

シンボリック・リンク

シンボリック・リンクは、ファイルに含まれるパス名であり、ソフト・リンクとも呼ばれます。システムは、シンボリック・リンクを検出すると、シンボリック・リンクが提供したパス名をたどり、シンボリック・リンクに続く残りのパスをたどります。パス名が / で始まっていれば、システムは / (「ルート」) ディレクトリーに戻り、そこからのパスをたどります。パス名が / で始まっていなければ、システムは直前のディレクトリーに戻り、そのディレクトリーから始まるシンボリック・リンクのパス名をたどります。

次の例は、シンボリック・リンクの使用方法を示したものです。

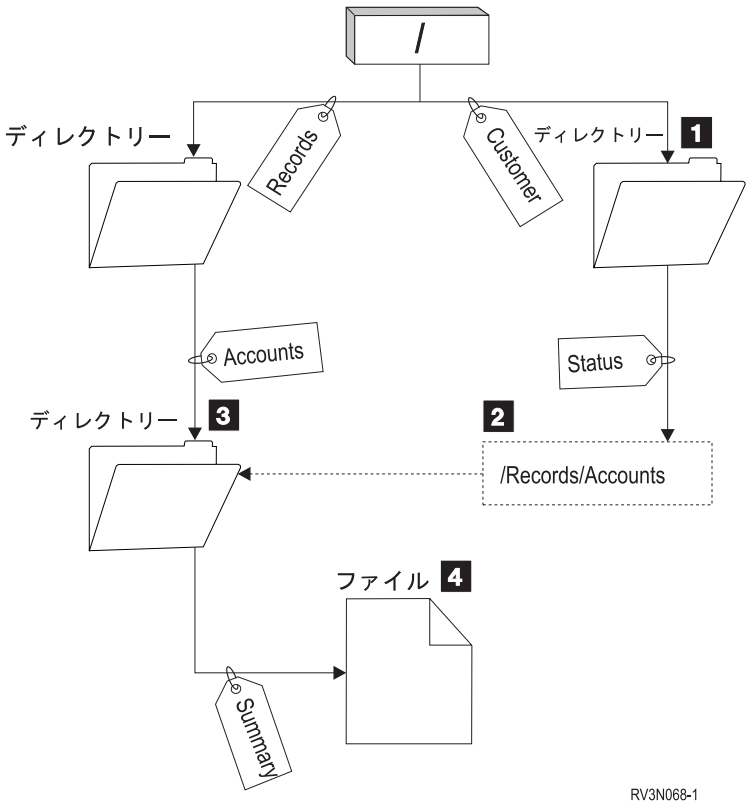


図 8. シンボリック・リンクの使用例

メニュー・オプションを選択して、顧客アカウントの状況を表示します。メニューを表示するプログラムは、次のパス名を使用します。

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

システムは、*Customer* リンクを通ってディレクトリー 1 に入り、そこから *Status* リンクに進みます。*Status* リンクは、パス名 2 を含むシンボリック・リンクです。パス名が */* で始まっているので、システムは */* (「ルート」) ディレクトリーに戻り、そこから *Records* リンクと *Accounts* リンクに進みます。このパスは、別のディレクトリー 3 につながっています。システムは、プログラムが提供するパス名に含まれるすべてのパスを通りました。 *Summary* リンクを通って、必要としているデータが入っているファイル 4 に到達します。

シンボリック・リンクは、ハード・リンクとは違って、オブジェクト (オブジェクト・タイプは *SYMLNK) であるため、リンクするオブジェクトがなくても、存在することができます。シンボリック・リンクは、あとで追加または置換されるファイルにパスを提供する場合などに使用します。

| また、シンボリック・リンクは、複数のファイル・システムにわたることができるという点でハード・リンクと異なります。たとえば、あるファイル・システムで作業しているときに、シンボリック・リンクを使用して別のファイル・システムのファイルにアクセスすることができます。 QSYS.LIB、独立 ASP QSYS.LIB、および QDLS ファイル・システムでは、シンボリック・リンクの作成および保管はサポートされていませんが、「ルート (*/*)」または QOpenSys ファイル・システムにシンボリック・リンクを作成して、以下のことを行うことができます。

- QSYS.LIB または独立 ASP QSYS.LIB ファイル・システムのデータベース・ファイル・メンバーへのアクセス。
- QDLS ファイル・システムの文書へのアクセス。

『比較: ハード・リンクとシンボリック・リンク』も参照してください。

比較: ハード・リンクとシンボリック・リンク

プログラムでパス名を使用するときには、ハード・リンクとシンボリック・リンクのどちらを使用するかを選択できます (19 ページの『リンク』を参照)。どちらのリンクにも、利点と欠点があります。以下の表では、状態ごとにリンクを比較しています。

表 1. ハード・リンクとシンボリック・リンクの比較

項目	ハード・リンク	シンボリック・リンク
ネーム・レゾリューション	速い。ハード・リンクは、オブジェクトを直接参照することができます。	遅い。シンボリック・リンクにはパス名が含まれていて、オブジェクトを検出するためにはそれを解決しなければなりません。
オブジェクトの存在	必須。オブジェクトとの間にハード・リンクを設定するためには、オブジェクトが必要です。	任意。シンボリック・リンクは、参照するオブジェクトが存在しなくても設定することができます。
オブジェクトの削除	制限あり。オブジェクトを削除するためには、オブジェクトへのハード・リンクをすべてリンク解除 (除去) しなければなりません。	制限なし。オブジェクトは、シンボリック・リンクが参照していても削除することができます。
動的オブジェクト (属性が変更される場合)	遅い。オブジェクトの属性の多くが、各ハード・リンクに保管されています。したがって、オブジェクトへのハード・リンクの数が増えるほど、動的オブジェクトへの変更は時間がかかります。	速い。動的オブジェクトへの変更は、シンボリック・リンクによる影響を受けません。
静的オブジェクト (属性が変更されない場合)	速い。静的オブジェクトについては、ネーム・レゾリューションがパフォーマンスに影響する最大の要素です。ハード・リンクを使用すると、ネーム・レゾリューションは速くなります。	遅い。シンボリック・リンクを使用すると、ネーム・レゾリューションは遅くなります。
有効範囲	制限あり。ハード・リンクは、複数のファイル・システムにわたることはできません。	制限なし。シンボリック・リンクは、複数のファイル・システムにわたることができます。

拡張属性

拡張属性 (EA) とは、オブジェクトに関連付けられる情報で、そのオブジェクトの詳細を提供するものです。 EA は、それを表す名前と値で構成されています。値は、テキスト、2 進データ、または別のタイプのデータのいずれかにすることができます。

オブジェクトの EA は、オブジェクトが存在している間だけ存在します。

EA には多くの種類があり、さまざまな情報を入れるのに使用できます。特に、次の 3 つの EA については、知っておく必要があります。

.SUBJECT

オブジェクトの内容または目的の要旨。

.TYPE オブジェクト内のデータのタイプ。データのタイプには、テキスト、バイナリー、プログラムのソース、コンパイル済みプログラム、その他の情報などがあります。

.CODEPAGE
オブジェクトで使用されるコード・ページ。オブジェクトに使用されるコード・ページは、そのオブジェクトに関連付けられている EA にも使用されます。

名前の最初のピリオド (.) は、この EA が標準システム EA (SEA) であり、システムでの使用のために予約されていることを意味します。

ファイル・システムによって、オブジェクトに EA がある場合とない場合があります。 QSYS.LIB および独立 ASP QSYS.LIB ファイル・システムは、 .SUBJECT、.TYPE、および .CODEPAGE という 3 つの事前定義 EA をサポートしています。文書ライブラリー・サービス (QDLS) ファイル・システムでは、フォルダーおよび文書には、どのような EA でも付けることができます。フォルダーおよび文書には、EA を付けても、付けなくてもかまいません。「ルート (/)」、オープン・システム (QOpenSys)、およびユーザー定義のファイル・システムでは、すべてのディレクトリー、ストリーム・ファイル、シンボリック・リンクに、どのような EA でも付けられます。しかしながら、どの EA も付けないものがあってもかまいません。

オブジェクト・リンクの処理 (WRKLNK) コマンドを使用して、オブジェクトの .SUBJECT 拡張属性 (EA) を表示することができます。統合ファイル・システムでは、アプリケーションまたはユーザーが EA へのアクセスや変更を行う他の方法はサポートされていません。この規則の例外として、 UDFS の表示 (DSPUDFS) およびマウント・ファイル・システムの情報の表示 (DSPMFSINF) の CL コマンドがあります。これらは拡張属性をユーザーに提示します。

ただし、QDLS のいくつかのオブジェクトに関連した EA については、階層ファイル・システム (HFS) が提供するインターフェースを介して変更することができます。これらのファイル・システムについての詳細は、 83 ページの『文書ライブラリー・サービス・ファイル・システム (QDLS)』および 85 ページの『光ファイル・システム (QOPT)』を参照してください。

クライアントの PC が、OS/2 または Windows により iSeries サーバーに接続されている場合には、それぞれのオペレーティング・システムのプログラミング・インターフェース (DosQueryFileInfo や DosSetFileInfo など) を使用して、任意のファイル・オブジェクトの EA の照会や設定を行うことができます。また、OS/2 ユーザーは、設定ノートブックを使用して、デスクトップでオブジェクトの EA を変更することができます (そのオブジェクトに関連するポップアップ・メニューから 「設定」を選択する)。

拡張属性を定義する際には、次の命名規則に従ってください。

- EA の名前の長さは、最大で 255 文字までです。
- 名前の最初の文字としてピリオド (.) を使用しないでください。ピリオドで始まる名前の EA は、標準システム EA と解釈されます。
- 名前の競合を最小限にするために、EA には整合性のある命名構造を使用してください。次の形式を使用することをお勧めします。

```
CompanyNameProductName.Attribute_Name
```

名前の継続性

「ルート (/)」、QOpenSys、およびユーザー定義ファイル・システムを使用する場合、オブジェクト名の文字が変更されないことを保証するシステム・サポートを利用することができます。このことは、iSeries サーバー相互間でこれらのファイル・システムを使用している場合や、異なる文字エンコード・スキーム (コード・ページ) をもつ装置に接続している場合でも適用されます。 サーバーでは、名前に使用される文字

は、 *TYPE1 ディレクトリーの場合には UCS2 レベル 1 (**Unicode** とも呼ばれる)、 *TYPE2 ディレクトリーの場合には UTF-16 という 16 ビット形式で保管されます。ディレクトリー・フォーマットについての詳細は、 『*TYPE2 ディレクトリー』を参照してください。 UCS2 レベル 1 および UTF-16 は ISO 10646 規格のサブセットです。名前が指定されると、システムは、保管されている文字の形式を、使用しているコード・ページの文字表示に変換します。各オブジェクトに関連する拡張属性の名前も、同じように処理されます。

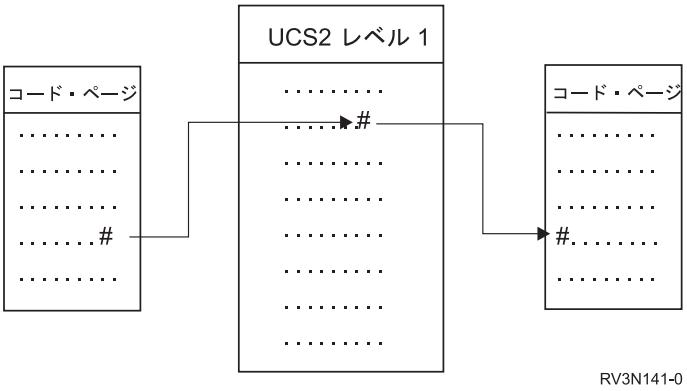


図 9. エンコード・スキーム間での同じ文字の保証

このサポートにより、異なるコード・ページを使用する装置から、サーバーへ、容易に対話することができます。たとえば、複数の PC ユーザーが、異なるコード・ページを使用するそれぞれの PC から、同じファイル名で iSeries サーバーのファイルにアクセスすることができます。あるコード・ページから別のコード・ページへの変換は、サーバーによって自動的に行われます。ただし、この装置は、名前に使用されている文字を含むコード・ページを使用しなければなりません。

第 3 章 従来のシステム・インターフェースを使用した統合ファイル・システムへのアクセス

システムのライブラリー、オブジェクト、データベース・ファイル、フォルダー、および文書での作業に使用する、メニュー、コマンド、表示画面などのユーザー・インターフェースは、すべて統合ファイル・システムの導入前と同様に操作できます。ただし、これらのインターフェースは、統合ファイル・システムがサポートするストリーム・ファイル、ディレクトリー、その他のオブジェクトを処理するために使用することはできません。

統合ファイル・システムには、これとは別にユーザー・インターフェースのセットが提供されています。これらのインターフェースは、統合ファイル・システムのディレクトリーによりアクセスできるすべてのファイル・システム内のオブジェクトで使用することができます。

メニューと表示画面を利用するか、または制御言語 (CL) コマンドを使用して、サーバーから統合ファイル・システムのディレクトリーやオブジェクトと対話できます。また、アプリケーション・プログラム・インターフェース (API) を使用して、ストリーム・ファイル、ディレクトリー、その他の統合ファイル・システムのサポートを利用できます。

さらに、Windows デスクトップからサーバーを管理および制御するために使用するグラフィカル・ユーザー・インターフェースの、 iSeries ナビゲーターを介して統合ファイル・システムと対話することもできます。

統合ファイル・システムと対話するためのいくつかの方法があります。

API の使用
統合ファイル・システムのディレクトリーおよびストリーム・ファイルの操作を実行するアプリケーション・プログラム・インターフェース (API) は、 C 言語の関数の形式になっています。

CL コマンドの使用
CL コマンドは、統合ファイル・システムを介してアクセス可能なすべてのファイル・システムのファイルや他のオブジェクトに使用することができます。

iSeries メニューおよび表示画面の使用
統合ファイル・システムでは、サーバーが提供するメニューと表示画面のセットを使用して、ファイルやオブジェクトを操作します。

iSeries ナビゲーターの使用
iSeries ナビゲーターは、Windows デスクトップからサーバーを管理および制御するためのグラフィカル・ユーザー・インターフェースです。

PC の使用
PC が iSeries サーバーに接続されている場合、統合ファイル・システムのディレクトリーおよびオブジェクトを、それらが PC に保管されているかのように扱うことができます。

iSeries メニューおよび表示画面を使用した操作の実行

統合ファイル・システムでは、サーバーが提供するメニューと表示画面のセットを使用して、ファイルやオブジェクトを操作します。統合ファイル・システムのメニューを表示するには、次のようにします。

1. サーバーにサインオンします。
2. **Enter** を押して続行します。

3. iSeries メイン・メニューから、**「ファイル、ライブラリー、およびフォルダー」**オプションを選択します。
4. 「ファイル、ライブラリー、およびフォルダー」メニューから、**「統合ファイル・システム」**オプションを選択します。

ここから必要に応じて、統合ファイル・システム内のディレクトリー・コマンド、オブジェクト・コマンド、またはセキュリティー・コマンドでの作業を行うことができます。ただし、使用する CL コマンドが事前に分かっている場合には、オプションのメニューをう回して、そのコマンドを画面下部のコマンド入力行に入力してから、**Enter** を押すことができます。

さらに、次のステップを実行することによって、サーバー上のどのメニューからでも統合ファイル・システムにアクセスすることができます。

1. 任意のコマンド行に GO DATA と入力して、「ファイル、ライブラリー、およびフォルダー」メニューを表示します。
2. 「統合ファイル・システム」というオプションを選択します。

ネットワーク・ファイル・システムのコマンドのメニューを表示するには、任意のコマンド入力行で GO CMDNFS と入力します。ユーザー定義ファイル・システムのコマンドのメニューを表示するには、任意のコマンド行で GO CMDUDFS と入力します。

統合ファイル・システム・メニューから、次の操作を行う表示画面を要求することができます。

- ディレクトリーの作成、変換、および除去
- 現行ディレクトリー名の表示および変更
- オブジェクト・リンクの追加、表示、変更、および除去
- オブジェクトのコピー、移動、および名前変更
- オブジェクトのチェックインおよびチェックアウト
- オブジェクトの保管 (バックアップ) および復元
- オブジェクト所有者およびユーザー権限の表示と変更
- ストリーム・ファイルとデータベース・ファイル・メンバーの間でのデータのコピー
- ユーザー定義ファイル・システムの作成、削除、および状況の表示
- サーバーからのファイル・システムのエクスポート
- クライアントでのファイル・システムのマウントおよびアンマウント

この中の一部の操作をサポートしていないファイル・システムもあります。特定のファイル・システムの制限事項については、4 ページの『統合ファイル・システムのファイル・システム』を参照してください。

統合ファイル・システムのメニューおよび表示画面についての詳細は、以下のトピックを参照してください。

- CL コマンドおよび表示画面のパス名規則

CL コマンドを使用した操作の実行

統合ファイル・システムのメニューおよび表示画面を使用して行える操作 (27 ページの『iSeries メニューおよび表示画面を使用した操作の実行』を参照) は、制御言語 (CL) コマンドを入力して行うことができます。これらのコマンドは、統合ファイル・システム・インターフェースを介してアクセス可能なすべてのファイル・システムのファイルや他のオブジェクトに使用することができます。

表 1 は、統合ファイル・システム・コマンドの要約です。ユーザー定義のファイル・システム、ネットワーク・ファイル・システム、およびマウントされている一般的なファイル・システムに特に関連した CL コマンドについての詳細は、 72 ページの『ユーザー定義ファイル・システム (UDFS)』および 97 ページの『ネットワーク・ファイル・システム (NFS)』を参照してください。 OS/2 コマンドまたは DOS コマンドと同様の操作を行うコマンドについては、 OS/2 と DOS のユーザーがわかりやすいように、別名 (代替コマンド名) を示しています。

表 2. 統合ファイル・システム・コマンド

コマンド	説明	別名
ADDLNK	リンクの追加。ディレクトリーとオブジェクトの間にリンクを追加する。	
ADDMFS	マウント・ファイル・システムの追加。エクスポートしたリモート・サーバー・ファイル・システムを、ローカル・クライアント・ディレクトリーに入れる。	MOUNT
APYJRNCHG[2]	ジャーナルされた変更の適用。ジャーナル項目を使用して、ジャーナル済みオブジェクトが保管されてから生じた変更を適用したり、特定の時点までの変更を適用する。	
CHGATR	属性の変更。ディレクトリー、その内容、およびそのすべてのサブディレクトリーの内容に変更された属性があるディレクトリー・ツリー、単一のオブジェクト、およびオブジェクトのグループの属性を変更する。	
CHGAUD	監査値の変更。オブジェクトの監査をオンまたはオフに切り替える。	
CHGAUT	権限の変更。ユーザーまたはユーザー・グループにオブジェクトに対する特定の権限を与える。	
CHGCURDIR	現行ディレクトリーの変更。現行ディレクトリーとして使用するディレクトリーを変更する。	CD、CHDIR
CHGNFSEXP	ネットワーク・ファイル・システムのエクスポートの変更。 NFS クライアントにエクスポートされたエクスポート・テーブルに対してディレクトリー・ツリーの追加や除去を行う。	EXPORTFS
CHGOWN	所有者の変更。オブジェクトの所有権を別のユーザーに移す。	
CHGPGP	1 次グループの変更。 1 次グループを別のユーザーに変更する。	
CHKIN	チェックイン。以前にチェックアウトされたオブジェクトをチェックインする。	
CHKOUT	チェックアウト。他のユーザーがオブジェクトを変更しないように、オブジェクトをチェックアウトする。	
CPY	コピー。 1 つのオブジェクトまたはオブジェクトのグループをコピーする。	COPY
CPYFRMSTMF	ストリーム・ファイルからのコピー。ストリーム・ファイルからデータベース・ファイル・メンバーにデータをコピーする。	
CPYTOSTMF	ストリーム・ファイルへのコピー。データベース・ファイル・メンバーからストリーム・ファイルにデータをコピーする。	
CRTDIR	ディレクトリーの作成。システムに新しいディレクトリーを追加する。	MD、MKDIR
CRTUDFS	UDFS の作成。ユーザー定義のファイル・システムを作成する。	

表 2. 統合ファイル・システム・コマンド (続き)

コマンド	説明	別名
CVTDIR	ディレクトリーの変換。 *TYPE1 フォーマットから *TYPE2 フォーマットへの統合ファイル・システム・ディレクトリーの変換に関する情報を提供するか、または変換を実行する。	
CVTRPCSRC	RPC ソースの変換。入力ファイルから C コードをリモート・プロシージャー・コール (RPC) 言語で生成する。	RPCGEN
DLTUDFS	UDFS の削除。ユーザー定義のファイルを削除する。	
DSPAUT	権限の表示。オブジェクトの許可ユーザーおよびその所有しているオブジェクト権限のリストを表示する。	
DSPCURDIR	現行ディレクトリーの表示。現行ディレクトリーの名前を表示する。	
DSPLNK	オブジェクト・リンクの表示。ディレクトリー内のオブジェクトのリストを表示して、オブジェクトの情報を表示するオプションを提供する。	
DSPF	ストリーム・ファイルの表示。ストリーム・ファイルまたはデータベース・ファイルを表示する。	
DSPMFSINF	マウント・ファイル・システムの情報の表示。マウントされているファイル・システムに関する情報を表示する。	STATFS
DSPUDFS	UDFS の表示。ユーザー定義のファイル・システムを表示する。	
EDTF	ストリーム・ファイルの編集。ストリーム・ファイルまたはデータベース・ファイルを編集する。	
ENDJRN[2]	ジャーナルの終了。オブジェクトまたはオブジェクトのリストの変更に関するジャーナル処理を終了する。	
ENDNFSSVR	ネットワーク・ファイル・システム・サーバーの終了。サーバーおよびクライアント上の 1 つまたはすべての NFS デーモンを終了する。	
ENDRPCBIND	RPC バインド・プログラム・デーモンの終了。リモート・プロシージャー・コール (RPC) RPCBind デーモンを終了する。	
MOV	移動。オブジェクトを別のディレクトリーに移動させる。	MOVE
RLSIFSLCK	統合ファイル・システムのロック解除。クライアントで保留された、またはオブジェクトに関する NFS バイト範囲のロックをすべて解除する。	
RMVDIR	ディレクトリーの除去。システムからディレクトリーを除去する。	RD、RMDIR
RMVLNK	リンクの除去。オブジェクトへのリンクを除去する。	DEL、ERASE
RMVMFS	マウント・ファイル・システムの除去。エクスポートしたリモート・サーバー・ファイル・システムを、ローカル・クライアント・ディレクトリーから除去する。	UNMOUNT
RNM	名前変更。ディレクトリー内のオブジェクトの名前を変更する。	REN
RPCBIND	RPC バインド・プログラム・デーモンの開始。リモート・プロシージャー・コール (RPC) RPCBind デーモンを開始する。	
RST	復元。オブジェクトまたはオブジェクトのグループを、バックアップ装置からシステムにコピーする。	
RTVCURDIR	現行ディレクトリーの検索。現行ディレクトリーの名前を検索し、指定された変数に入れる (CL プログラムで使用)。	

表 2. 統合ファイル・システム・コマンド (続き)

コマンド	説明	別名
SAV	保管。オブジェクトまたはオブジェクトのグループを、システムからバックアップ装置にコピーする。	
SNDJRNE[2]	ジャーナル項目の送信。ユーザー・ジャーナル項目を、オプションでジャーナル・オブジェクトと関連付けて、ジャーナル・レシーバーに追加する。	
STRJRN[2]	ジャーナルの開始。特定のジャーナルへの (オブジェクトまたはオブジェクトのリストに加えた) ジャーナル処理の変更を開始する。	
STRNFSSVR	ネットワーク・ファイル・システム・サーバーの開始。サーバーおよびクライアント上の 1 つまたはすべての NFS デーモンを開始する。	
WRKAUT	権限の処理。ユーザーとその権限のリストを表示し、ユーザーの追加、ユーザー権限の変更、ユーザーの削除などを行うオプションを提供する。	
WRKLNK	オブジェクト・リンクの処理。ディレクトリー内のオブジェクトのリストを表示して、オブジェクトに処置を行うオプションを提供する。	
WRKOBJOWN[1]	所有者によるオブジェクトの処理。ユーザー・プロファイルが所有するオブジェクトのリストを表示し、オブジェクトに処置を行うオプションを提供する。	
WRKOBJPGP[1]	1 次グループによるオブジェクトの処理。 1 次グループが制御するオブジェクトのリストを表示し、オブジェクトに処置を行うオプションを提供する。	

注:

1. WRKOBJOWN コマンドおよび WRKOBJPGP コマンドは、すべてのオブジェクト・タイプを表示しますが、すべてのファイル・システムで機能するというわけではありません。
2. 詳細については、iSeries Information Center の『ジャーナル管理』を参照してください。

| 統合ファイル・システムの CL コマンドと、特定のファイル・システムでのこれらのコマンドの使用における制限事項の詳細は、以下のトピックを参照してください。

| • 統合ファイル・システムのファイル・システム
| • CL コマンドおよび表示画面のパス名規則
| • iSeries Information Center 内の CL に関するトピック

CL コマンドおよび表示画面のパス名規則

統合ファイル・システムのコマンドまたは表示画面を使用してオブジェクトを操作するときは、パス名を指定してオブジェクトを識別します。パス名を指定する際の規則の要約を、以下に示します。これらの規則の中で、**オブジェクト**という用語は、任意のディレクトリー、ファイル、リンク、その他のオブジェクトを表します。

- オブジェクトは、各ディレクトリー内で固有でなければなりません。
- 統合ファイル・システムの CL コマンドに渡されるパス名は、現在ジョブの CCSID で表さなければなりません。ジョブの CCSID が 65535 の場合は、パス名はそのジョブのデフォルトの CCSID で表さな

ければなりません。テキスト・ストリングは通常 CCSID 37 でエンコードされているため、パスをコマンドに渡す前に、ハードコーディングされたパス名をジョブ CCSID に変換する必要があります。

- コマンド行にパス名を入力するときには、アポストロフィ (') で囲んでください。アポストロフィは、表示画面にパス名を入力するときは、任意指定です。ただし、引用符 (") で囲まれたストリングがパス名に含まれる場合には、アポストロフィ (' ') で囲まなければなりません。
- パス名は、最高レベルのディレクトリーから始まって、そのコマンドで操作するオブジェクトの名前で終るように、左から右へ入力します。パス名の各構成要素は、スラッシュ (/) または円記号 (¥) で区切ります。たとえば、次のようになります。

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

  または

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

- /、¥、およびヌル文字は、パス名の構成要素として使用することはできません (/ および ¥ は区切り文字として使用されるため)。コマンドによって、小文字が大文字に変更されることはありません。名前が大文字に変更されるかどうかは、オブジェクトが含まれるファイル・システムが大文字小文字の区別を行うかどうか、およびオブジェクトが作成されるのか検索されるのかによって異なります。
- オブジェクト名の長さは、オブジェクトが含まれるファイル・システムと、コマンド・ストリングの最大長に制限されます。コマンドが受け入れるオブジェクト名は 255 文字まで、パス名は 5000 文字までです。

  各ファイル・システムでのパス名の制限事項については、 『統合ファイル・システムのファイル・システム』を参照してください。
- 次のように、パス名の先頭が / または ¥ である場合は、パスが最上位のディレクトリー (「ルート (/)」ディレクトリー) から始まることを示します。

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

- 次のように、パス名の先頭が / または ¥ でない場合、コマンドを入力するユーザーの現行ディレクトリーからパスが始まるものと見なされます。

  ```
  'MyDir/MyFile'
  ```

  MyDir は、ユーザーの現行ディレクトリーのサブディレクトリーです。
- 次のように、波形記号 (~) にスラッシュ (または円記号) が続く形でパス名が始まっている場合は、コマンドを入力するユーザーのホーム・ディレクトリーからパスが始まるものと見なされます。

  ```
  '~/UsrDir/UsrObj'
  ```

- 次のように、波形記号 (~) のあとにユーザー名が、そのあとにスラッシュが続いている場合は、そのユーザー名で識別されるユーザーのホーム・ディレクトリーからパスが始まるものと見なされます。

  ```
  '~user-name/UsrDir/UsrObj'
  ```

- コマンドの中には、パス名の最後の構成要素としてアスタリスク (*) または疑問符 (?) を使用して、名前のパターンを検索できるものもあります。 * は、* の位置に任意の文字 (何文字でもよい) がある名前を検索することを、システムに指定します。 ? は、? の位置に 1 つの文字がある名前を検索することを、システムに指定します。次の例は、 *d* で始まって *txt* で終わる名前のオブジェクトを検索します。

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

  次の例は、*d* で始まり、任意の 1 文字が入って *txt* で終わる名前のオブジェクトを検索することを指定します。

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

- iSeries サーバーの特殊値と混同しないようにするため、パス名の先頭を単一アスタリスク (*) 文字にすることはできなくなっています。パス名の先頭でパターン照合を行うには、アスタリスクを 2 つ使用してください。たとえば、次のようになります。

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

  注: これは、アスタリスク (*) の前に他の文字がない相対パス名だけに適用されます。
- QSYS.LIB ファイル・システム内のオブジェクトを操作する場合、構成要素名は、 *name.object-type* の形式になっていなければなりません。

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

  詳細については、77 ページの『ライブラリー・ファイル・システム (QSYS.LIB)』を参照してください。
- 独立 ASP QSYS.LIB ファイル・システム内のオブジェクトを操作する場合、構成要素名は、 *name.object-type* の形式になっていなければなりません。たとえば、以下のようになります。

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

  詳細については、80 ページの『独立 ASP QSYS.LIB』を参照してください。
- 構成要素名に以下のいずれかの文字が使用されている場合には、パス名をアポストロフィ (') または引用符 (") で囲まなければなりません。
  - アスタリスク (*)
  - 疑問符 (?)
  - アポストロフィ (')
  - 引用符 (")
  - 波形記号 (~)。ただし、パス名の最初の構成要素名の 1 文字目として使用されている場合 (その他の位置で使用されていれば、別の文字として解釈されます)。

  次はその一例です。

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

  または

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

  このようなパス名の使用は、コマンド・ストリングの文字の意味が混乱し、コマンド・ストリングを誤って入力する可能性を高めるため、お勧めしません。
- パス名の中でコロン (:) を使用しないでください。コロンはシステムで特殊な意味をもちます。
- コマンドおよび関連するユーザー表示画面の処理サポートでは、 16 進数で 40 未満のコード・ポイントは、コマンド・ストリング内または表示画面で使用できる文字として認識されません。これらのコード・ポイントを使用する場合は、以下のように 16 進表記として入力しなければなりません。

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

  そのため、パス名に 16 進数で 40 未満のコード・ポイントを使用することはお勧めしません。この制限が該当するのはコマンドおよび関連する表示画面だけで、 API には当てはまりません (54 ページの『API を使用した操作の実行』を参照してください)。

特定のコマンドの使用に関する制限事項については、コマンド・ヘルプまたは iSeries Information Center の『制御言語 (CL)』のトピックを参照してください。

PC を使用した操作の実行

PC が iSeries サーバーに接続されている場合、統合ファイル・システムのディレクトリーおよびオブジェクトを、それらが PC に保管されているかのように扱うことができます。 Windows のエクスプローラのドラッグ・アンド・ドロップ機能を使用して、ディレクトリー間でオブジェクトをコピーできます。オブジェクトを物理的にサーバーから PC にコピーする必要がある場合には、サーバー・ドライブ内のオブジェクトを選択して、そのオブジェクトを PC ドライブにドラッグします。

Windows インターフェースを使用して iSeries サーバーと PC の間でコピーされるオブジェクトを、EBCDIC と ASCII の間で自動的に変換することもできます。 EBCDIC は拡張 2 進化 10 進コードのことで、 ASCII は情報交換用米国標準コードのことです。この変換を自動的に実行するよう、iSeries Access を構成できます。また、特定の拡張子のファイルに対して、この変換が実行されるように指定することもできます。 OS/400 バージョン 4 リリース 4 (V4R4) では、ファイルに関する変換を実行するよう iSeries ネットサーバーを構成することもできます。

オブジェクトのタイプによっては、 PC インターフェースや PC アプリケーションを使用して、この作業を行うこともできます。たとえば、テキストを含むストリーム・ファイルを、 PC のエディターで編集することができます。

PC を使用して iSeries サーバーに接続すると、統合ファイル・システムにより、サーバーのディレクトリーやオブジェクトが PC で使用できるようになります。 PC からは、Windows オペレーティング・システムに組み込まれているファイル共用クライアント、 FTP クライアント、または iSeries ナビゲーター (iSeries Access の一部) を使用することにより、統合ファイル・システム内のファイルを処理することができます。 PC は、Windows ファイル共用クライアントを使用して、iSeries サーバー上で稼働する iSeries ネットサーバーにアクセスします。

FTP を使用したファイルの転送
FTP クライアントを使用すると、「ルート (/)」、QSYS.LIB、独立 ASP QSYS.LIB、QOpenSys、QOPT、および QFileSvr.400 ファイル・システム内のファイルを含め、 iSeries サーバー上にあるファイルを転送することができます。また、FTP クライアントを使用すると、文書ライブラリー・サービス (QDLS) ファイル・システム内のフォルダーおよび文書を転送することもできます。

iSeries ナビゲーターを使用したファイルの処理
iSeries Access には、iSeries ナビゲーターが含まれています。 iSeries ナビゲーターは iSeries サーバーに接続し、PC が統合ファイル・システムを使用できるようにします。 iSeries ナビゲーターは、Windows デスクトップから iSeries サーバーを管理および制御するためのグラフィカル・ユーザー・インターフェースです。

iSeries ネットサーバーを使用したファイルの処理
iSeries ネットサーバーは OS/400 の一部で、これを使用すると Windows クライアントに組み込まれているファイルおよび印刷共用機能をサーバーと一緒に実行することができます。

注: 新しいバージョンの iSeries Access では、統合ファイル・システムにアクセスするのに NetServer に完全に依存しています。 NetServer サポートは、OS/400 V4R2 以上を実行している iSeries サーバーに対する TCP/IP 接続のみで使用できます。

FTP を使用したファイルの転送

ファイル転送プロトコル (FTP) クライアントを使用すると、「ルート (/)」、QOpenSys、QSYS.LIB、独立 ASP QSYS.LIB、QOPT、および QFileSvr.400 ファイル・システム内のファイルを含め、 iSeries サーバー上にあるファイルを転送することができます。また、FTP クライアントを使用すると、文書ライブラリー・サービス (QDLS) ファイル・システム内のフォルダーおよび文書を転送することもできます。 FTP クライアントは、クライアント・サブコマンドがファイルから読み取られ、これらのサブコマンドへの応答が

ファイルに書き込まれる、不在バッチ・モードで対話式に実行されることがあります。また、FTP クライアントには、サーバー上でファイルを取り扱うためのその他の機能が含まれています。

FTP サポートを使用して、以下のいずれかのファイル・システムとの間でファイルを転送できます。

- 「ルート (/)」ファイル・システム
- オープン・システム・ファイル・システム (QOpenSys)
- ライブラリー・ファイル・システム (QSYS.LIB)
- 独立 ASP QSYS.LIB ファイル・システム
- 文書ライブラリー・サービス・ファイル・システム (QDLS)
- 光ファイル・システム (QOPT)
- ネットワーク・ファイル・システム (NFS)
- NetWare ファイル・システム (QNetWare)
- Windows NT Server ファイル・システム (QNTC)

ただし、以下の制限事項に注意してください。

- 統合ファイル・システムでは、FTP サポートはファイル・データの転送のみに制限されます。 FTP を使用して属性データを転送することはできません。
- QSYS.LIB および独立 ASP QSYS.LIB ファイル・システムでは、 FTP サポートは物理ファイル・メンバー、ソース物理ファイル・メンバー、および保管ファイルに制限されます。 FTP を使用して他のオブジェクト・タイプ (プログラム (*PGM) など) を転送することはできません。しかし、他のオブジェクト・タイプを保管ファイルに保管し、その保管ファイルを転送してから、そのオブジェクトを復元することができます。

FTP については、iSeries Information Center の**『ネットワーキング』**カテゴリー内の以下のトピックを参照してください。

- FTP
- FTP を使用したファイルの転送

iSeries ネットサーバーを使用したファイルの処理

iSeries Support for Windows Network Neighborhood (iSeries ネットサーバー) は、 Windows クライアントが OS/400 共用ディレクトリー・パスおよび共用出力キューにアクセスできるようにする IBM Operating System/400 バージョン 5 (OS/400) の機能です。 iSeries ネットサーバーを使用すると、PC は、 iSeries によって管理されているデータおよびプリンターにシームレスにアクセスするために Windows ソフトウェアを実行することができます。ネットワーク上の PC クライアントは、単に、オペレーティング・システムに含まれているファイルおよび印刷共用機能を使用します。これは、iSeries ネットサーバーを使用するために PC 上に追加ソフトウェアをインストールする必要がないことを意味しています。

Samba クライアント・ソフトウェアがインストールされている LINUX クライアントは、 iSeries ネットサーバーを介してデータおよびプリンターにシームレスにアクセスすることもできます。 Samba ファイル・システム (smbfs) は、 iSeries から NFS ファイル・システムをマウントするのと類似の方法で iSeries ネットサーバーからマウントできます。詳しくは、iSeries Information Center の『iSeries ネットサーバー』のトピックを参照してください。

iSeries ネットサーバー・ファイル共用は、 iSeries ネットサーバーが iSeries ネットワーク上のクライアントと共用するディレクトリー・パスです。ファイル共用は、 iSeries 上の任意の統合ファイル・システム・ディレクトリーで構成することができます。 iSeries ネットサーバーを使用してファイル共用を処理する前

に、 iSeries ネットサーバー・ファイル共用を作成する必要があります。また、必要であれば、iSeries ナビゲーターを使用して iSeries ネットサーバー・ファイル共用を変更します。

iSeries ネットサーバーを使用して統合ファイル・システム・ファイル共用にアクセスするには、以下のようにします。

1. **「スタート」**を右マウス・ボタンでクリックし、**「エクスプローラ」**を選択し、Windows PC 上で Windows Explorer を開きます。
2. **「ツール」**メニューをオープンし、 **「ネットワーク・ドライブを共用にマップする」**を選択します。
3. ファイル共用のための空きドライブ (I:¥ ドライブなど) の文字を選択します。
4. iSeries ネットサーバー・ファイル共用の名前を入力します。たとえば、以下の構文を入力することができます。 **¥¥QSYSTEM1¥Sharename**

   注: QSYSTEM1 は iSeries ネットサーバーのシステム名で、 Sharename は使用するファイル共用の名前です。
5. 「OK」をクリックします。

注: iSeries ネットサーバーを使用して接続するとき、サーバー名は、iSeries Access によって使用される名前とは異なる場合があります。たとえば、iSeries ネットサーバー名が QAS400X だとすると、ファイルを処理するためのパスは ¥¥QAS400X¥QDLS¥MYFOLDER.FLR¥MYFILE.DOC になります。しかし、iSeries Access 名が AS400X だとすると、ファイルを処理するためのパスは ¥¥AS400X¥QDLS¥MYFOLDER.FLR¥MYFILE.DOC になります。

iSeries ネットサーバーを使用したネットワークと共用するディレクトリーを選択します。この種のディレクトリーは、サーバー名の下の 第 1 レベルとして表されます。たとえば、/home/fred ディレクトリーを名前 fredsdir と共用すると、ユーザーは、¥¥QAS400X¥FREDSDIR という名前で PC から、または //qas400x/fredsdir という名前で LINUX クライアントからそのディレクトリーにアクセスすることができます。

PC ファイルのサービスについては、「ルート (/)」ファイル・システムの方が、他の iSeries ファイル・システムよりパフォーマンスが良くなります。「ルート (/)」ファイル・システムにファイルを移動したい場合もあるでしょう。詳細については、『別のファイル・システムへオブジェクトを移動させる際の考慮事項』を参照してください。

iSeries ネットサーバーおよびファイル共用の詳細については、 iSeries Information Center の**『ネットワーキング』**カテゴリー内の以下のトピックを参照してください。

- iSeries ネットサーバー
- iSeries ネットサーバー・ファイル共用
- Windows PC による iSeries ネットサーバー・ファイル共用へのアクセス

別のファイル・システムへのオブジェクトの移動

統合ファイル・システムを使用して異なるファイル・システム間でオブジェクトを移動させる前に、 37 ページの『別のファイル・システムへオブジェクトを移動させる際の考慮事項』を参照してください。

オブジェクトを移動させるには、以下のステップを実行してください。

1. 移動させるすべてのオブジェクトのコピーを保管します。

バックアップ・コピーを保管しておけば、移動先のファイル・システムで、アプリケーションがオブジェクトにアクセスできない場合に、元のファイル・システムでオブジェクトを復元することができます。

注: あるファイル・システムに保管したオブジェクトを、別のファイル・システムで復元することはできません。

2. ディレクトリーの作成 (CRTDIR) コマンドを使用して、オブジェクトの移動先のファイル・システムにディレクトリーを作成します。

   オブジェクトが現在保管されているディレクトリーの属性を詳しく調べて、作成するディレクトリーに、それらの属性をコピーするかどうか決めます。たとえば、ディレクトリーの所有者は作成者であり、元のディレクトリーの所有者ではありません。ファイル・システムが、ディレクトリーの所有者の設定をサポートしていれば、ディレクトリーを作成したあとで、その所有権を移すことができます。

3. 移動 (MOV) コマンドを使用して、選択したファイル・システムにファイルを移します。

   MOV の使用をお勧めする理由は、ファイル・システムがオブジェクトの所有者の設定をサポートしている場合、オブジェクトの所有者が変わらないためです。ただし、コピー (CPY) コマンドに OWNER(*KEEP) パラメーターを指定して使用することによっても、オブジェクトの所有者を保持することができます。この方法が有効なのは、ファイル・システムがオブジェクトの所有者の設定をサポートしている場合だけであることに注意してください。 MOV または CPY を使用する場合には、次のことに注意してください。

   - 属性が一致せずに、廃棄されることがあります。
   - 拡張属性が廃棄されることがあります。
   - 権限が同等でなく、廃棄されることがあります。

   このため、オブジェクトを元のファイル・システムに戻そうとしても、単に移動またはコピーしただけでは、属性や権限が廃棄された状態になっていることがあります。オブジェクトを戻す方法としては、保管したバックアップから復元するのが最も確実です。

別のファイル・システムへオブジェクトを移動させる際の考慮事項

各ファイル・システムには、それぞれに固有の特性があります。ただし、別のファイル・システムにオブジェクトを移動させると、そのオブジェクトが現在保管されているファイル・システムの特性を利用することはできなくなります。別のファイル・システムにオブジェクトを移動させて、その特性を利用したい場合もあるでしょう。オブジェクトを別のファイル・システムに移動させる場合、その前に統合ファイル・システム上のファイル・システムとその特性に精通している必要があります。詳しくは、4 ページの『統合ファイル・システムのファイル・システム』を参照してください。

また、以下の点も考慮する必要があります。

- 使用しているアプリケーションが、オブジェクトが現在保管されているファイル・システムを利用するものであるかどうか。

  ファイル・システムの中には、統合ファイル・システムがサポートしていないインターフェースをサポートするものもあります。これらのインターフェースを使用するアプリケーションは、別のファイル・システムに移されたオブジェクトにアクセスすることはできません。たとえば、QDLS ファイル・システムと QOPT ファイル・システムは、文書およびフォルダーのオブジェクトを処理する階層ファイル・システム (HFS) の API とコマンドをサポートしています。これらのインターフェースは、別のファイル・システムのオブジェクトで使用することはできません。

- オブジェクトの特性の中で何が最も重要か。

どのファイル・システムでも、すべての特性がサポートされているというわけではありません。たとえば、QSYS.LIB または独立 ASP QSYS.LIB ファイル・システムでは、いくつかの拡張属性の保管と検索のみがサポートされていますが、「ルート」(/) および QOpenSys ファイル・システムでは、すべての拡張属性の保管と検索がサポートされています。したがって、QSYS.LIB および 独立 ASP QSYS.LIB は、拡張属性を持つオブジェクトの保管には適していないということになります。 QDLS は多くの『オフィス』 属性をサポートしていますが、他のファイル・システムはサポートしていません。したがって、オフィス文書の保管には、QDLS が適しているということになります。

QDLS 内に格納されている PC ファイルは、移動に適しています。多くの PC アプリケーションは、QDLS から別のファイル・システムに移動された PC ファイルで作業を続けることができます。これらの PC ファイルを保管するには、「ルート (/)」、QOpenSys、QNetWare、および QNTC ファイル・システムを選択することをお勧めします。これらのファイル・システムは多数の OS/2 ファイル・システムの特性をサポートしているので、ファイルに高速アクセスできます。

統合ファイル・システムで提供されるディレクトリー

システムを再始動する際に、以下のディレクトリーがないと、統合ファイル・システムにより作成されます。

/tmp /tmp ディレクトリーは、アプリケーションが一時ファイルを保管する場所になります。このディレクトリーは、『root』 (/) ディレクトリーのサブディレクトリーなので、パス名は /tmp です。

アプリケーションによってファイルが /tmp ディレクトリーに入れられると、ユーザーかアプリケーションにより除去されない限りそのディレクトリー内にとどまります。システムは、自動的に /tmp からファイルを除去したり、 /tmp 内のファイルに対する特別な処理を実行したりしません。

/tmp ディレクトリーおよびこのディレクトリー内のファイルを管理するために、統合ファイル・システムをサポートする、ユーザー表示画面やコマンドを使用できます。たとえば、「オブジェクト・リンクの処理」画面または WRKLNK コマンドを使用すれば、 /tmp ディレクトリーまたはこのディレクトリー内のファイルをコピーしたり、除去したり、名前変更したりできます。すべてのユーザーにはディレクトリーに対する *ALL 権限があり、このディレクトリーに対して有効なアクションの大部分を実行することができます。

アプリケーションは、統合ファイル・システムをサポートするアプリケーション・プログラム・インターフェース (API) を使用して、 /tmp とその中のファイルを管理できます (54 ページの『API を使用した操作の実行』を参照)。たとえば、アプリケーション・プログラムで unlink() API を使用すると、 /tmp 内のファイルを除去することができます。

/tmp は、除去されても、システムの次の再始動時に自動的に再作成されます。

/home システム管理者は /home ディレクトリーを使用して、ユーザーごとに別々のディレクトリーを保管します。 システム管理者は、ユーザー・プロファイルに関連したホーム・ディレクトリーが、/home 内のユーザー・ディレクトリーになるよう設定することがよくあります。たとえば、/home/john のように設定します。詳細については、 9 ページの『現行のディレクトリーとホーム・ディレクトリー』を参照してください。

/etc /etc ディレクトリーは管理ファイル、構成ファイル、その他のシステム・ファイルを保管します。

/usr /usr ディレクトリーには、システムで使用される情報の入ったサブディレクトリーがあります。/usr 中のファイルは、通常は頻繁に変更が加えられることはありません。

/usr/bin

/usr/bin ディレクトリーには、標準的なユーティリティー・プログラムが入ります。

/QIBM /QIBM ディレクトリーとは、システム・ディレクトリーのことで、システムとともに提供されます。

/QIBM/ProdData
/QIBM/ProdData ディレクトリーは、ライセンス・プログラムの製品データのために使用されるシステム・ディレクトリーです。

/QIBM/UserData
/QIBM/UserData ディレクトリーは、構成ファイルなど、ライセンス・プログラムのユーザー・データのために使用されるシステム・ディレクトリーです。

/QOpenSys/QIBM
/QOpenSys/QIBM ディレクトリーは、 QOpenSys ファイル・システムのためのシステム・ディレクトリーです。

/QOpenSys/QIBM/ProdData
/QOpenSys/QIBM/ProdData ディレクトリーは、 QOpenSys ファイル・システムのためのシステム・ディレクトリーで、ライセンス・プログラムの製品データのために使用されます。

/QOpenSys/QIBM/UserData
/QOpenSys/QIBM/UserData ディレクトリーは、 QOpenSys ファイル・システムのためのシステム・ディレクトリーで、構成ファイルなど、ライセンス・プログラムのユーザー・データのために使用されます。

/asp_name/QIBM
/asp_name/QIBM ディレクトリーは、システム上に存在する独立 ASP のためのシステム・ディレクトリーで、 asp_name は独立 ASP の名前です。

/asp_name/QIBM/UserData
/asp_name/QIBM/UserData ディレクトリーは、システム上に存在する独立 ASP 用の構成ファイルなど、ライセンス・プログラムのユーザー・データのために使用されるシステム・ディレクトリーで、 asp_name は独立 ASP の名前です。

第 4 章 iSeries ナビゲーターを使用した統合ファイル・システムへのアクセス

iSeries ナビゲーターは、Windows デスクトップからシステムを管理および制御するためのグラフィカル・ユーザー・インターフェースです。 iSeries ナビゲーターによって、システムの運用および管理はより簡単でより生産的なものになります。 たとえば、1 つの iSeries サーバーから別の iSeries サーバーにドラッグするだけで、ユーザー・プロファイルを別のシステムにコピーすることができます。セキュリティー・サービス、TCP/IP サービス、およびアプリケーションのセットアップを手引きするウィザードが備えられています。

iSeries ナビゲーターを使用して、数多くのタスクを実行できます。以下にリストしているのは、手始めに役立ついくつかの共通ファイル・システム・タスクです。

ファイルおよびフォルダーの処理

- 44 ページの『フォルダーの作成』
- 44 ページの『フォルダーの除去』
- 『ファイルのチェックイン』
- 42 ページの『ファイルのチェックアウト』
- 42 ページの『ファイルまたはフォルダーに対する許可のセットアップ』
- 42 ページの『ファイル・テキスト変換のセットアップ』
- 43 ページの『他のシステムへのファイルまたはフォルダーの送信』
- 43 ページの『パッケージ定義のオプションの変更』
- 44 ページの『ファイルまたはフォルダーの送信日時のスケジュール』

ファイル共用の処理

- 45 ページの『ファイル共用の作成』
- 45 ページの『ファイル共用の変更』

ユーザー定義ファイル・システムの処理

- 45 ページの『新規のユーザー定義ファイル・システムの作成』
- 46 ページの『ユーザー定義ファイル・システムのマウント』
- 46 ページの『ユーザー定義ファイル・システムのアンマウント』

オブジェクトのジャーナル処理

- 47 ページの『ジャーナル処理の開始』
- 47 ページの『ジャーナル処理の終了』

ファイルのチェックイン

ファイルをチェックインするには、次のようにします。

1. **「iSeries ナビゲーター」**で、チェックインしたいファイルを右クリックします。
2. **「プロパティー」**を選択します。
3. **「ファイルのプロパティー」->「使用」**ページを選択します。

4. **「チェックイン」**をクリックします。

ファイルのチェックアウト

ファイルをチェックアウトするには、次のようにします。

1. **「iSeries ナビゲーター」**で、チェックアウトしたいファイルを右クリックします。
2. **「プロパティー」**を選択します。
3. **「ファイルのプロパティー」->「使用」**ページを選択します。
4. **「チェックアウト」**を選択します。

ファイルまたはフォルダーに対する許可のセットアップ

オブジェクトに対する許可を追加することによって、他のユーザーがそのオブジェクトを操作する機能を制御することができます。さまざまな許可によって、あるユーザーにはオブジェクトの表示だけを許可し、別のユーザーにはオブジェクトの実際の編集を許可することができます。

ファイルまたはフォルダーに対する許可を設定するには、次のようにします。

1. **「iSeries ナビゲーター」**ウィンドウで、使用したいシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. **「統合ファイル・システム」**を展開します。許可を追加したいオブジェクトが表示されるまで、展開を続けます。
4. 許可を追加したいオブジェクトを右クリックして、**「許可」**を選択します。
5. **「許可」**ダイアログで**「追加」**をクリックします。
6. 1 つ以上のユーザーおよびグループを選択するか、または**「追加」**ダイアログ内のユーザーまたはグループ名フィールドにユーザーまたはグループの名前を入力します。
7. **「OK」**をクリックします。これにより、そのユーザーまたはグループがリストの最上部に追加されます。
8. 詳細な許可を実装するには、**「詳細」**ボタンをクリックします。
9. 該当するチェック・ボックスのボックスにチェックを入れることにより、必要な許可をユーザーに適用します。
10. **「OK」**をクリックします。

ファイル・テキスト変換のセットアップ

| iSeries ナビゲーターで自動テキスト・ファイル変換をセットアップすることができます。自動テキスト・
| ファイル変換により、ファイル・データ変換にファイル拡張子を使用できるようになります。統合ファイ
| ル・システムは、iSeries と PC との間で転送されるときにデータ・ファイルを変換することができます。
| PC からデータ・ファイルにアクセスするとき、データ・ファイルは ASCII であるかのように処理されま
| す。

ファイル・テキスト変換をセットアップするには、次のようにします。

1. **「iSeries ナビゲーター」**で、使用したいシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. **「統合ファイル・システム」**を右クリックして、**「プロパティー」**を選択します。

4. 自動的に変換したいファイル拡張子を、**「自動テキスト・ファイル変換を行うファイル拡張子」**テキスト・ボックスに入力して、**「追加」**をクリックします。
5. 自動的に変換したいすべてのファイル拡張子について、ステップ 4 を繰り返します。
6. **「OK」**をクリックします。

他のシステムへのファイルまたはフォルダーの送信

ファイルまたはフォルダーを他のシステムに送信するには、次のようにします。

1. **「iSeries ナビゲーター」**で、使用したいシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. **「統合ファイル・システム」**を展開します。送信したいファイルまたはフォルダーが表示されるまで、展開を続けます。
4. そのファイルまたはフォルダーを右クリックして、**「送信」**を選択します。そのファイルまたはフォルダーが、**「ファイルの送信元」**ダイアログの「選択されたファイルおよびフォルダー」リストに表示されます。
5. 使用可能なシステムおよびグループのリストを展開します。
6. システムを選択して**「追加」**をクリックして、システムを**「ターゲット・システムおよびグループ」**リストに追加します。このファイルまたはフォルダーに送信したいすべてのシステムについて、このステップを繰り返します。
7. **「OK」**をクリックして、ファイルおよびフォルダーを、現行のデフォルト・パッケージ定義およびスケジュール情報と共に送信します。

『パッケージ定義のオプションの変更』または 44 ページの『ファイルまたはフォルダーの送信日時のスケジュール』を実行することもできます。

パッケージ定義を作成するとそれは保管され、複数のエンドポイント・システムまたはシステム・グループに、定義済みのファイルおよびフォルダーのセットを送信するために、いつでも再使用することができます。ファイルのスナップショットを作成するよう選択する場合、同じファイル・セットの複数のバージョンのコピーを保持することができます。スナップショットを送信すると、配布中にファイルへの更新が行われないので、最後のターゲット・システムは最初のターゲット・システムと同じオブジェクトを受信することになります。

パッケージ定義のオプションの変更

パッケージ定義を使用すると、OS/400 オブジェクトのセットや統合ファイル・システム・ファイルのセットを一緒にグループ化することができます。また、パッケージ定義を使用すると、後の配布のためにそれらのファイルを保存するようにファイルのスナップショットをとることにより、この同じグループのファイルを論理セットまたは物理セットとして表示することができます。

パッケージ定義のオプションを変更するには、次のようにします。

1. 『他のシステムへのファイルまたはフォルダーの送信』のステップを完了します。
2. **「オプション」**タブをクリックします。デフォルト・オプションでは、ファイルをパッケージして送信するときにサブフォルダーを組み込んで、既存のファイルを、送信するファイルで置き換えます。
3. これらのオプションは必要に応じて変更します。
4. **「拡張」**をクリックして、保管および復元の拡張オプションを設定します。
5. **「OK」**をクリックして拡張オプションを保管します。

6. **「OK」**をクリックしてファイルを送信するか、または**「スケジュール」**をクリックしてファイルを送信する時刻を設定します。

関連トピック:

- 『ファイルまたはフォルダーの送信日時のスケジュール』

ファイルまたはフォルダーの送信日時のスケジュール

スケジューラー機能を使用すると、ユーザーにとってファイルまたはフォルダーの送信を行うのに都合のよい日時に柔軟に行うことができます。ファイルまたはフォルダーを送信する日時をスケジュールするには、次のようにします。

1. 43 ページの『他のシステムへのファイルまたはフォルダーの送信』のステップを完了します。
2. **「スケジュール」**をクリックします。
3. ファイルまたはフォルダーの送信日時についてのオプションを選択します。

フォルダーの作成

フォルダーを作成するには、次のようにします。

1. **「iSeries ナビゲーター」**で、使用したいシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. **「統合ファイル・システム」**を展開します。
4. 新規のフォルダーを追加したいファイル・システムを右クリックして、**「新規フォルダー」**を選択します。
5. オブジェクトの新しい名前を**「新規フォルダー」**ダイアログに入力します。
6. **「OK」**をクリックします。

| iSeries サーバー上にフォルダーを作成するとき、ジャーナル管理を使用して新規フォルダー (またはオブジェクト) を保護するかどうかを考慮する必要があります。詳細については、『ジャーナル管理』のトピックを参照してください。

| 関連トピック:

- | ジャーナル処理の開始
- | ジャーナル処理の終了

フォルダーの除去

フォルダーを除去するには、次のようにします。

1. **「iSeries ナビゲーター」**で、使用したいシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. **「統合ファイル・システム」**を展開します。除去したいファイルまたはフォルダーが表示されるまで、展開を続けます。
4. そのファイルまたはフォルダーを右クリックして、**「削除」**を選択します。

ファイル共用の作成

ファイル共用は、iSeries ネットサーバーが iSeries ネットワーク上の PC クライアントと共用するディレクトリー・パスです。 ファイル共用は、iSeries 上の任意の統合ファイル・システム・ディレクトリーで構成することができます。

ファイル共用を作成するには、次のようにします。

1. **「iSeries ナビゲーター」**でシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. **「統合ファイル・システム」**を展開します。
4. 共用を作成したいフォルダーを含むファイル・システムを展開します。
5. 共用を作成したいフォルダーを右クリックして、**「共用」**を選択します。
6. **「新規共用」**を選択します。

ファイル共用の変更

ファイル共用は、iSeries ネットサーバーが iSeries ネットワーク上の PC クライアントと共用するディレクトリー・パスです。 ファイル共用は、iSeries 上の任意の統合ファイル・システム・ディレクトリーで構成することができます。

ファイル共用を変更するには、次のようにします。

1. **「iSeries ナビゲーター」**でシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. **「統合ファイル・システム」**を展開します。
4. 変更したい共用が定義されているフォルダーを展開します。
5. **共用**したい共用が定義されているフォルダーを右クリックします。
6. **「新規共用」**を選択します。

新規のユーザー定義ファイル・システムの作成

ユーザー定義ファイル・システム (UDFS) は、属性を作成および定義するファイル・システムです。 UDFS は、システム上の補助記憶域プール (ASP) 内に存在します。

新規のユーザー定義ファイル・システム (UDFS) を作成するには、次のようにします。

1. **「iSeries ナビゲーター」**でシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. **「統合ファイル・システム」**を展開します。
4. **「ルート (Root)」**を展開します。
5. **「Dev」**を展開します。
6. 新規の UDFS を含めたい補助記憶域プール (ASP) をクリックします。
7. **「ファイル」**メニューから**「新規の UDFS」**を選択します。
8. UDFS 名、説明 (オプション)、監査値、デフォルトのファイル形式の指定、および新規の UDFS 内のファイルのファイル名では大文字小文字を区別するかどうかの指定を、**「新規のユーザー定義ファイル・システム」**ダイアログで行います。

ユーザー定義ファイル・システムのマウント

ユーザー定義ファイル・システム (UDFS) は、属性を作成および定義するファイル・システムです。 UDFS は、システム上の補助記憶域プール (ASP) 内に存在します。 UDFS 内に保管されているデータをアクセスまたは表示するには、UDFS をマウントしなければなりません。

UDFS をマウントすると、フォルダー階層内のマウント・ポイントの下に存在するファイル・システム、ディレクトリー、またはオブジェクトを隠します。これにより、UDFS をアンマウントするまで、それらのファイル・システム・ディレクトリー、またはオブジェクトへのアクセスは不能になります。統合ファイル・システム内のすべてのデータへのアクセスが確実に保持されるようにするには、空のフォルダー上で UDFS をマウントしてください。 UDFS をマウントした後、UDFS 内のファイルはそのフォルダー内からアクセス可能になります。フォルダー内で行われる変更は、隠されたフォルダーにではなく、UDFS に対する変更になります。

注: 独立 ASP 上の UDF を上書きマウントすることはできません。

ユーザー定義ファイル・システム (UDFS) をマウントするには、次のようにします。

1. **「iSeries ナビゲーター」**でシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. **「統合ファイル・システム」**を展開します。
4. **「ルート (Root)」**を展開します。
5. **「Dev」**を展開します。
6. マウントしたい UDFS を含む補助記憶域プール (ASP) をクリックします。
7. オペレーション・ナビゲーターの右ペインにある**「UDFS 名」**欄で、マウントしたい UDFS を右クリックします。
8. **「マウント」**を選択します。

ドラッグ・アンド・ドロップで実行したい場合、同じサーバー上にある統合ファイル・システム内のフォルダーに UDFS をドラッグしてマウントすることもできます。 UDFS を /dev、/dev/QASPxx、/dev/asp_name、他のシステム、またはデスクトップ上にドロップすることはできません。

ユーザー定義ファイル・システムのアンマウント

UDFS をマウントすると、フォルダー階層内のマウント・ポイントの下に存在するファイル・システム、ディレクトリー、またはオブジェクトを隠します。 これにより、UDFS をアンマウントするまで、それらのファイル・システム・ディレクトリー、またはオブジェクトへのアクセスは不能になります。

ユーザー定義ファイル・システム (UDFS) をアンマウントするには、次のようにします。

1. **「オペレーション・ナビゲーター」**でシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. **「統合ファイル・システム」**を展開します。
4. **「ルート (Root)」**を展開します。
5. **「Dev」**を展開します。
6. アンマウントしたい UDFS を含む補助記憶域プール (ASP) をクリックします。
7. iSeries ナビゲーターの右ペインにある**「UDFS 名」**欄で、アンマウントしたい UDFS を右クリックします。

8. **「アンマウント」**を選択します。

ジャーナル処理の開始

ジャーナル処理の主な目的は、オブジェクトの最後の保管以降にそのオブジェクトに加えられた変更を回復できるようにすることです。

オブジェクト上でジャーナル処理を開始するには、以下のようにします。

1. **「iSeries ナビゲーター」**でシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. ジャーナル処理したいオブジェクトを右クリックして、**「ジャーナル処理...」**を選択します。
4. 適切なジャーナル処理オプションを選択した後、**「開始」**をクリックします。

統合ファイル・システム・オブジェクトのジャーナル処理についての詳細は、 iSeries Information Center にある『ジャーナル管理』を参照してください。

ジャーナル処理の終了

ジャーナル処理の主な目的は、オブジェクトの最後の保管以降にそのオブジェクトに加えられた変更を回復できるようにすることです。 オブジェクトのジャーナル処理の開始方法についての詳細は、 『ジャーナル処理の開始』を参照してください。オブジェクト上でいったんジャーナル処理を開始したら、いろいろな理由で、このオブジェクト上でのジャーナル処理を終了したい場合もあるでしょう。

オブジェクト上でのジャーナル処理を終了するには、以下のようにします。

1. **「iSeries ナビゲーター」**でシステムを展開します。
2. **「ファイル・システム」**を展開します。
3. ジャーナル処理を停止したいオブジェクトを右クリックして、**「ジャーナル処理...」**を選択します。
4. **「終了」**をクリックします。

統合ファイル・システム・オブジェクトのジャーナル処理についての詳細は、 iSeries Information Center にある『ジャーナル管理』を参照してください。

第 5 章 統合ファイル・システムのプログラミング・サポート

iSeries サーバーに統合ファイル・システムを追加しても、既存の iSeries サーバー・アプリケーションに影響はありません。プログラミング言語、ユーティリティー、およびデータ記述仕様などのシステム・サポートは、統合ファイル・システムの追加前と同様に操作することができます。

ただし、ストリーム・ファイル、ディレクトリー、その他の統合ファイル・システムのサポートを利用するためには、統合ファイル・システムの機能にアクセスするために提供される C 言語のアプリケーション・プログラム・インターフェース (API) のセットを使用する必要があります。

さらに、統合ファイル・システムを追加することにより、物理データベース・ファイルとストリーム・ファイルとの間でデータをコピーすることができます。 CL コマンド、iSeries Access のデータ転送機能、または API を使用してこのコピーを実行することができます。

以下のトピックでは、統合ファイル・システムのストリーム・ファイルによるコピー機能の使用方法について説明し、統合ファイル・システム機能にアクセスする方法として API を紹介します。

- ストリーム・ファイルとデータベース・ファイルの間でのデータのコピー
- ストリーム・ファイルと保管ファイルの間でのデータのコピー
- API によるデータのコピー
- ソケット・サポート
- 命名および国際サポート
- データ変換

ストリーム・ファイルとデータベース・ファイルの間でのデータのコピー

データ記述仕様 (DDS) などのレコード単位機能を使用するデータベース・ファイルの操作に慣れていると、 ストリーム・ファイルの操作方法との間に、いくつかの根本的な違いがあることがわかります。この違いは、ストリーム・ファイルの構造がデータベース・ファイルとは異なっている (すなわち、ストリーム・ファイルには構造がない) ことによるものです。ストリーム・ファイルのデータにアクセスするには、バイト・オフセットと長さを示します。データベース・ファイルのデータにアクセスするには、一般には、使用するフィールドと処理するレコード数を定義します。

レコード単位ファイルの形式および特性は事前に定義するので、オペレーティング・システムには、ファイルに関する情報があります。そのため、ファイルの形式や特性に適さない操作を避けることができます。ストリーム・ファイルの場合は、オペレーティング・システムには、その形式についての情報がほとんどありません。アプリケーションは、そのファイル形式と、正しい操作方法を調べなければなりません。ストリーム・ファイルを使用すると、非常に柔軟なプログラミング環境が提供されますが、その代わりにオペレーティング・システムから援助を受けることはできません。プログラミング状況によって、ストリーム・ファイルが適している場合と、レコード単位ファイルが適している場合があります。

統合ファイル・システムにおいてストリーム・ファイルとデータベース・ファイルの間でデータをコピーするためには、以下に示すいくつかの方法があります。

- CL コマンドによるデータのコピー
- API によるデータのコピー
- データ転送機能を使用したデータのコピー

CL コマンドによるデータのコピー

ストリーム・ファイルとデータベース・ファイル・メンバーとの間でのデータのコピーを可能にする、以下の 2 セットの CL コマンドがあります。

- CPYTOSTMF および CPYFRMSTMF
- CPYTOIMPF および CPYFRMIMPF

CPYTOSTMF および CPYFRMSTMF コマンド

ストリーム・ファイルからのコピー (CPYFRMSTMF) コマンドおよびストリーム・ファイルへのコピー (CPYTOSTMF) コマンドを使用して、ストリーム・ファイルとデータベース・ファイル・メンバーとの間で、データのコピーを行うことができます。 CPYTOSTMF コマンドを使用すると、データベース・ファイル・メンバーからストリーム・ファイルを作成することができます。また、CPYFRMSTMF コマンドを使用すると、ストリーム・ファイルからデータベース・ファイル・メンバーを作成することができます。コピー先にファイルまたはメンバーが存在しなければ、作成されます。

ただし、いくつかの制限事項があります。データベース・ファイルは、フィールドを 1 つだけ含むプログラム記述物理ファイルか、あるいはテキスト・フィールドを 1 つだけ含むソース物理ファイルのいずれかでなければなりません。コマンドによって、コピーするデータを変換および再編成するための、さまざまなオプションを使用することができます。

CPYTOSTMF および CPYFRMSTMF コマンドを使用して、ストリーム・ファイルと保管ファイルとの間でデータをコピーすることもできます。

CPYTOIMPF および CPYFRMIMPF コマンド

インポート・ファイルへのコピー (CPYTOIMPF) コマンドおよびインポート・ファイルからのコピー (CPYFRMIMPF) コマンドを使用して、ストリーム・ファイルとデータベース・メンバーとの間で、データのコピーを行うこともできます。 CPYTOSTMF コマンドと CPYFRMSTMF コマンドを使用しても、複雑な外部記述 (DDS 記述) のデータベース・ファイルからデータを移動させることはできません。*インポート・ファイル* という語は、ストリーム・タイプのファイルのことを指します。通常この用語は、異機種のデータベース間でデータをコピーする目的で作成されたファイルのことを指します。

ストリーム (つまりインポート) ファイルからコピーする場合に、 CPYFRMIMPF コマンドを使用すると、フィールド定義ファイル (FDF) を指定できます。 FDF は、ストリーム・ファイル中のデータについて記述したものです。また、ストリーム・ファイルが区切られていると指定し、ストリング、フィールド、およびレコードの境界にマークを付ける文字を指定することもできます。時刻や日付など特殊なデータ・タイプを変換するためのオプションも指定できます。

ターゲットのストリーム・ファイルやデータベース・メンバーがすでにある場合は、これらのコマンドを実行するとデータ変換が行われます。ファイルがない場合は、以下の 2 つのステップに従うとデータ変換を行えます。

1. CPYTOIMPF コマンドと CPYFRMIMPF コマンドを使用して、外部記述ファイルとソース物理ファイルの間でデータをコピーします。
2. CPYTOSTMF コマンドと CPYFRMSTMF コマンド (ターゲット・ファイルがあってもなくてもデータ変換の全機能を実行できる) を使用して、ソース物理ファイルとストリーム・ファイルの間でコピーします。

以下に例を示します。

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

DTAFMT パラメーターは、入力ストリーム (インポート) ファイルが区切られていると指定します。他に DTAFMT(*FIXED) も選択できますが、その場合はフィールド定義を指定する必要があります。
FLDDLM、RCDDLM、および STRDLM パラメーターは、区切り文字 (フィールド、レコード、およびストリングの区切り記号として使用される文字) を指定します。

DATFMT パラメーターと TIMFMT パラメーターは、インポート・ファイルにコピーされる日時に関する情報の形式を指定します。

これらのコマンドはプログラムに入れられ、サーバー全体で実行されるので便利です。しかし、インターフェースは複雑になります。

詳しくは、コマンド・ヘルプまたは iSeries Information Center にある『コマンド言語 (CL)』のトピックを参照してください。

API によるデータのコピー

アプリケーション中のストリーム・ファイルにデータベース・ファイル・メンバーをコピーしたい場合は、統合ファイル・システム open()、read()、および write() 関数を使用して、メンバーをオープンし、データの読み取りや書き込みを行うことができます。詳しくは、iSeries Information Center にある『統合ファイル・システム API』のトピックを参照してください。

データ転送機能を使用したデータのコピー

iSeries Access のデータ転送アプリケーションの利点として、わかりやすいグラフィカル・インターフェースと、数値データおよび文字データの自動変換があります。ただし、データ転送を使用するには、iSeries Access をインストールする必要があり、 PC リソースと iSeries サーバー・リソース、およびその両者の間の通信を使用する必要があります。

iSeries Access を PC とサーバーにインストールしてある場合は、データ転送アプリケーションを使用して、ストリーム・ファイルとデータベース・ファイルの間でデータを転送できます。さらに、既存のデータベース・ファイルに基づく新しいデータベース・ファイル内、外部記述データベース・ファイル内、または新しいデータベース・ファイル定義およびファイルに、データを転送することもできます。

以下のタスクは、データ転送アプリケーションを使用してデータをコピーおよび転送するのに役立ちます。

- データベース・ファイルからストリーム・ファイルへのデータの転送
- ストリーム・ファイルからデータベース・ファイルへのデータの転送
- 新しく作成したデータベース・ファイル定義およびファイルへのデータの転送
- 形式記述ファイルの作成

データベース・ファイルからストリーム・ファイルへのデータの転送

サーバー上でデータベース・ファイルからストリーム・ファイルにファイルを転送するには、次のようにします。

1. サーバーへの接続を確立します。
2. iSeries ファイル・システム内の適切なパスにネットワーク・ドライブをマップします。
3. 「iSeries Access for Windows」ウィンドウから、**「iSeries サーバーからのデータ転送 (Data Transfer from iSeries server)」**を選択します。
4. 転送元のサーバーを選択します。

5. iSeries データベース・ライブラリーを利用してコピー対象のファイル名を選択し、転送先のストリーム・ファイルがあるネットワーク・ドライブを選択します。 PC の**「詳細」**を選択して、ストリーム・ファイルの PC ファイル形式を選択することもできます。データ転送は ASCII テキスト、BIFF3、CSV、DIF、タブで区切られたテキスト、 WK4 などの共通 PC ファイル・タイプをサポートします。
6. **「iSeries からデータを転送 (Transfer data from iSeries)」**をクリックして、ファイル転送を実行します。

データ転送アプリケーションを使用して、バッチ・ジョブの中でデータを移動させることもできます。上記の手順に従いますが、**「ファイル」**メニュー・オプションを選択して転送要求を保管します。「iSeries サーバーへのデータ転送 (Data Transfer to iSeries server)」アプリケーションにより、 .DTT または .TFR ファイルが作成されます。「iSeries サーバーからのデータ転送 (Data Transfer from iSeries server)」アプリケーションにより、 .DTF または .TTO ファイルが作成されます。 iSeries Access ディレクトリーで、コマンド行からバッチの中の以下の 2 つのプログラムを実行できます。

- RTOPCB。.DTF ファイルか .TTO ファイルがパラメーターになります。
- RFROMPCB。.DTT ファイルか .TFR ファイルがパラメーターになります。

スケジューラー・アプリケーションを使用して、スケジュールに基づいて上記のどちらかのコマンドが実行されるように設定できます。たとえば、システム・エージェント・ツール (Microsoft Plus Pack の一部) を使用して、実行するプログラム (たとえば、`RTOPCB MYFILE.TTO`) と、そのプログラムを実行する時点を指定できます。

ストリーム・ファイルからデータベース・ファイルへのデータの転送

サーバー上でストリーム・ファイルからデータベース・ファイルにデータを転送するには、次のようにします。

1. サーバーへの接続を確立します。
2. iSeries ファイル・システム内の適切なパスにネットワーク・ドライブをマップします。
3. 「iSeries Access for Windows」ウィンドウから、**「iSeries サーバーへのデータ転送 (Data Transfer to iSeries server)」**を選択します。
4. 転送したい PC ファイル名を選択します。 PC ファイル名の場合は、割り当てたネットワーク・ドライブについて**「参照」**を選択し、ストリーム・ファイルを選択できます。 PC 自体にあるストリーム・ファイルを使用することもできます。
5. 外部記述データベース・ファイルを置きたいサーバーを選択します。
6. **「iSeries サーバーにデータを転送 (Transfer data to iSeries server)」**をクリックして、ファイル転送を実行します。

   注: サーバー上の既存のデータベース・ファイル定義にデータを移動させる場合は、「iSeries サーバーへのデータ転送 (Data Transfer To iSeries)」アプリケーションでは関連した形式記述ファイル (FDF) を使用する必要があります。 FDF ファイルは、ストリーム・ファイルの形式を記述したもので、データをデータベース・ファイルからストリーム・ファイルに転送する際に、「iSeries サーバーからのデータ転送 (Data Transfer from iSeries server)」アプリケーションによって作成されます。ストリーム・ファイルからデータベース・ファイルへのデータ転送を完了するには、**「iSeries へのデータ転送 (Transfer data to iSeries)」**をクリックします。既存の .FDF ファイルが使用できない場合は、すぐに .FDF ファイルを作成できます。

データ転送アプリケーションを使用して、バッチ・ジョブの中でデータを移動させることもできます。上記の手順に従いますが、**「ファイル」**メニュー・オプションを選択して転送要求を保管します。「iSeries サーバーへのデータ転送 (Data Transfer to iSeries server)」アプリケーションにより、 .DTT または .TFR ファイルが作成されます。「iSeries サーバーからのデータ転送 (Data Transfer from iSeries server)」アプリケ

ーションにより、 .DTF または .TTO ファイルが作成されます。 iSeries Access ディレクトリーで、コマンド行からバッチの中の以下の 2 つのプログラムを実行できます。

- RTOPCB。.DTF ファイルか .TTO ファイルがパラメーターになります。
- RFROMPCB。.DTT ファイルか .TFR ファイルがパラメーターになります。

スケジューラー・アプリケーションを使用して、スケジュールに基づいて上記のどちらかのコマンドが実行されるように設定できます。たとえば、システム・エージェント・ツール (Microsoft Plus Pack の一部) を使用して、実行するプログラム (たとえば、`RTOPCB MYFILE.TTO`) と、そのプログラムを実行する時点を指定できます。

新しく作成したデータベース・ファイル定義およびファイルへのデータの転送

新しく作成したデータベース・ファイル定義およびファイルにデータを転送するには、以下のようにします。

1. サーバーへの接続を確立します。
2. iSeries ファイル・システム内の適切なパスにネットワーク・ドライブをマップします。
3. 「iSeries Access for Windows」ウィンドウから、**「iSeries サーバーへのデータ転送 (Data Transfer to iSeries server)」**を選択します。
4. 「iSeries サーバーへのデータ転送 (Data Transfer to iSeries server)」アプリケーションの**「ツール」**メニューを開きます。
5. **「iSeries データベース・ファイルの作成 (Create iSeries database file)」**を選択します。

   新規の iSeries データベース・ファイルを既存の PC ファイルから作成するためのウィザードが表示されます。 iSeries ファイルの基になる PC ファイルの名前、作成する iSeries ファイルの名前、および他のいくつかの必要な詳細情報を指定するように要求されます。このツールは、指定されたストリーム・ファイルを解析して、転送先のデータベース・ファイルに必要なフィールドの数、タイプ、およびサイズを判別します。続いてサーバー上にデータベース・ファイル定義を作成します。

形式記述ファイルの作成

サーバー上の既存のデータベース・ファイル定義にデータを移動させる場合は、「iSeries サーバーへのデータ転送 (Data Transfer To iSeries server)」アプリケーションでは関連した形式記述ファイル (FDF) を使用する必要があります。 FDF ファイルは、ストリーム・ファイルの形式を記述したもので、データをデータベース・ファイルからストリーム・ファイルに転送する際に、「iSeries サーバーからのデータ転送 (Data Transfer from iSeries server)」アプリケーションによって作成されます。

.FDF ファイルを作成するには、次のようにします。

1. ソース・ストリーム・ファイルと同じ形式 (フィールド数、データ・タイプ) の外部記述データベース・ファイルを作成します。
2. そのデータベース・ファイルの中に一時データ・レコードを 1 つ作成します。
3. 「iSeries サーバーからのデータ転送 (Data Transfer from iSeries server)」機能を使用して、このデータベース・ファイルからストリーム・ファイルとそれに関連した .FDF ファイルを作成します。
4. それから「iSeries サーバーへデータ転送 (Data Transfer to iSeries server)」機能を使用することができます。この .FDF ファイルと転送したいソース・ストリーム・ファイルを指定します。

ストリーム・ファイルと保管ファイルの間でのデータのコピー

保管ファイルは、保管コマンドおよび復元コマンドと共に使用し、それ以外の方法ではテープまたはディスケットに書き込まれることがないデータを保存します。そのファイルはデータベース・ファイルと同様の方法で使用して、保管 / 復元情報を含むレコードの読み取りまたは書き込みを行うこともできます。保管ファイルはさらに、オブジェクトを SNADS ネットワーク上の他のユーザーに送信するためにも使用できます。

CPY コマンドを使用して、保管ファイルをストリーム・ファイルにコピーすることができます。またその逆も可能です。しかし、ストリーム・ファイルを保管ファイル・オブジェクトにコピーして戻す場合、そのデータは有効な保管ファイル・データでなければなりません (保管ファイルを元としており、ストリーム・ファイルにコピーされたデータでなければなりません)。

PC クライアントを使用することによって、保管ファイルにアクセスしてデータを PC ストレージまたは LAN にコピーすることもできます。ただし、保管ファイル内のデータには、ネットワーク・ファイル・システム (NFS) を介してはアクセスできないことに注意してください。

API を使用した操作の実行

統合ファイル・システムのディレクトリーおよびストリーム・ファイルの操作を実行するアプリケーション・プログラム・インターフェース (API) は、 C 言語の関数の形式になっています。選択できる関数の種類には以下の 2 種類があり、どちらの関数も ILE (統合言語環境) C/400 で作成されたプログラム内で使用できます。

- OS/400 に組み込まれている統合ファイル・システム C 関数。
- ILE C/400 ライセンス・プログラムで提供される C 関数。

統合ファイル・システムの関数は、統合ファイル・システムのストリーム入出力サポートを介する場合だけ作動します。以下の API がサポートされています。

表 3. 統合ファイル・システム API

関数	説明
access()	ファイルのアクセス可能性を判別する
accessx()	ユーザーのクラスのファイルのアクセス可能性を判別する
chdir()	現行ディレクトリーを変更する
chmod()	ファイル権限を変更する
chown()	ファイルの所有者とグループを変更する
close()	ファイル記述子をクローズする
closedir()	ディレクトリーをクローズする
creat()	新規ファイルを作成するか、既存のファイルに上書きする
creat64()	新規ファイルを作成するか、既存のファイルに上書きする (ラージ・ファイル使用可能)
DosSetFileLocks()	ファイルのバイト範囲をロックおよびアンロックする
DosSetFileLocks64()	ファイルのバイト範囲をロックおよびアンロックする (ラージ・ファイル使用可能)
DosSetRelMaxFH()	ファイル記述子の最大数を変更する
dup()	オープン・ファイル記述子をコピーする
dup2()	オープン・ファイル記述子を別の記述子にコピーする

表 3. 統合ファイル・システム API (続き)

関数	説明
faccessx()	記述子でユーザーのクラスのファイルのアクセス可能性を判別する
fchdir()	記述子で現行ディレクトリーを変更する
fchmod()	記述子でファイル権限を変更する
fchown()	記述子でファイルの所有者とグループを変更する
fcntl()	ファイル制御処置を行う
fpathconf()	構成可能なパス名変数を記述子によって入手する
fstat()	記述子によってファイル情報を入手する
fstat64()	記述子によってファイル情報を入手する (ラージ・ファイル使用可能)
fstatvfs()	記述子から情報を入手する
fstatvfs64()	記述子から情報を入手する (64 ビット使用可能)
fsync()	ファイルへの変更を同期化する
ftruncate()	ファイルを切り捨てる
ftruncate64()	ファイルを切り捨てる (ラージ・ファイル使用可能)
getcwd()	現行ディレクトリーのパス名を入手する
getegid()	有効なグループ ID を入手する
geteuid()	有効なユーザー ID を入手する
getgid()	実際のグループ ID を入手する
getgrgid()	グループ ID を使用してグループ情報を入手する
getgrnam()	グループ名を使用してグループ情報を入手する
getgroups()	グループ ID を入手する
getwpnam()	ユーザー名のユーザー情報を入手する
getpwuid()	ユーザー ID のユーザー情報を入手する
getuid()	実際のユーザー ID を入手する
givedescriptor()	別のジョブにファイル・アクセス権を与える
ioctl()	ファイル入出力制御処置を行う
link()	ファイルへのリンクを設定する
lseek()	ファイルの読み取り / 書き込みオフセットを設定する
lseek64()	ファイルの読み取り / 書き込みオフセットを設定する (ラージ・ファイル使用可能)
lstat()	ファイル情報またはリンク情報を入手する
lstat64()	ファイル情報またはリンク情報を入手する (ラージ・ファイル使用可能)
mmap()	メモリー・マップを作成する
mmap64()	メモリー・マップを作成する (ラージ・ファイル使用可能)
mprotect()	メモリー・マップ保護を変更する
msync()	メモリー・マップを同期化する
munmap()	メモリー・マップを除去する
mkdir()	ディレクトリーを作成する

表 3. 統合ファイル・システム API (続き)

関数	説明
mkfifo()	FIFO 特殊ファイルを作成する
open()	ファイルをオープンする
open64()	ファイルをオープンする (ラージ・ファイル使用可能)
opendir()	ディレクトリーをオープンする
pathconf()	構成可能なパス名変数を入手する
pipe()	ソケットを使ったプロセス間チャネルを作成する
pread()	オフセットを指定して記述子から読み取る
pread64()	オフセットを指定して記述子から読み取る (ラージ・ファイル使用可能)
pwrite()	オフセットを指定して記述子に書き込む
pwrite64()	オフセットを指定して記述子に書き込む (ラージ・ファイル使用可能)
QjoEndJournal()	ジャーナル処理を終了する
QjoRetrieveJournal Information()	ジャーナル情報を検索する
QJORJIDI()	ジャーナル ID 情報を検索する
QJOSJRNE()	ジャーナル項目を送信する
QjoStartJournal()	ジャーナル処理を開始する
QlgAccess()	ファイルのアクセス可能性を判別する (NLS 化パス名を使用)
QlgAccessx()	ユーザーのクラスのファイルのアクセス可能性を判別する (NLS 化パス名を使用)
QlgChdir()	現行ディレクトリーを変更する (NLS 化パス名を使用)
QlgChmod()	ファイル許可を変更する (NLS 化パス名を使用)
QlgChown()	ファイルの所有者およびグループを変更する (NLS 化パス名を使用)
QlgCreat()	新規ファイルを作成するか、既存のファイルに上書きする (NLS 化パス名を使用)
QlgCreat64()	新規ファイルを作成するか、既存のファイルに上書きする (ラージ・ファイル使用可能、NLS 化パス名を使用)
QlgCvtPathToQSYSObjName()	統合ファイル・システムのパス名を QSYS オブジェクト名に解決する (NLS 化パス名を使用)
QlgGetAttr()	オブジェクトのシステム属性を入手する (NLS 化パス名を使用)
QlgGetcwd()	現行ディレクトリーのパス名を入手する (NLS 化パス名を使用)
QlgGetPathFromFileID()	オブジェクトのパス名をファイル ID から入手する (NLS 化パス名を使用)
QlgGetpwnam()	ユーザー名についてのユーザー情報を入手する (NLS 化パス名を使用)
QlgGetpwnam_r()	ユーザー名についてのユーザー情報を入手する (NLS 化パス名を使用)
QlgGetpwuid()	ユーザー ID についてのユーザー情報を入手する (NLS 化パス名を使用)

表 3. 統合ファイル・システム API (続き)

関数	説明
QlgGetpwuid_r()	ユーザー ID についてのユーザー情報を入手する (NLS 化パス名を使用)
QlgLchown()	シンボリック・リンクの所有者およびグループを変更する (NLS 化パス名を使用)
QlgLink()	ファイルへのリンクを作成する (NLS 化パス名を使用)
QlgLstat()	ファイル情報またはリンク情報を入手する (NLS 化パス名を使用)
QlgLstat64()	ファイル情報またはリンク情報を入手する (ラージ・ファイル使用可能、NLS 化パス名を使用)
QlgMkdir()	ディレクトリーを作成する (NLS 化パス名を使用)
QlgMkfifo()	FIFO 特殊ファイルを作成する (NLS 化パス名を使用)
QlgOpen()	ファイルをオープンする (NLS 化パス名を使用)
QlgOpen64()	ファイルをオープンする (ラージ・ファイル使用可能、NLS 化パス名を使用)
QlgOpendir()	ディレクトリーをオープンする (NLS 化パス名を使用)
QlgPathconf()	構成可能なパス名変数を入手する (NLS 化パス名を使用)
QlgProcessSubtree()	ディレクトリー・ツリー内のディレクトリーまたはオブジェクトを処理する (NLS 化パス名を使用)
QlgReaddir()	ディレクトリー項目を読み取る (NLS 化パス名を使用)
QlgReaddir_r()	ディレクトリー項目を読み取る (スレッド・セーフ、NLS 化パス名を使用)
QlgReadlink()	シンボリック・リンクの値を読み取る (NLS 化パス名を使用)
QlgRenameKeep()	ファイルまたはディレクトリーの名前を変更する。名前がすでに存在していれば、*新規のもの* を保持する (NLS 化パス名を使用)
QlgRenameUnlink()	ファイルまたはディレクトリーの名前を変更する。名前がすでに存在していれば、*新規のもの* をリンク解除する (NLS 化パス名を使用)
QlgRmdir()	ディレクトリーを除去する (NLS 化パス名を使用)
QlgSaveStgFree()	オブジェクト・データを保管してその記憶域を解放する (NLS 化パス名を使用)
QlgSetAttr()	オブジェクトのシステム属性を設定する (NLS 化パス名を使用)
QlgStat()	ファイル情報を入手する (NLS 化パス名を使用)
QlgStat64()	ファイル情報を入手する (ラージ・ファイル使用可能、NLS 化パス名を使用)
QlgStatvfs()	ファイル・システム情報を入手する (NLS 化パス名を使用)
QlgStatvfs64()	ファイル・システム情報を入手する (ラージ・ファイル使用可能、NLS 化パス名を使用)
QlgSymlink()	シンボリック・リンクを作成する (NLS 化パス名を使用)
QlgUnlink()	ファイルをリンク解除する (NLS 化パス名を使用)
QlgUtime()	ファイル・アクセスおよび変更回数を設定する (NLS 化パス名を使用)

表 3. 統合ファイル・システム API (続き)

関数	説明
QP0FPTOS()	各種ファイル・システム機能を実行する
Qp0lCvtPathToSYSObjName()	統合ファイル・システムのパス名を QSYS オブジェクト名に解決する
Qp0lFLOP()	オブジェクトに各種操作を実行する
Qp0lGetAttr()	オブジェクトのシステム属性を入手する
Qp0lGetPathFromFileID()	オブジェクトのパス名をファイル ID から入手する
Qp0lOpen()	パス名が NLS 化されたファイルをオープンする
Qp0lProcessSubtree()	ディレクトリー・ツリー内のディレクトリーまたはオブジェクトを処理する
Qp0lRenameKeep()	ファイルまたはディレクトリーの名前を変更する。名前がすでに存在していれば、*新規のもの* を保持する
Qp0lRenameUnlink()	ファイルまたはディレクトリーの名前を変更する。名前がすでに存在していれば、*新規のもの* をリンク解除する
QP0LROR()	オブジェクト参照子を検索する
Qp0lSaveStgFree()	オブジェクト・データを保管しその記憶域を解放する
Qp0lSetAttr()	オブジェクトのシステム属性を設定する
Qp0lUnlink()	パス名が NLS 化されたファイルをリンク解除する
qsysetegid()	有効なグループ ID を設定する
qsyseteuid()	有効なユーザー ID を設定する
qsysetgid()	グループ ID を設定する
qsysetregid()	実際の、有効なグループ ID を設定する
qsysetreuid()	実際の、有効なユーザー ID を設定する
qsysetuid()	ユーザー ID を設定する
QZNFRTVE()	NFS エクスポート情報を検索する
read()	ファイルから読み取る
readdir()	ディレクトリー項目を読み取る
readdir_r()	ディレクトリー項目 (スレッド・セーフ) を読み取る
readlink()	シンボリック・リンクの値を読み取る
readv()	ファイル (ベクトル) から読み取る
rename()	ファイルまたはディレクトリーの名前を変更する。Qp0lRenameKeep() または Qp0lRenameUnlink() のセマンティクスを持つように定義することができる
rewinddir()	ディレクトリー・ストリームをリセットする
rmdir()	ディレクトリーを削除する
select()	複数のファイル記述子の入出力状況を調べる
stat()	ファイル情報を入手する
stat64()	ファイル情報を入手する (ラージ・ファイル使用可能)
statvfs()	ファイル・システム情報を入手する
statvfs64()	ファイル・システム情報を入手する (ラージ・ファイル使用可能)
symlink()	シンボリック・リンクを設定する

表 3. 統合ファイル・システム API (続き)

関数	説明
sysconf()	システム構成変数を入手する
takedescriptor()	別のジョブからファイル・アクセス権を受け取る
umask()	ジョブに権限マスクを設定する
unlink()	ファイルへのリンクを除去する
utime()	ファイル・アクセスおよび修正回数を設定する
write()	ファイルに書き込む
writev()	ファイル (ベクトル) に書き込む

注: これらの関数は、OS/400 ソケットにも使用されます。これらの関数を、特定のファイル・システムで使用する際の制限事項については、 4 ページの『統合ファイル・システムのファイル・システム』を参照してください。統合ファイル・システム C 関数を使用したプログラムの例については、 113 ページの『付録 B. 統合ファイル・システムの C 関数を使用するプログラムの例』を参照してください。

統合ファイル・システム API についての詳細は、以下のトピックを参照してください。

- ILE C/400 の関数
- API のラージ・ファイル・サポート
- API のパス名規則
- ファイル記述子
- セキュリティー
- iSeries Information Center にある『Application programming interfaces (APIs)』のトピック

ILE C/400 の関数

ILE C/400 では、米国規格協会 (ANSI) で定義されている標準 C 関数を備えています。これらの関数については、データ管理入出力サポートか統合ファイル・システム・ストリーム入出力サポートのうち、 C プログラムの作成時に指定されたもののどちらかを介して操作できます。コンパイラーは、特に指定されなければデータ管理入出力を使用します。

統合ファイル・システム・ストリーム入出力の使用をコンパイラーに指示するには、 ILE C/400 モジュールの作成 (CRTCMOD) またはバインド済み C プログラムの作成 (CRTBNDC) コマンドのシステム・インターフェース・オプション (SYSIFCOPT) パラメーターに、 *IFSIO と指定しなければなりません。*IFSIO が指定されると、データ管理入出力関数の代わりに統合ファイル・システム入出力関数がバインドされます。その結果、ILE C/400 の C 関数は統合ファイル・システム関数を使用して入出力を実行します。

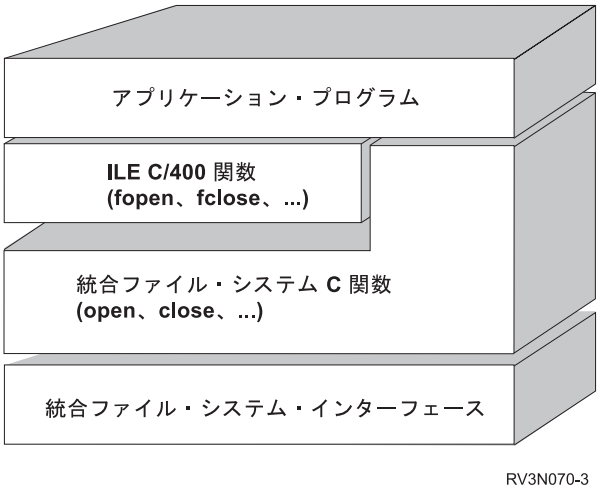


図 10. ILE C/400 関数の統合ファイル・システム・ストリーム入出力関数の使用

統合ファイル・システム・ストリーム入出力での ILE C/400 関数の使用についての詳細は、 WebSphere Development Studio: ILE C/C++ Programmers Guide を参照してください。個々の ILE C/400 の C 関数についての詳細は、 WebSphere Development Studio: C/C++ Language Reference を参照してください。

API のラージ・ファイル・サポート

統合ファイル・システム API は拡張され、非常に大きなファイルの保管や操作がアプリケーションで行えるようになっています。 統合ファイル・システムを使用すると、「ルート (/)」、オープン・システム・ファイル・システム (QOpenSys)、およびユーザー定義ファイル・システムで最大 256 ギガバイトのストリーム・ファイル・サイズを使用できます。

統合ファイル・システムには 64 ビット UNIX タイプ API のセットが備えられているので、既存の 32 ビット API を 64 ビット API に簡単にマッピングし、 8 バイト整数引き数を使用して、ラージ・ファイル・サイズおよびオフセットにアクセスできます。 各 64 ビット API についての詳細は、 iSeries Information Center にある『Integrated File System APIs』のトピックを参照してください。

以下の情報は、アプリケーションでラージ・ファイル・サポートを使用できるようにするために提供されています。

1. コンパイル時にマクロ・ラベル _LARGE_FILE_API を定義すると、 64 ビット使用可能 API とデータ構造に対するアクセス権がアプリケーションに付与されます。たとえば、アプリケーションで stat64() API と stat64 構造を使用したい場合は、コンパイル時に _LARGE_FILE_API を定義する必要があります。
2. コンパイル時にアプリケーションにより、マクロ・ラベル _LARGE_FILES が定義されると、既存の API とデータ構造が 64 ビット・バージョンにマップされます。 たとえば、コンパイル時にアプリケーションで _LARGE_FILES が定義されると、 stat() API に対する呼び出しが stat64() API にマップされ、stat() 構造が stat64() 構造にマップされます。

アプリケーションでラージ・ファイル・サポートを使用したい場合は、コンパイル時に _LARGE_FILE_API を定義して直接 64 ビット API にコーディングするか、またはコンパイル時に _LARGE_FILES を定義することができます。該当する API とデータ構造はすべて 64 ビット・バージョンに自動的にマップされます。

アプリケーションでラージ・ファイル・サポートを使用しないのであれば、操作に影響はないので、変更を加えずに引き続き統合ファイル・システム API を使用できます。

API のパス名規則

統合ファイル・システムまたは ILE C/400 API を使用してオブジェクトを操作する場合には、ディレクトリー・パスでオブジェクトを識別します。 API でパス名を指定する際の規則の要約を、以下に示します。これらの規則の中で、**オブジェクト**という用語は、任意のディレクトリー、ファイル、リンク、その他のオブジェクトを表します。

- パス名は、ディレクトリー階層の最高レベルから、階層順に指定します。パスの各構成要素は、スラッシュ (/) で区切ります。たとえば、次のようになります。

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

  円記号 (¥) は、区切り文字とは認識されません。名前に使われている他の文字と同様に扱われます。
- オブジェクト名は、各ディレクトリー内で固有のものでなければなりません。
- パス名の各構成要素の最大長と、パス名ストリングの最大長は、ファイル・システムごとに異なります。各ファイル・システムでの長さの制限については、 65 ページの『ファイル・システムの比較』を参照してください。
- 次のように、パス名の先頭が / である場合は、パスが「ルート (/)」ディレクトリーから始まることを示します。

  ```
  /Dir1/Dir2/Dir3/UsrFile
  ```
- 次のように、パス名が / で始まっていなければ、現行ディレクトリーからパスが始まるものと見なされます。

  ```
  MyDir/MyFile
  ```

  MyDir は、現行ディレクトリーのサブディレクトリーです。
- iSeries サーバーの特殊値と混同しないようにするため、パス名の先頭を単一アスタリスク (*) 文字にすることはできなくなっています。パス名の先頭でパターン照合を行うには、アスタリスクを 2 つ使用してください。たとえば、次のようになります。

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

  これは、アスタリスク (*) の前に他の文字がない相対パス名だけに適用されるので注意してください。
- QSYS.LIB ファイル・システム内のオブジェクトを操作する場合、構成要素名は、 *name.object-type* の形式になっていなければなりません。

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

| 詳細については、 77 ページの『ライブラリー・ファイル・システム (QSYS.LIB)』を参照してください。

| • 独立 ASP QSYS.LIB ファイル・システム内のオブジェクトを操作する場合、構成要素名は、
| *name.object-type* の形式になっていなければなりません。たとえば、以下のようになります。

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

| 詳細については、 80 ページの『独立 ASP QSYS.LIB』を参照してください。

- パス名の中でコロン (:) を使用しないでください。コロンはサーバーで特殊な意味をもちます。

| • 統合ファイル・システム・コマンドのパス名 (31 ページの『CL コマンドおよび表示画面のパス名規則』を参照) とは異なり、アスタリスク (*)、疑問符 (?)、アポストロフィ (')、引用符 (")、および波形

記号 (~) に特別な意味はなく、名前に使われている他の文字と同様に扱われます。この規則の例外である API は、 QjoEndJournal() と QjoStartJournal のみです。

ファイル記述子

米国規格協会 (ANSI) で定義されているように、 ILE C/400 ストリーム入出力の機能を使用して、ファイルの操作を実行する場合、ファイルの識別にはポインターが使用されます。統合ファイル・システム C 関数を使用する場合には、**ファイル記述子**を指定してファイルを識別します。ファイル記述子は、各ジョブの中で固有な正の整数でなければなりません。ジョブは、ファイルの操作を実行するときに、ファイル記述子を使用してオープン・ファイルを識別します。ファイル記述子は、統合ファイル・システムで使用する C 関数では変数 *fildes* で表され、ソケットで使用する C 関数では変数 *descriptor* で表されます。

各ファイル記述子は、ファイル・オフセット、ファイルの状況、およびファイルへのアクセス・モードなどの情報を含む**オープン・ファイル記述**を参照します。複数のファイル記述子が同じオープン・ファイル記述を参照することができますが、 1 つのファイル記述子では 1 つのオープン・ファイル記述のみ参照することができます。

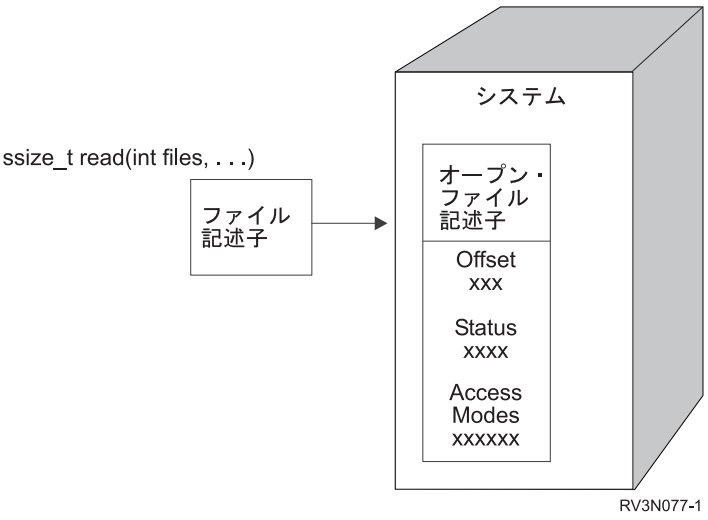


図 11. ファイル記述子とオープン・ファイル記述

ILE C/400 ストリーム入出力関数が統合ファイル・システムで使用される場合、 ILE C/400 実行時サポートが、ファイル・ポインターをファイル記述子に変換します。

「ルート (/)」、QOpenSys、またはユーザー定義のファイル・システムを使用している場合には、オープン・ファイル記述へのアクセス権を 1 つのジョブから別のジョブに渡して、そのジョブがファイルにアクセスできるようにすることができます。これは、givedescriptor() または takedescriptor() 関数を使用して、ジョブ間でファイル記述子を渡すことによって行います。これらの関数についての詳細は、 iSeries Information Center にある『Socket プログラミング』、または『Sockets APIs』のトピックを参照してください。

セキュリティー

統合ファイル・システム API を使用している場合には、データ管理インターフェースを使用しているときと同様に、オブジェクトへのアクセスを制限することができます。ただし、借用権限はサポートされていないことに注意してください。統合ファイル・システム API は、ジョブを実行しているユーザー・プロファイルの権限を使用します。

各ファイル・システムには、それぞれ独自の特別な権限要件があります。 NFS サーバー・ジョブのみはこの規則の例外になります。ネットワーク・ファイル・システム・サーバーの要求は、要求時に NFS サーバーで受け取ったユーザー識別コード (UID) 番号で示されるユーザーのプロファイル下で実行されます。

サーバーにおける権限は、UNIX システムにおける**許可**と等しいものです。許可のタイプは、読み取りと書き込み (ファイルまたはディレクトリーの場合) および実行 (ファイルの場合)、または検索 (ディレクトリーの場合) です。許可は、ファイルまたはディレクトリーの 『アクセスのモード』 を作成する、許可ビットのセットで識別されます。許可ビットは、「モード変更」関数 chmod() または fchmod() を使用して変更することができます。 umask() 関数を使用して、ジョブがファイルを作成するごとに、セットされるファイル許可ビットを制御することもできます。

データ・セキュリティーと権限についての詳細は、機密保護解説書  を参照してください。

ソケット・サポート

アプリケーションが「ルート (/)」、QOpenSys、またはユーザー定義のファイル・システムを使用している場合には、統合ファイル・システムの**ローカル・ソケット**・サポートを利用することができます。ローカル・ソケット・オブジェクト (オブジェクト・タイプは *SOCKET) を使用すると、同じシステムで実行されている 2 つのジョブの間に、通信接続を設定することができます。

片方のジョブでは、C 言語関数 bind() を使用して、ローカル・ソケット・オブジェクトを作成するための接続点を設定します。もう片方のジョブでは、connect()、sendto()、または sendmsg() 関数に、ローカル・ソケット・オブジェクトの名前を指定します。これらの関数および一般的なソケットの概念については、iSeries Information Center にある『Socket プログラミング』のトピックを参照してください。

接続が設定されると、 write() や read() などの統合ファイル・システム関数を使用して、 2 つのジョブ間でデータの送受信を行うことができます。転送されるデータは、実際にはソケット・オブジェクトを通りません。ソケット・オブジェクトは、2 つのジョブが互いを見つけることができる接点にすぎません。

2 つのジョブの通信が終了すると、それぞれのジョブは close() 関数を使用して、ソケット接続をクローズします。 ローカル・ソケット・オブジェクトは、 unlink() 関数または リンクの除去 (RMVLNK) コマンドを使用して除去するまで、システムに残ります。

ローカル・ソケット・オブジェクトを保管することはできません。

命名および国際サポート

「ルート (/)」および QOpenSys ファイル・システムのサポートでは、異なる各国語および装置で使用されるエンコード・スキーム間で、オブジェクト名の文字が変わらないことが保証されています。オブジェクト名がシステムに渡されると、名前に使用されているすべての文字が、 16 ビット形式に変換され、標準のコード化表現となります (24 ページの『名前の継続性』を参照)。名前を使用するとき、使用するコード・ページに適するコード化形式に変換されます。

名前で使用している文字が、変換されるコード・ページに含まれていない場合、その名前は無効であるとして拒否されます。

コード・ページが変わっても文字は同じになるので、あるコード・ページを使用したときに、特定の文字が別の特定の文字に変換されることを前提とした操作は行わないでください。たとえば、番号記号とポンド記号は、別々のコード・ページで同じコード化表現をもちますが、番号記号がポンド記号に変わると見なしてはなりません。

オブジェクトの拡張属性の名前は、オブジェクト名と同様に変換されるので、同じ考慮事項が適用されます。

コード・ページについての詳細は、 iSeries Information Center にある『グローバリゼーション』のトピックを参照してください。

データ変換

統合ファイル・システムによりファイルにアクセスするときに、ファイル内のデータが変換されるかどうかは、ファイルのオープン時に要求するオープン・モードによって異なります。

オープン・ファイルは、次の 2 つのオープン・モードのいずれかです。

バイナリー
データは、ファイルでの読み取りおよび書き込み時には変換されません。データの処理は、アプリケーションで行われます。

テキスト
ファイルでのデータの読み取りおよび書き込みは、データがテキスト形式になっているものとして行われます。ファイルから読み取られたデータは、ファイルのコード化文字セット ID (CCSID) から、受信先のアプリケーション、ジョブ、あるいはシステムの CCSID に変換されます。ファイルにデータを書き込むときは、アプリケーション、ジョブ、またはシステムの CCSID から、ファイルの CCSID に変換されます。真のストリーム・ファイルの場合には、行書式設定文字 (復帰、タブ、ファイル終わりなど) は、すべて 1 つの CCSID から別の CCSID に変換されるだけです。

ストリーム・ファイルとして使用されているレコード・ファイルから読み取る場合には、行の終わりの文字 (復帰と改行) が、各レコードのデータの終わりに付加されます。レコード・ファイルに書き込む場合は、次のようになります。

- 行の終わりの制御文字は削除されます。
- タブ文字は、次のタブ位置まで適切な数のブランクに置き換えられます。
- 行は、ブランク (ソース物理ファイル・メンバーの場合) またはヌル文字 (データ物理ファイル・メンバーの場合) で埋められます。

オープン要求の際に、次のいずれかを指定してください。

Binary, Forced (バイナリー、強制)
データは、ファイルの実際の内容に関係なく、バイナリーとして処理されます。データの処理方法は、アプリケーションで決められます。

Text, Forced (テキスト、強制)
データは、テキストであるものと見なされます。データは、ファイルの CCSID から、アプリケーションの CCSID に変換されます。

統合ファイル・システムの open() 関数では、*Binary, Forced* がデフォルトとして使用されます。

第 6 章 統合ファイル・システムのファイル・システム

統合ファイル・システム内のファイル・システムは、次のとおりです。

- 「ルート」
- QOpenSys
- UDFS
- QSYS.LIB
- 独立 ASP QSYS.LIB
- QDLS
- QOPT
- QNetWare
- QNTC
- QFileSvr.400
- NFS

それぞれのファイル・システムの概要については、 『ファイル・システムの比較』を参照してください。

ファイル・システムの比較

表 4 と 67 ページの表 5 は、各ファイル・システムの機能と制限事項を要約したものです。

表 4. ファイル・システムの要約 (1/2)

機能	/ (「ルート」)	QOpenSys	QSYS.LIB[16]	QDLS	QNTC
OS/400 の標準機能	Yes	Yes	Yes	Yes	Yes
ファイルのタイプ	ストリーム	ストリーム	レコード [12]	ストリーム	ストリーム
OfficeVision との統合 (たとえば、ファイルがメールできる)	No	No	No	Yes	No
OS/400 ファイル・サーバーによるアクセス	Yes	Yes	Yes	Yes	Yes
ファイル・サーバー I/O プロセッサーによる直接アクセス [1]	No	No	No	No	Yes
オープン / クローズの相対的な速度	中 [2]	中 [2]	低 [2]	低 [2]	中 [2]
英語の大文字小文字を区別した名前の検索	No	Yes	No [4]	No [5]	No
パス名の各構成要素の最大長	255 文字	255 文字	10.6 文字 [6]	8.3 文字 [7]	255 文字
パス名の最大長 [8]	16MB	16MB	55 ～ 66 文字 [4]	82 文字	255 文字
オブジェクトの拡張属性の最大長	2GB	2GB	不定 [9]	32KB	64KB
ファイル・システム内のディレクトリー階層の最高レベル	制限なし [10]	制限なし [10]	3	32	127
オブジェクトごとのリンクの最大数 [11]	不定 [15]	不定 [15]	1	1	1

表 4. ファイル・システムの要約 (1/2) (続き)

機能	/ (「ルート」)	QOpenSys	QSYS.LIB[16]	QDLS	QNTC
シンボリック・リンクのサポート	Yes	Yes	No	No	No
オブジェクト / ファイルの所有者の設定	Yes	Yes	Yes	Yes	No
統合ファイル・システム・コマンドのサポート	Yes	Yes	Yes	Yes	Yes
統合ファイル・システム API のサポート	Yes	Yes	Yes	Yes	Yes
階層ファイル・システム (HFS) API のサポート	No	No	No	Yes	No
スレッド・セーフ [13]	Yes	Yes	Yes	No	Yes
オブジェクト・ジャーナル処理のサポート	Yes	Yes	Yes[14]	No	No

表 4. ファイル・システムの要約 (1/2) (続き)

機能	/ (「ルート」)	QOpenSys	QSYS.LIB[16]	QDLS	QNTC

注:

1. ファイル・サーバー I/O プロセッサーは、LAN サーバーが使用するハードウェアです。
2. OS/400 ファイル・サーバーによりアクセスした場合。
3. LAN サーバーのクライアント PC によりアクセスした場合。 iSeries API を使用したアクセスは、比較的遅くなります。
4. QSYS.LIB ファイル・システムのパス名の最大長は 55 文字です。詳細については、 77 ページの『ライブラリー・ファイル・システム (QSYS.LIB)』を参照してください。 独立 ASP QSYS.LIB ファイル・システムのパス名の最大長は 66 文字です。詳細については、 80 ページの『独立 ASP QSYS.LIB』を参照してください。
5. 詳細については、 83 ページの『文書ライブラリー・サービス・ファイル・システム (QDLS)』を参照してください。
6. オブジェクト名は 10 文字まで、オブジェクト・タイプは 6 文字までです。詳細については、 77 ページの『ライブラリー・ファイル・システム (QSYS.LIB)』を参照してください。
7. 名前は 8 文字まで、ファイル・タイプの拡張子 (ある場合) は 1〜3 文字です。詳細については、 83 ページの『文書ライブラリー・サービス・ファイル・システム (QDLS)』を参照してください。
8. / で始まり、ファイル・システム名が続く絶対パス名 (/QDLS... など) を前提とします。
9. QSYS.LIB および独立 ASP QSYS.LIB ファイル・システムは、 .SUBJECT、.CODEPAGE、および .TYPE という 3 つの事前定義拡張属性をサポートします。最大長は、これらの 3 つの拡張属性の合計の長さによって決まります。
10. 実際には、ディレクトリー・レベルは、プログラムおよびシステムのスペース制限によって制限されます。
11. ディレクトリー以外の場合。他のディレクトリーとのリンクは 1 つしか設定できません。
12. QSYS.LIB および独立 ASP QSYS.LIB ファイル・システム内のユーザー・スペースは、ストリーム・ファイルの入出力をサポートします。
13. 統合ファイル・システム API は、スレッド・セーフ・ファイル・ストリームに存在するオブジェクトに操作が向けられている場合、スレッド・セーフです。複数のスレッドがジョブで実行している場合、スレッド・セーフでないファイル・システムのオブジェクト上でこれらの API が実行されると、 API は失敗します。
14. QSYS.LIB および独立 ASP QSYS.LIB ファイル・システムは、ルート、UDFS、および QOpenSys ファイル・システム以外のオブジェクト・タイプのジャーナル処理をサポートすることができます。 QSYS.LIB または独立 ASP QSYS.LIB ファイル・システム内にあるジャーナル処理オブジェクトについての詳細は、 iSeries Information Center の『ジャーナル管理』のトピックを参照してください。
15. *TYPE2 ディレクトリーには、オブジェクトごとに 100 万リンクの制限があります。 *TYPE1 ディレクトリーには、オブジェクトごとに 32,767 リンクの制限があります。詳細については、『*TYPE2 ディレクトリー』を参照してください。
16. この列内のデータは、QSYS.LIB ファイル・システムと独立 ASP QSYS.LIB ファイル・システムの両方を指しています。

略語

B = バイト　KB = キロバイト　MB = メガバイト　GB = ギガバイト

表 5. ファイル・システムの要約 (2/2)

機能	QOPT	QFileSvr.400	UDFS	NFS	QNetWare
OS/400 の標準機能	Yes	Yes	Yes	Yes	No
ファイルのタイプ	ストリーム	ストリーム	ストリーム	ストリーム	ストリーム

表 5. ファイル・システムの要約 (2/2) (続き)

機能	QOPT	QFileSvr.400	UDFS	NFS	QNetWare
OfficeVision との統合 (たとえば、ファイルがメールできる)	No	No	No	No	No
OS/400 ファイル・サーバーによるアクセス	Yes	Yes	Yes	Yes	Yes
統合 PC サーバー (FSIOP) を介した直接アクセス [1]	No	No	No	No	Yes
オープン / クローズの相対的な速度	低	低 [2]	中 [2]	中 [2]	高 [11]
英語の大文字小文字を区別した名前の検索	No	No [2]	Yes [12]	不定 [2]	No
パス名の各構成要素の最大長	不定 [4]	不定 [2]	255 文字	不定 [2]	255 文字 [13]
パス名の最大長	294 文字	制限なし [2]	16MB	制限なし [2]	255 文字
オブジェクトの拡張属性の最大長	8MB	0 [6]	2GB [10]	0 [6]	64KB
ファイル・システム内のディレクトリー階層の最高レベル	制限なし [7]	制限なし [2]	制限なし [7]	制限なし [2]	100
オブジェクトごとのリンクの最大数 [7]	1	1	不定 [15]	不定 [2]	1
シンボリック・リンクのサポート	No	No	Yes	Yes [2]	No
オブジェクト / ファイルの所有者の設定	No	No [9]	Yes	Yes [2]	Yes
統合ファイル・システム・コマンドのサポート	Yes	Yes	Yes	Yes	Yes
統合ファイル・システム API のサポート	Yes	Yes	Yes	Yes	Yes
階層ファイル・システム (HFS) API のサポート	Yes	No	No	No [2]	No
スレッド・セーフ [14]	Yes	No	Yes	No	No
オブジェクト・ジャーナル処理のサポート	No	No	Yes	No	No

表 5. ファイル・システムの要約 (2/2) (続き)

機能	QOPT	QFileSvr.400	UDFS	NFS	QNetWare

注:

1. ファイル・サーバー I/O プロセッサーは、LAN サーバーが使用するハードウェアです。
2. どのリモート・ファイル・システムにアクセスしているかによって異なります。
3. OS/400 ファイル・サーバーによりアクセスした場合。
4. 詳細については、 85 ページの『光ファイル・システム (QOPT)』を参照してください。
5. / で始まり、ファイル・システム名が続く絶対パス名を前提とします。
6. QFileSvr.400 ファイル・システムは、アクセス中のファイル・システムが拡張属性をサポートする場合でも、拡張属性を戻しません。
7. 実際には、ディレクトリー・レベルは、プログラムおよびシステムのスペース制限によって制限されます。
8. ディレクトリー以外の場合。他のディレクトリーとのリンクは 1 つしか設定できません。
9. アクセスされるファイル・システムは、オブジェクト所有者をサポートする可能性があります。
10. オブジェクトの拡張属性の最大長は、40 バイトを超えることができません。
11. Novell NetWare クライアント PC を介してアクセスされる場合。 iSeries API を使用したアクセスは、比較的遅くなります。
12. 大文字小文字の区別を UDFS 作成時に指定できます。 *MIXED パラメーターが UDFS 作成時に使用される場合、大文字小文字を区別する検索が許可されます。
13. NetWare ディレクトリー・サービスは最大 255 文字です。ファイルおよびディレクトリーは、DOS 8.3 形式に限定されます。
14. 統合ファイル・システム API は、マルチスレッド機能のあるプロセスでアクセスされると、スレッド・セーフです。ファイル・システムはスレッド・セーフでないファイル・システムへのアクセスを許可しません。
15. *TYPE2 ディレクトリーには、オブジェクトごとに 100 万リンクの制限があります。 *TYPE1 ディレクトリーには、オブジェクトごとに 32,767 リンクの制限があります。詳細については、『*TYPE2 ディレクトリー』を参照してください。

略語

B = バイト　KB = キロバイト　MB = メガバイト　GB = ギガバイト

「ルート (/)」ファイル・システム

「ルート (/)」ファイル・システムは、統合ファイル・システムのストリーム・ファイル・サポートと、階層ディレクトリー構造を有効に利用します。「ルート (/)」ファイル・システムには、ディスク・オペレーティング・システム (DOS) と OS/2 ファイル・システムの特性があります。

さらに、

- ストリーム・ファイル入出力用に最適化されています。
- 複数のハード・リンクおよびシンボリック・リンクをサポートします。
- ローカル・ソケットをサポートします。
- *OOPOOL オブジェクトをサポートします。
- スレッド・セーフ・アプリケーション・プログラム・インターフェース (API) をサポートします。
- *FIFO オブジェクトをサポートします。

| • /dev/null および /dev/zero の *CHRSF オブジェクトに加えて、その他の *CHRSF オブジェクトをサポートします。
| • オブジェクト変更のジャーナル処理をサポートします。

| 「ルート (/)」ファイル・システムには、 /dev/null および /dev/zero という文字特殊ファイル (*CHRSF) 用のサポートがあります。文字特殊ファイルは、コンピューター・システムの装置またはリソースに関連付けられます。それらはディレクトリーに表されるパス名を持ち、通常のファイルと同じアクセス保護があります。 /dev/null または /dev/zero 文字特殊ファイルは常に空であり、 /dev/null または /dev/zero に書き込まれるデータは破棄されます。ファイル /dev/null および /dev/zero のオブジェクト・タイプは *CHRSF であり、通常のファイルと同様に使用できます。ただし、/dev/null ファイル内のデータは読み取られず、 /dev/zero ファイルは常にデータをゼロにクリアして正常に戻されます。

「ルート (/)」ファイル・システムについての詳細は、 「ルート (/)」ファイル・システムの使用を参照してください。

「ルート (/)」ファイル・システムの使用

「ルート (/)」ファイル・システムには、 OS/400 ファイル・サーバーまたは統合ファイル・システムのコマンド、ユーザー表示画面、および C 言語 API を使用して、統合ファイル・システム・インターフェースによりアクセスすることができます。

「ルート (/)」ファイル・システムでの大文字小文字の区別

ファイル・システムは、オブジェクト名の大文字と小文字を、入力された状態で保持しますが、サーバーが名前を検索するときには大文字と小文字の区別はしません。

「ルート (/)」ファイル・システムでのパス名

- パス名の形式は、次のとおりです。

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

- 共用名の後のパス名の構成要素は、Windows NT のパス名規則に従います。結合されたディレクトリー名およびオブジェクト名は、合計で 255 文字のパス長さまで満たすことができます。
- ディレクトリー階層の深さには、プログラムおよびサーバーのスペース制限以外の制限はありません。
| • 名前に使用されている文字は、名前が保管されるときに UCS2 のレベル 1 形式 (*TYPE1 ディレクトリーの場合) および UTF-16 (*TYPE2 ディレクトリーの場合) に変換されます (24 ページの『名前の継続性』を参照)。ディレクトリー・フォーマットについての詳細は、 『*TYPE2 ディレクトリー』を参照してください。

「ルート (/)」ファイル・システムでのリンク

「ルート (/)」ファイル・システムでは、 1 つのオブジェクトに複数のハード・リンクを設定することができます。シンボリック・リンクは、完全にサポートされています。シンボリック・リンクは、「ルート (/)」ファイル・システムと別のファイル・システム (QSYS.LIB、独立 ASP QSYS.LIB、または QDLS など) のオブジェクトとの間のリンクに使用することができます。

リンクについては、 19 ページの『リンク』を参照してください。

「ルート (/)」ファイル・システムでの統合ファイル・システム・コマンドの使用

「ルート (/)」ファイル・システムでは、 28 ページの『CL コマンドを使用した操作の実行』にリストしてあるすべてのコマンドと、 27 ページの『iSeries メニューおよび表示画面を使用した操作の実行』に説明してある表示画面を使用することができます。しかし、マルチスレッド可能プロセスでこれらのコマンドを使用するのは、安全性を考えると避けた方がよい場合があります。

「ルート (/)」ファイル・システムでの統合ファイル・システム API の使用

「ルート (/)」ファイル・システムでは、 54 ページの『API を使用した操作の実行』 にリストしてあるすべての C 言語 API を使用することができます。

「ルート (/)」ファイル・システムでのオブジェクト変更のジャーナル処理

「ルート (/)」ファイル・システムでのオブジェクトをジャーナル処理することができます。ジャーナル管理の主な目的は、オブジェクトの最後の保管以降にそのオブジェクトに加えられた変更を回復できるようにすることです。「ルート (/)」ファイル・システムでのオブジェクト変更のジャーナル処理についての詳細は、 101 ページの『第 7 章 統合ファイル・システム・オブジェクトのジャーナル処理サポート』を参照してください。

オープン・システム・ファイル・システム (QOpenSys)

QOpenSys ファイル・システムは、 POSIX や XPG などの UNIX ベースのオープン・システム標準との互換性があります。「ルート (/)」ファイル・システムと同様に、このファイル・システムは統合ファイル・システムが提供するストリーム・ファイルおよびディレクトリーのサポートを利用します。

さらに、

- UNIX システムと同様の階層ディレクトリー構造によりアクセスされます。
- ストリーム・ファイル入出力用に最適化されています。
- 複数のハード・リンクおよびシンボリック・リンクをサポートします。
- 名前の大文字と小文字を区別します。
- ローカル・ソケットをサポートします。
- *OOPOOL オブジェクトをサポートします。
- スレッド・セーフ API をサポートします。
- *FIFO オブジェクトをサポートします。
- オブジェクト変更のジャーナル処理をサポートします。

QOpenSys システムの特性は、「ルート (/)」ファイル・システムと同じですが、 UNIX ベースのオープン・システム標準をサポートするために、大文字小文字の区別を行います。

QOpenSys についての詳細は、『QOpenSys の使用』を参照してください。

*TYPE 1 ディレクトリーから *TYPE2 ディレクトリーへの変換および QOpenSys ファイル・システムの制限事項については、 『*TYPE2 ディレクトリー』を参照してください。

QOpenSys の使用

QOpenSys には、OS/400 ファイル・サーバーまたは統合ファイル・システムのコマンド、ユーザー表示画面、および C 言語 API を使用して、統合ファイル・システム・インターフェースによりアクセスすることができます。

QOpenSys ファイル・システムでの大文字小文字の区別

「ルート (/)」ファイル・システムとは異なり、 QOpenSys ファイル・システムは、オブジェクト名の探索時に大文字と小文字を区別します。たとえば、すべてが大文字で指定された文字ストリングは、どれか一文字でも小文字になっている文字ストリングとは一致しません。

大文字と小文字が区別されるため、大文字と小文字という違いがあれば、同じ名前を重複して使用することができます。たとえば、QOpenSys の中の同じディレクトリーに、 Payroll、PayRoll、PAYROLL という名前で 3 つのオブジェクトを持つことができます。

QOpenSys ファイル・システムでのパス名

- パス名の形式は、次のとおりです。

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

- パス名の各構成要素は、255 文字までの長さにすることができます。全パス名は、16 メガバイトまでの長さにすることができます。
- ディレクトリー階層の深さには、プログラムおよびサーバーのスペース制限以外の制限はありません。
- 名前に使用されている文字は、名前が保管されるときに UCS2 のレベル 1 形式 (*TYPE1 ディレクトリーの場合) および UTF-16 (*TYPE2 ディレクトリーの場合) に変換されます (24 ページの『名前の継続性』を参照)。ディレクトリー・フォーマットについての詳細は、 『*TYPE2 ディレクトリー』を参照してください。

QOpenSys ファイル・システムでのリンク

QOpenSys ファイル・システムでは、 1 つのオブジェクトに複数のハード・リンクを設定することができます。シンボリック・リンクは、完全にサポートされています。シンボリック・リンクは、 QOpenSys ファイル・システムと別のファイル・システムのオブジェクトとの間のリンクに使用できます。

リンクについては、 19 ページの『リンク』を参照してください。

QOpenSys ファイル・システムでの統合ファイル・システム・コマンドおよび表示画面の使用

QOpenSys ファイル・システムでは、 28 ページの『CL コマンドを使用した操作の実行』 にリストしてあるすべてのコマンドと、 27 ページの『iSeries メニューおよび表示画面を使用した操作の実行』 に説明してある表示画面を使用することができます。しかし、マルチスレッド可能プロセスでこれらのコマンドを使用するのは、安全性を考えると避けた方がよい場合があります。

QOpenSys ファイル・システムでの統合ファイル・システム API の使用

スレッド・セーフ方式の QOpenSys ファイル・システムでは、 54 ページの『API を使用した操作の実行』 にリストしてあるすべての C 言語関数を使用することができます。

QOpenSys ファイル・システムでのオブジェクト変更のジャーナル処理

QOpenSys ファイル・システムでのオブジェクトをジャーナル処理することができます。ジャーナル管理の主な目的は、オブジェクトの最後の保管以降にそのオブジェクトに加えられた変更を回復できるようにすることです。 QOpenSys ファイル・システムでのオブジェクト変更のジャーナル処理についての詳細は、101 ページの『第 7 章 統合ファイル・システム・オブジェクトのジャーナル処理サポート』を参照してください。

ユーザー定義ファイル・システム (UDFS)

UDFS ファイル・システムは、ユーザーが選択した補助記憶域プール (ASP)、または独立補助記憶域プール (ASP) にあります。ユーザーがこれらのファイル・システムを作成し、管理します。

さらに、次のことを行います。

- DOS や OS/2 などの PC オペレーティング・システムと同様の階層ディレクトリー構造を提供します。
- ストリーム・ファイル入出力用に最適化されています。

- 複数のハード・リンクおよびシンボリック・リンクをサポートします。
- ローカル・ソケットをサポートします。
- スレッド・セーフ API をサポートします。
- *FIFO オブジェクトをサポートします。
- オブジェクト変更のジャーナル処理をサポートします。

それぞれに固有の名前を指定することによって、複数の UDFS を作成できます。作成中に、次のものを含む他の属性も指定できます。

- UDFS に位置するオブジェクトが保管される、ASP 番号または独立 ASP 名。
- UDFS 内に位置するオブジェクト名の大文字小文字を区別する特性。

  UDFS の大文字小文字の区別は、UDFS 内のオブジェクト名の検索時に、大文字と小文字がどちらも一致するかどうかを判別します。

ユーザー定義のファイル・システムについての詳細は、以下のトピックを参照してください。

- UDFS の概念
- 統合ファイル・システム・インターフェースを介した UDFS の使用

UDFS の概念

UDFS では、「ルート (/)」および QOpenSys ファイル・システムの場合と同様、ディレクトリー、ストリーム・ファイル、シンボリック・リンク、ローカル・ソケット、および SOM オブジェクトを作成できます。

単一のブロック特殊ファイル・オブジェクト (*BLKSF) は UDFS を表します。 UDFS を作成すると、自動的にブロック特殊ファイルも作成することになります。ブロック特殊ファイルは、統合ファイル・システム総称コマンド、API、および QFileSvr.400 インターフェースを介してのみ、ユーザーからアクセスできます。

UDFS は、2 つの状態、**マウント**および**アンマウント**でのみ存在します。 UDFS をマウントすると、その中のオブジェクトはアクセス可能になります。 UDFS をアンマウントすると、その中のオブジェクトはアクセス不可になります。

UDFS 内のオブジェクトにアクセスするには、ディレクトリー (たとえば、 /home/JON) 上に UDFS をマウントする必要があります。ディレクトリー上で UDFS をマウントすると、オブジェクトおよびサブディレクトリーを含めて、そのディレクトリーの元の内容がアクセス不可になります。 UDFS をマウントすると、UDFS の内容は UDFS をマウントしたディレクトリー・パスを介して、アクセス可能になります。たとえば、/home/JON ディレクトリーに、ファイル /home/JON/payroll が入っているとします。 UDFS には 3 つのディレクトリー mail、action、および outgoing が入っています。 /home/JON で UDFS をマウントした後、 /home/JON/payroll ファイルはアクセス不可になり、 3 つの UDFS ディレクトリーは /home/JON/mail、 /home/JON/action、および /home/JON/outgoing としてアクセス可能になります。 UDFS のマウントを解除した後、 /home/JON/payroll ファイルは再びアクセス可能になり、 UDFS の 3 つのディレクトリーはアクセス不可になります。

注: 独立 ASP 上の UDF を上書きマウントすることはできません。

ファイル・システムのマウントについての詳細は、OS/400 ネットワーク・ファイル・システム・サポート

を参照してください。

統合ファイル・システム・インターフェースを介した UDFS の使用

UDFS は、OS/400 ファイル・サーバーまたは統合ファイル・システムのいずれかのコマンド、ユーザー表示画面、および API を使用した統合ファイル・システム・インターフェースを介してアクセスできます。統合ファイル・システム・インターフェースを使用する際には、以下の考慮事項および制限事項に注意してください。

統合ファイル・システム UDFS での大文字小文字の区別

UDFS のオブジェクト名の作成時に、大文字小文字を区別するか、または大文字小文字を区別しないかを指定できます。

大文字小文字の区別を選択すると、オブジェクト名の検索時に大文字小文字が区別されます。たとえば、すべてが大文字で指定された名前は、どれか 1 文字でも小文字になっている名前とは一致しません。したがって、 /home/MURPH/ と /home/murph/ は異なるディレクトリーとして識別されます。大文字小文字を区別する UDFS を作成するには、CRTUDFS コマンドの使用時に、 CASE パラメーターに *MIXED を指定することができます。

大文字小文字の区別なしを選択する場合、サーバーは検索中に名前の大文字と小文字を区別しません。したがって、サーバーは /home/CAYCE と /HOME/cayce を 2 つの別個のディレクトリーではなく、同じディレクトリーとして識別します。大文字小文字を区別しない UDFS を作成するには、CRTUDFS コマンドの使用時に、 CASE パラメーターに *MONO を指定することができます。

どちらの場合でも、ファイル・システムはユーザーがオブジェクト名を入力するのと同じ形で大文字および小文字を保管します。大文字小文字の区別オプションは、サーバーを介してユーザーが名前を検索する方法にのみ適用されます。

統合ファイル・システム UDFS でのパス名

| ブロック特殊ファイル (*BLKSF) は、 UDFS 全体およびその中のすべてのオブジェクトを操作する必要がある場合に、UDFS を表します。 UDFS がシステムまたは基本ユーザー ASP 上に存在する場合、ブロック特殊ファイル名は、以下の形式でなければなりません。

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

| ここで、XX は UDFS を保管する ASP 番号、 udfs_name はその ASP 内の UDFS の固有名です。 UDFS 名が必ず .udfs という拡張子で終わらなければならないことに注意してください。

| UDFS が独立 ASP 上に存在する場合、ブロック特殊ファイル名は、以下の形式でなければなりません。

```
/dev/asp_name/udfs_name.udfs
```

| ここで、asp_name は UDFS を保管する独立 ASP の名前で、 udfs_name はその独立 ASP 内の UDFS の固有名です。 UDFS 名が必ず .udfs という拡張子で終わらなければならないことに注意してください。

UDFS 内のオブジェクトのパス名は、UDFS をマウントするディレクトリーに対する相対パス名です。たとえば、 UDFS /dev/qasp01/wysocki.udfs を /home/dennis のもとでマウントする場合、 UDFS 内のすべてのオブジェクトのパス名は、/home/dennis で始まります。

| 追加のパス名規則は、以下のとおりです。

| • パス名の各構成要素は、255 文字までの長さにすることができます。全パス名は、16 メガバイトまでの長さにすることができます。
| • ディレクトリー階層の深さには、プログラムおよびサーバーのスペース制限以外の制限はありません。

| • 名前に使用されている文字は、名前が保管されるときに UCS2 のレベル 1 形式 (*TYPE1 ディレクトリー
| の場合) および UTF-16 (*TYPE2 ディレクトリーの場合) に変換されます (24 ページの『名前の継続性』を参照)。ディレクトリー・フォーマットについての詳細は、 『*TYPE2 ディレクトリー』を参照してください。

統合ファイル・システム UDFS でのリンク

UDFS 内のオブジェクトにより、同じオブジェクトに対する複数のハード・リンクが可能になり、シンボリック・リンクを完全にサポートします。シンボリック・リンクにより、 UDFS から別のファイル・システムのオブジェクトへのリンクを作成することができます。

リンクについては、 19 ページの『リンク』を参照してください。

UDFS での統合ファイル・システム・コマンドの使用

ユーザー定義ファイル・システムでは、 28 ページの『CL コマンドを使用した操作の実行』 にリストしてあるすべてのコマンドと、 27 ページの『iSeries メニューおよび表示画面を使用した操作の実行』 に説明してある表示画面を使用することができます。ユーザー定義ファイル・システムおよび他の一般のマウント・ファイル・システムに特有の CL コマンドがいくつかあります。次の表で、それらを説明します。

表 6. ユーザー定義ファイル・システムの CL コマンド

コマンド	説明
ADDMFS	マウント・ファイル・システムの追加。ローカル・クライアント・ディレクトリー上に、エクスポートされたリモート・サーバー・ファイル・システムを配置する。
CRTUDFS	UDFS の作成。ユーザー定義ファイル・システムを作成する。
DLTUDFS	UDFS の削除。ユーザー定義ファイル・システムを削除する。
DSPMFSINF	マウント・ファイル・システム情報の表示。マウント・ファイル・システムについての情報を表示する。
DSPUDFS	UDFS の表示。ユーザー定義ファイル・システムについての情報を表示する。
MOUNT	ファイル・システムのマウント。ローカル・クライアント・ディレクトリー上に、エクスポートされたリモート・サーバー・ファイル・システムを配置する。このコマンドは、ADDMFS コマンドの別名である。
RMVMFS	マウント・ファイル・システムの除去。ローカル・クライアント・ネーム・スペースから、エクスポートされたリモート・サーバー・ファイル・システムを除去する。
UNMOUNT	ファイル・システムのアンマウント。ローカル・クライアント・ネーム・スペースから、エクスポートされたリモート・サーバー・ファイル・システムを除去する。このコマンドは、RMVMFS コマンドの別名である。

注: 統合ファイル・システム・コマンドを、 UDFS 上に保管されるオブジェクト上で操作する前に、その UDFS をマウントする必要があります。

UDFS での統合ファイル・システム API の使用

ユーザー定義ファイル・システムでは、 54 ページの『API を使用した操作の実行』 にリストしてあるすべての C 言語関数を使用することができます。

注: 統合ファイル・システム・コマンドを、 UDFS 上に保管されるオブジェクト上で操作する前に、その UDFS をマウントする必要があります。

UDFS のグラフィカル・ユーザー・インターフェース

iSeries ナビゲーター (PC 上のグラフィカル・ユーザー・インターフェース) により、 UDFS に簡単かつ便利にアクセスできます。このインターフェースによって、Windows クライアントから、 UDFS を作成、削除、表示、マウント、およびアンマウントすることができます。

iSeries ナビゲーターを介して UDFS に対する操作を実行できます。基本タスクには、次のものが含まれます。

- 45 ページの『新規のユーザー定義ファイル・システムの作成』
- 46 ページの『ユーザー定義ファイル・システムのマウント』
- 46 ページの『ユーザー定義ファイル・システムのアンマウント』

統合ファイル・システム UDFS の作成

ユーザー定義ファイル・システムの作成 (CRTUDFS) は、統合ファイル・システム・ネーム・スペース、API、および CL コマンドを介して可視にできるファイル・システムを作成します。 ADDMFS または MOUNT コマンドは、 UDFS を既存のローカル・ディレクトリーの 『一番上に』 置きます。ユーザーが選択した ASP または独立 ASP で UDFS を作成することができます。また、大文字小文字の区別を指定することもできます。

統合ファイル・システム UDFS の削除

ユーザー定義ファイル・システムの削除 (DLTUDFS) コマンドは、既存のアンマウントした UDFS と、その中のすべてのオブジェクトを削除します。マウントした UDFS があると、コマンドは失敗します。 UDFS の削除によって、UDFS 内のすべてのオブジェクトも削除されます。 UDFS 内のすべてのオブジェクトを削除する適切な権限がない場合、どのオブジェクトも削除されることはありません。

統合ファイル・システム UDFS の表示

ユーザー定義ファイル・システムの表示 (DSPUDFS) コマンドは、既存の UDFS の属性、マウントされているかいないかを表示します。マウント・ファイル・システムの情報の表示 (DSPMFSINF) コマンドも、マウントされた UDFS と、マウント・ファイル・システムについての情報を表示します。

統合ファイル・システム UDFS のマウント

マウント・ファイル・システムの追加 (ADDMFS) および MOUNT コマンドは、ファイル・システム中のオブジェクトを、統合ファイル・システムのネーム・スペースからアクセス可能にします。 UDFS をマウントするには、 ADDMFS コマンド上の TYPE パラメーターに *UDFS を指定する必要があります。

注: 独立 ASP 上の UDF を上書きマウントすることはできません。

統合ファイル・システム UDFS のアンマウント

アンマウント・コマンドは、UDFS の内容を、統合ファイル・システム・インターフェースからアクセス不可にします。 UDFS 中のオブジェクトは、一度 UDFS がアンマウントされると、個別にアクセス可能になることはありません。マウント・ファイル・システムの除去 (RMVMFS) または UNMOUNT コマンドは、マウントされたファイル・システムを、統合ファイル・システムのネーム・スペースからアクセス不可にします。コマンド使用時にファイル・システム中のオブジェクトのいずれかが使用中である場合 (たとえばファイルがオープンしている場合)、エラー・メッセージを受け取ります。 UDFS はマウントされたままです。 UDFS の一部が上書きマウントされている場合、この UDFS は上書きしているファイル・システムを外さない限りアンマウントできません。

たとえば、UDFS /dev/qasp02/jenn.udfs を、統合ファイル・システム・ネーム・スペースの /home/judy にマウントしたとします。その後別のファイル・システム /pubs を /home/judy にマウントすると、

jenn.udfs の内容はアクセス不可になります。さらに、/home/judy から 2 番目のファイル・システムをアンマウントしない限り、 jenn.udfs をアンマウントすることはできません。

注: 独立 ASP 上の UDF を上書きマウントすることはできません。

統合ファイル・システム UDFS の保管および復元

すべての UDFS オブジェクト、およびそれに関連した権限を保管し、復元する機能があります。保管コマンド (SAV) によって UDFS 中のオブジェクトを保管できる一方、復元コマンド (RST) によって UDFS オブジェクトを復元することができます。両方のコマンドは、UDFS がマウントされているかどうかにかかわらず機能します。ただし、単に UDFS 内のオブジェクトではなく、UDFS 属性を正しく保管する場合は、 UDFS をアンマウントしないでください。

UDFS ファイル・システムでのオブジェクト変更のジャーナル処理

ユーザー定義のファイル・システムでのオブジェクトをジャーナル処理することができます。ジャーナル管理の主な目的は、オブジェクトの最後の保管以降にそのオブジェクトに加えられた変更を回復できるようにすることです。 UDFS ファイル・システムでのオブジェクト変更のジャーナル処理についての詳細は、101 ページの『第 7 章 統合ファイル・システム・オブジェクトのジャーナル処理サポート』を参照してください。

ライブラリー・ファイル・システム (QSYS.LIB)

QSYS.LIB ファイル・システムは、iSeries サーバー・ライブラリー構造をサポートします。このファイル・システムは、データベース・ファイルと、ライブラリー・サポートがシステムおよび基本ユーザー ASP 内で管理する、他のすべての iSeries サーバー・オブジェクト・タイプへのアクセスを提供します。

さらに、

- iSeries サーバー・ライブラリーとその中のオブジェクトを操作する、ユーザー・インターフェースおよびプログラミング・インターフェースを、すべてサポートします。
- データベース・ファイルを操作するプログラミング言語および機能を、すべてサポートします。
- iSeries サーバー・オブジェクトを管理するための、広範な管理サポートを提供します。
- 物理ファイル・メンバー、ユーザー・スペース、および保管ファイル上のストリーム入出力をサポートします。

OS/400 のバージョン 3 より前には、QSYS.LIB ファイル・システムが、 iSeries サーバー・システムそのもの であると考えられていました。 RPG または COBOL などのプログラミング言語や、アプリケーションの開発 DDS のような機能を使用していたプログラマーは、 QSYS.LIB ファイル・システムを使用していました。コマンド、メニュー、および表示画面を使用して、出力待ち行列を操作していたシステム・オペレーターや、ユーザー・プロファイルの作成および変更を行っていたシステム管理者も、 QSYS.LIB ファイル・システムを使用していました。

これらの機能およびこれらの機能にもとづくアプリケーションは、すべて 統合ファイル・システムの導入前と同様に操作できます。ただし、これらの機能では、統合ファイル・システム・インターフェースから QSYS.LIB にアクセスすることはできません。

QSYS.LIB についての詳細は、 『統合ファイル・システム・インターフェースを介した QSYS.LIB の使用』を参照してください。

統合ファイル・システム・インターフェースを介した QSYS.LIB の使用

QSYS.LIB ファイル・システムは、OS/400 ファイル・サーバーまたは統合ファイル・システムのコマンド、ユーザー表示画面、および C 言語 API を使用して、統合ファイル・システム・インターフェースによりアクセスすることができます。これらの統合ファイル・システム・インターフェースを使用する際には、次に挙げる考慮事項および制限事項に注意してください。

QSYS.LIB ファイル・システムの QPWFSERVER 権限リスト

QPWFSERVER は、リモート・クライアントを介してアクセスされる QSYS.LIB ファイル・システムにあるすべてのオブジェクトに、追加のアクセスを提供する権限リストです。この権限リストで指定された権限は、 QSYS.LIB ファイル・システム内のすべてのオブジェクトに適用されます。

このオブジェクトに対するデフォルト権限は PUBLIC *USE 権限です。管理者は、EDTAUTL (権限リストの編集) または WRKAUTL (権限リストの処理) コマンドを使用して、この権限の値を変更することができます。管理者は、一般人がリモート・クライアントから QSYS.LIB オブジェクトにアクセスできないように、 PUBLIC *EXCLUDE 権限を権限リストに割り当てることができます。

QSYS.LIB ファイル・システムでのファイル処理についての制限事項

- 論理ファイルはサポートされていません。
- テキスト・モード・アクセス用にサポートされている物理ファイルは、 1 つのフィールドを含むプログラム記述物理ファイル、および 1 つのテキスト・フィールドを含むソース物理ファイルのみです。バイナリー・モード・アクセス用にサポートされる物理ファイルには、テキスト・モード・アクセス用にサポートされるこれらのファイルに加えて、外部記述の物理ファイルが含まれます。
- バイト範囲のロックは、サポートされていません。 (バイト範囲ロックについての詳細は、 iSeries Information Center の 『fcntl()』のトピックを参照してください。)
- ジョブがデータベース・ファイル・メンバーをオープンする場合、そのファイル・メンバーへの書き込みアクセス権は、常に 1 つのジョブにしか与えられません。それ以外の要求には、読み取りアクセス権だけが認められます。

QSYS.LIB ファイル・システムでのユーザー・スペースのサポート

QSYS.LIB は、ユーザー・スペース・オブジェクトへのストリーム入出力操作をサポートします。たとえば、プログラムでユーザー・スペースにストリーム・データを書き込んだり、ユーザー・スペースからデータを読み取ったりできます。ユーザー・スペースの最大サイズは、16 776 704 バイトです。

ユーザー・スペースは CCSID (コード化文字セット ID) でタグ付けされない点に注意してください。このため、戻される CCSID は、ジョブのデフォルト CCSID です。

QSYS.LIB ファイル・システムでの保管ファイルのサポート

QSYS.LIB ファイル・システムは、ファイル・オブジェクトを保管するためのストリーム入出力操作をサポートします。たとえば、既存の保管ファイルには、別の、既存の、そして空の保管ファイル・オブジェクトに移動させることが必要になるまで、読み取りや他のファイルへのコピーが可能なデータが入っています。保管ファイルが書き込みのためにオープンしている場合、そのファイルについての他のオープン・インスタンスは許可されません。保管ファイルでは、複数のファイルのインスタンスが読み取り用にオープンされているジョブがない場合、読み取り用に複数のインスタンスをオープンすることが**可能です**。保管ファイルを読み取り / 書き込みアクセスのためにオープンすることはできません。 1 つのジョブで複数のスレッドが実行されているときは、保管ファイル・データへのストリーム入出力操作を行うことはできません。

保管ファイルまたはそのディレクトリーがネットワーク・ファイル・システムのサーバーを介してエクスポートされる場合には、保管ファイル上でのストリーム入出力操作はサポートされていません。しかし、PC クライアントから、または QFileSvr.400 ファイル・システムを介して、それらにアクセスすることは可能です。

QSYS.LIB ファイル・システムでの大文字小文字の区別

一般に、QSYS.LIB ファイル・システムでは、オブジェクトの名前の大文字と小文字を区別しません。オブジェクト名の検索は、大文字と小文字のどちらで行っても同じです。

ただし、名前が引用符で囲まれていれば、名前の大文字小文字が区別されます。したがって、引用符で囲まれた名前の検索については、大文字と小文字が区別されます。

QSYS.LIB ファイル・システムでのパス名

- パス名の各構成要素には、オブジェクト名とオブジェクト・タイプが含まれていなければなりません。次はその一例です。

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

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

  オブジェクト名とオブジェクト・タイプは、ピリオド (.) で区切ります。オブジェクト・タイプが異なっていれば、 1 つのライブラリーに同じ名前の複数のオブジェクトを入れることができます。したがって、そのオブジェクトを固有のものとして識別するために、必ずオブジェクト・タイプを指定してください。
- 各構成要素のオブジェクト名は 10 文字まで、オブジェクト・タイプは 6 文字までの長さにすることができます。
- QSYS.LIB 内のディレクトリー階層は、アクセスされるオブジェクトのタイプによって、 2 レベルまたは 3 レベルの深さ (パス名の構成要素が 2 つまたは 3 つ) のいずれかになります。オブジェクトがデータベース・ファイルであれば、階層は 3 レベル (ライブラリー、ファイル、メンバー) になります。それ以外の場合には、2 レベル (ライブラリー、オブジェクト) のみになります。各構成要素名の長さとディレクトリーのレベル数の組み合わせによって、パス名の最大長が決まります。

  最初の 2 レベルに「ルート (/)」および QSYS.LIB が含まれていれば、 QSYS.LIB のディレクトリー階層は、5 レベルまでの深さにすることができます。
- 名前に使用されている文字は、名前が保管されるときに、CCSID 37 に変換されます。ただし、引用符で囲まれた名前は、ジョブの CCSID で保管されます。

  CCSID についての詳細は、 iSeries Information Center にある『グローバリゼーション』のトピックを参照してください。

QSYS.LIB ファイル・システムでのリンク

QSYS.LIB ファイル・システムでは、シンボリック・リンクを作成、保管することはできません。

ライブラリーとその中のオブジェクトとの関係は、ライブラリーと各オブジェクトとの間に 1 つのハード・リンクが設定されているのと同等です。統合ファイル・システムは、ライブラリーとオブジェクトとの関係を、リンクとして扱います。したがって、シンボリック・リンクをサポートするファイル・システムから、 QSYS.LIB ファイル・システムのオブジェクトにリンクすることが可能です。

リンクについては、 19 ページの『リンク』を参照してください。

QSYS.LIB ファイル・システムでの統合ファイル・システム・コマンドおよび表示画面の使用

28 ページの『CL コマンドを使用した操作の実行』にリストしてあるコマンドは、次の場合を除いて、QSYS.LIB ファイル・システムで使用することができます。

- ADDLNK コマンドは、QSYS.LIB のオブジェクトにシンボリック・リンクを作成する場合のみ使用できます。
- ファイル操作は、プログラム記述物理ファイルとソース物理ファイルに対してのみ行うことができます。
- STRJRN および ENDJRN コマンドは、データベース物理ファイル上では使用できません。

27 ページの『iSeries メニューおよび表示画面を使用した操作の実行』で説明したユーザー表示画面にも、同じ制限があります。

QSYS.LIB ファイル・システムでの統合ファイル・システム API の使用

54 ページの『API を使用した操作の実行』 にリストしてある C 言語関数は、次のものを除き、QSYS.LIB ファイル・システムで使用することができます。

- ファイル操作は、プログラム記述物理ファイルとソース物理ファイルに対してのみ行うことができます。
- symlink() 関数は、シンボリック・リンクをサポートする別のファイル・システムから QSYS.LIB のオブジェクトにリンクする場合のみ使用できます。
- QjoStartJournal() および QjoEndJournal() API は、データベース物理ファイル上では使用できません。

独立 ASP QSYS.LIB

独立 ASP QSYS.LIB ファイル・システムは、ユーザーが作成および定義する独立補助記憶域プール (ASP) 内の iSeries サーバー・ライブラリー構造をサポートします。このファイル・システムは、データベース・ファイルと、ライブラリー・サポートが独立 ASP 内で管理する、他のすべての iSeries サーバー・オブジェクト・タイプへのアクセスを提供します。

さらに、

- 独立 ASP 内で、iSeries サーバー・ライブラリーとその中のオブジェクトを操作する、ユーザー・インターフェースおよびプログラミング・インターフェースを、すべてサポートします。
- データベース・ファイルを操作するプログラミング言語および機能を、すべてサポートします。
- iSeries サーバー・オブジェクトを管理するための、広範な管理サポートを提供します。
- 物理ファイル・メンバー、ユーザー・スペース、および保管ファイル上のストリーム入出力をサポートします。

独立 ASP QSYS.LIB ファイル・システムについての詳細は、 『統合ファイル・システム・インターフェースを介した独立 ASP QSYS.LIB の使用』を参照してください。

統合ファイル・システム・インターフェースを介した独立 ASP QSYS.LIB の使用

独立 ASP QSYS.LIB ファイル・システムは、 OS/400 ファイル・サーバーまたは統合ファイル・システムのいずれかのコマンド、ユーザー表示画面、および C 言語 API を使用した統合ファイル・システム・インターフェースを介してアクセスできます。これらの統合ファイル・システム・インターフェースを使用する際には、次に挙げる考慮事項および制限事項に注意してください。

独立 ASP QSYS.LIB ファイル・システムの QPWFSERVER 権限リスト

QPWFSERVER は、リモート・クライアントを介してアクセスされる独立 ASP QSYS.LIB ファイル・システムにあるすべてのオブジェクトに、追加のアクセスを提供する権限リストです。 この権限リストで指定された権限は、QSYS.LIB ファイル・システム内のすべてのオブジェクトに適用されます。

このオブジェクトに対するデフォルト権限は PUBLIC *USE 権限です。管理者は、EDTAUTL (権限リストの編集) または WRKAUTL (権限リストの処理) コマンドを使用して、この権限の値を変更することができます。管理者は、一般人がリモート・クライアントから独立 ASP QSYS.LIB オブジェクトにアクセスできないように、 PUBLIC *EXCLUDE 権限を権限リストに割り当てることができます。

独立 ASP QSYS.LIB ファイル・システムでのファイル処理についての制約事項

- 論理ファイルはサポートされていません。
- テキスト・モード・アクセス用にサポートされている物理ファイルは、 1 つのフィールドを含むプログラム記述物理ファイル、および 1 つのテキスト・フィールドを含むソース物理ファイルのみです。バイナリー・モード・アクセス用にサポートされる物理ファイルには、テキスト・モード・アクセス用にサポートされるこれらのファイルに加えて、外部記述の物理ファイルが含まれます。
- バイト範囲のロックは、サポートされていません。 (バイト範囲ロックについての詳細は、 iSeries Information Center の 『fcntl()』のトピックを参照してください。)
- ジョブがデータベース・ファイル・メンバーをオープンする場合、そのファイル・メンバーへの書き込みアクセス権は、常に 1 つのジョブにしか与えられません。それ以外の要求には、読み取りアクセス権だけが認められます。

独立 ASP QSYS.LIB ファイル・システムでのユーザー・スペースのサポート

独立 ASP QSYS.LIB は、ユーザー・スペース・オブジェクトへのストリーム入出力操作をサポートします。たとえば、プログラムでユーザー・スペースにストリーム・データを書き込んだり、ユーザー・スペースからデータを読み取ったりできます。ユーザー・スペースの最大サイズは、16 776 704 バイトです。

ユーザー・スペースは CCSID (コード化文字セット ID) でタグ付けされない点に注意してください。このため、戻される CCSID は、ジョブのデフォルト CCSID です。

独立 ASP QSYS.LIB ファイル・システムでの保管ファイルのサポート

独立 ASP QSYS.LIB は、ファイル・オブジェクトを保管するためのストリーム入出力操作をサポートします。たとえば、既存の保管ファイルには、別の、既存の、そして空の保管ファイル・オブジェクトに移動させることが必要になるまで、読み取りや他のファイルへのコピーが可能なデータが入っています。保管ファイルが書き込みのためにオープンしている場合、そのファイルについての他のオープン・インスタンスは許可されません。保管ファイルでは、複数のファイルのインスタンスが読み取り用にオープンされているジョブがない場合、読み取り用に複数のインスタンスをオープンすることが**可能です**。保管ファイルを読み取り / 書き込みアクセスのためにオープンすることはできません。 1 つのジョブで複数のスレッドが実行されているときは、保管ファイル・データへのストリーム入出力操作を行うことはできません。

保管ファイルまたはそのディレクトリーがネットワーク・ファイル・システムのサーバーを介してエクスポートされる場合には、保管ファイル上でのストリーム入出力操作はサポートされていません。しかし、PC クライアントから、または QFileSvr.400 ファイル・システムを介して、それらにアクセスすることは可能です。

独立 ASP QSYS.LIB ファイル・システムでの大文字小文字の区別

一般に、独立 ASP QSYS.LIB ファイル・システムでは、オブジェクトの名前の大文字と小文字を区別しません。オブジェクト名の検索は、大文字と小文字のどちらで行っても同じです。

ただし、名前が引用符で囲まれていれば、名前の大文字小文字が区別されます。したがって、引用符で囲まれた名前の検索については、大文字と小文字が区別されます。

独立 ASP QSYS.LIB ファイル・システムでのパス名

- パス名の各構成要素には、オブジェクト名とオブジェクト・タイプが含まれていなければなりません。次はその一例です。

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

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

  ここで、asp_name は独立 ASP の名前です。オブジェクト名とオブジェクト・タイプは、ピリオド (.) で区切ります。オブジェクト・タイプが異なっていれば、 1 つのライブラリーに同じ名前の複数のオブジェクトを入れることができます。したがって、そのオブジェクトを固有のものとして識別するために、必ずオブジェクト・タイプを指定してください。
- 各構成要素のオブジェクト名は 10 文字まで、オブジェクト・タイプは 6 文字までの長さにすることができます。
- 独立 ASP QSYS.LIB 内のディレクトリー階層は、アクセスされるオブジェクトのタイプによって、 2 レベルまたは 3 レベルの深さ (パス名の構成要素が 2 つまたは 3 つ) のいずれかになります。 オブジェクトがデータベース・ファイルであれば、階層は 3 レベル (ライブラリー、ファイル、メンバー) になります。それ以外の場合には、2 レベル (ライブラリー、オブジェクト) のみになります。各構成要素名の長さとディレクトリーのレベル数の組み合わせによって、パス名の最大長が決まります。

  最初の 3 レベルに /、asp_name、および QSYS.LIB が含まれていれば、独立 ASP QSYS.LIB ファイル・システムのディレクトリー階層は、6 レベルまでの深さにすることができます。
- 名前に使用されている文字は、名前が保管されるときに、CCSID 37 に変換されます。ただし、引用符で囲まれた名前は、ジョブの CCSID で保管されます。

  CCSID についての詳細は、 iSeries Information Center にある『グローバリゼーション』のトピックを参照してください。

独立 ASP QSYS.LIB ファイル・システムでのリンク

独立 ASP QSYS.LIB ファイル・システムでは、シンボリック・リンクを作成、保管することはできません。

ライブラリーとその中のオブジェクトとの関係は、ライブラリーと各オブジェクトとの間に 1 つのハード・リンクが設定されているのと同等です。統合ファイル・システムは、ライブラリーとオブジェクトとの関係を、リンクとして扱います。したがって、シンボリック・リンクをサポートするファイル・システムから、独立 ASP QSYS.LIB ファイル・システムのオブジェクトにリンクすることが可能です。

リンクについては、 19 ページの『リンク』を参照してください。

独立 ASP QSYS.LIB ファイル・システムでの統合ファイル・システム・コマンドおよび表示画面の使用

28 ページの『CL コマンドを使用した操作の実行』にリストしてあるコマンドは、次の場合を除いて、QSYS.LIB ファイル・システムで使用することができます。

- ADDLNK コマンドは、独立 ASP QSYS.LIB のオブジェクトに シンボリック・リンクを作成する場合のみ使用できます。
- ファイル操作は、プログラム記述物理ファイルとソース物理ファイルに対してのみ行うことができます。
- STRJRN および ENDJRN コマンドは、データベース物理ファイル上では使用できません。

- MOV コマンドを使用して基本補助記憶域プール (ASP) に独立 ASP QSYS.LIB ファイル・システム内のライブラリーを移動することはできません。ただし、システム ASP またはその他の独立 ASP に、独立 ASP QSYS.LIB 内のライブラリーを移動することができます。
- SAV または RST を使用して独立 ASP 上にライブラリー・オブジェクトを保管または復元する場合、その独立 ASP が SAV または RST を行うジョブと関連しているか、またはその独立 ASP が ASPDEV パラメーター上に指定されていなければなりません。 /asp_name/QSYS.LIB/object.type のパス名命名規則は、SAV および RST 上ではサポートされていません。

27 ページの『iSeries メニューおよび表示画面を使用した操作の実行』で説明したユーザー表示画面にも、同じ制限があります。

独立 ASP QSYS.LIB ファイル・システムでの統合ファイル・システム API の使用

54 ページの『API を使用した操作の実行』にリストしてある C 言語関数は、次のものを除き、独立 ASP QSYS.LIB ファイル・システムで使用することができます。

- ファイル操作は、プログラム記述物理ファイルとソース物理ファイルに対してのみ行うことができます。
- symlink() 関数は、シンボリック・リンクをサポートする別のファイル・システムから独立 ASP QSYS.LIB のオブジェクトにリンクする場合のみ使用できます。
- QjoStartJournal() および QjoEndJournal() API は、データベース物理ファイル上では使用できません。

文書ライブラリー・サービス・ファイル・システム (QDLS)

QDLS ファイル・システムは、フォルダー構造をサポートします。文書とフォルダーへのアクセスを提供します。

さらに、

- iSeries サーバーのフォルダーおよび文書ライブラリー・オブジェクト (DLO) をサポートします。
- ストリーム・ファイルに保管されるデータをサポートします。

QDLS についての詳細は、 『統合ファイル・システム・インターフェースを介した QDLS の使用』を参照してください。

統合ファイル・システム・インターフェースを介した QDLS の使用

QDLS ファイル・システムは、OS/400 ファイル・サーバーまたは統合ファイル・システムのコマンド、ユーザー表示画面、および C 言語 API を使用して、統合ファイル・システム・インターフェースによりアクセスすることができます。これらの統合ファイル・システム・インターフェースを使用する際には、次に挙げる考慮事項および制限事項に注意してください。

QDLS ファイル・システムでの統合ファイル・システムおよび HFS

QDLS ファイル・システムのオブジェクトは、文書ライブラリー・オブジェクト (DLO) CL コマンドだけではなく、階層ファイル・システム (HFS) が提供する統合ファイル・システム・インターフェースまたは API を介して、操作することができます。統合ファイル・システム が ILE (統合言語環境) プログラム・モデルに基づいているのに対し、 HFS は従来の iSeries サーバー・プログラム・モデルに基づいています。

HFS API を使用すると、統合ファイル・システムでサポートされていない操作をいくつか行うことができます。特に、HFS API では、ディレクトリー拡張属性 (*ディレクトリー項目属性* とも呼ばれる) のアクセ

スおよび変更を行うことができます。 HFS API を使用するための命名規則は、統合ファイル・システム・インターフェースを使用する API の命名規則とは異なるので、注意してください。

HFS についての詳細は、iSeries Information Center にある 『Hierarchical File System APIs』を参照してください。

QDLS ファイル・システムでのユーザー登録

QDLS のオブジェクトを処理するユーザーは、システム配布ディレクトリーに登録されていなければなりません。

QDLS ファイル・システムでの大文字小文字の区別

QDLS では、オブジェクト名に使用される英語のアルファベットの小文字 (**a** から **z** まで) を、大文字に変換します。したがって、英語のアルファベットだけを使用しているオブジェクト名の検索では、大文字と小文字は区別されません。

他のすべての文字については、QDLS では大文字と小文字が区別されます。

詳細については、iSeries Information Center にある 『フォルダー名および文書名』のトピックを参照してください。

QDLS ファイル・システムでのパス名

- パス名の各構成要素は、次に示すように名前のみ、

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

  あるいは、次に示すように名前とエクステンション (DOS のファイル拡張子と同様) で構成されます。

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

- 各構成要素の名前は 8 文字まで、エクステンション (ある場合) は 3 文字までの長さにすることができます。パス名の最大長は、82 文字です (パス名が /QDLS で始まる絶対パス名の場合)。
- QDLS 内のディレクトリー階層は、32 レベルまでの深さにすることができます。最初の 2 レベルに / および QDLS が含まれていれば、ディレクトリー階層は、34 レベルまでの深さにすることができます。
- 名前に使用される文字は、データ域 Q0DEC500 が QUSRSYS ライブラリーに作成されていない限り、名前の保管時にジョブのコード・ページに変換されます。データ域が存在する場合、名前に使用されている文字は、名前が保管されるときにコード・ページ 500 に変換されます。この機能は、前のリリースの QDLS ファイル・システムの動作との互換性を提供します。適切なコード・ページに変換できない名前は、拒否されます。

  コード・ページについての詳細は、iSeries Information Center にある 『グローバリゼーション』のトピックを参照してください。

QDLS ファイル・システムでのリンク

QDLS ファイル・システムでは、シンボリック・リンクを作成、保管することはできません。

統合ファイル・システムは、フォルダーと文書ライブラリー・オブジェクトの関係を、フォルダーとフォルダー内の各オブジェクトとの間に、それぞれリンクが設定されているのと同等として扱います。したがって、シンボリック・リンクをサポートするファイル・システムから、 QDLS ファイル・システムのオブジェクトにリンクすることが可能です。

リンクについては、 19 ページの『リンク』を参照してください。

QDLS ファイル・システムでの統合ファイル・システム・コマンドおよび表示画面の使用

28 ページの『CL コマンドを使用した操作の実行』にリストしてあるコマンドは、次の場合を除いて、QDLS ファイル・システムで使用することができます。

- ADDLNK コマンドは、シンボリック・リンクをサポートする別のファイル・システムから、 QDLS のオブジェクトにリンクする場合のみ使用できます。
- CHKIN および CHKOUT コマンドは、ファイルについてはサポートされていますが、ディレクトリーについてはサポートされていません。
- APYJRNCHG、ENDJRN、SNDJRNE、および STRJRN コマンドはサポートされていません。

27 ページの『iSeries メニューおよび表示画面を使用した操作の実行』で説明したユーザー表示画面にも、同じ制限があります。

QDLS ファイル・システムでの統合ファイル・システム API の使用

54 ページの『API を使用した操作の実行』 にリストしてある C 言語関数は、次のものを除き、 QDLS ファイル・システムで使用することができます。

- symlink() 関数は、シンボリック・リンクをサポートする別のファイル・システムから、 QDLS のオブジェクトにリンクする場合のみ使用できます。
- 次の関数は、サポートされていません。

  givedescriptor()

  ioctl()

  link()

  QjoEndJournal()

  QjoRetrieveJournalInformation()

  QJORJIDI()

  QJOSJRNE()

  QjoStartJournal()

  Qp0lGetPathFromFileID()

  readlink()

  takedescriptor()

光ファイル・システム (QOPT)

QOPT ファイル・システムにより、光メディアに保管されたストリーム・データにアクセスできます。

さらに、

- DOS や OS/2 などの PC オペレーティング・システムと同様の階層ディレクトリー構造を提供します。
- ストリーム・ファイル入出力用に最適化されています。
- ストリーム・ファイルに保管されるデータをサポートします。

QOPT についての詳細は、 『統合ファイル・システム・インターフェースを介した QOPT の使用』を参照してください。

統合ファイル・システム・インターフェースを介した QOPT の使用

QOPT ファイル・システムは、OS/400 ファイル・サーバーまたは統合ファイル・システムのコマンド、ユーザー表示画面、および API を使用して、統合ファイル・システム・インターフェースによりアクセスすることができます。これらの統合ファイル・システム・インターフェースを使用する際には、次に挙げる考慮事項および制限事項に注意してください。

詳細については、オプティカル・サポート を参照してください。

QOPT ファイル・システムでの統合ファイル・システムおよび HFS

QOPT ファイル・システムのオブジェクトは、統合ファイル・システム・インターフェースまたは階層ファイル・システム (HFS) が提供する API を介して操作することができます。統合ファイル・システムが ILE (統合言語環境) プログラム・モデルに基づいているのに対し、 HFS は従来の iSeries サーバー・プログラム・モデルに基づいています。

HFS API を使用すると、統合ファイル・システムでサポートされていない操作をいくつか行うことができます。特に、ディレクトリー拡張属性 (*ディレクトリー項目属性* とも呼ばれる) のアクセスや変更をしたり、保留されている光ファイルを処理したりするのに使用できます。 HFS API を使用するための命名規則は、統合ファイル・システム・インターフェースを使用する API の命名規則とは異なるので、注意してください。

HFS API についての詳細は、iSeries Information Center にある 『Hierarchical File System APIs』のトピック、または オプティカル・サポート を参照してください。

QOPT ファイル・システムでの大文字小文字の区別

光メディアのフォーマットに応じて、QOPT 内にファイルまたはディレクトリーを作成する際に、大文字小文字が保たれる場合と保たれない場合があります。しかし、光メディアのフォーマットに関係なく、ファイルおよびディレクトリーの検索では大文字小文字を区別しません。

QOPT ファイル・システムでのパス名

- パス名は斜線(/)で開始しなければなりません。パスは、ファイル・システム名、ボリューム名、ディレクトリー名とサブディレクトリー名、およびファイル名で構成されます。次はその一例です。

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

- ファイル・システム名の QOPT は必須です。
- ボリュームおよびパス名の長さは、光メディアのフォーマットに応じて異なります。
- パス名の中に単に /QOPT を指定するか、またはパス名の中に 1 つ以上のディレクトリーまたはサブディレクトリーを含めることができます。ディレクトリー名およびファイル名には、X'00'～X'3F' および X'FF' を除く、任意の文字を使用できます。光メディアのフォーマットによっては、その他の制限事項が適用されることがあります。
- ファイル名は、パス名の最後の要素です。ファイル名の長さは、パス内のディレクトリー名の長さによって制限されます。

QOPT ファイル・システムのパス名規則についての詳細は、 オプティカル・サポート の『パス名の規則』を参照してください。

QOPT ファイル・システムでのリンク

QOPT ファイル・システムでは、 1 つのオブジェクトにつき 1 つのリンクのみがサポートされています。 QOPT では、シンボリック・リンクを作成、保管することはできません。ただし、QOPT のファイルには、「ルート (/)」または QOpenSys ファイル・システムから、シンボリック・リンクを使用してアクセスすることができます。

リンクについては、 19 ページの『リンク』を参照してください。

QOPT ファイル・システムでの統合ファイル・システム・コマンドおよび表示画面の使用

28 ページの『CL コマンドを使用した操作の実行』にリストしているほとんどのコマンドは、QOPT ファイル・システムで使用することができます。ただし、QOPT ファイル・システムではいくつかの例外があります。マルチスレッドが可能なプロセスでのこれらの CL コマンドの使用は安全でない場合があることに注意してください。光メディアのフォーマットに応じて、いくつかの制限事項が適用される場合があります。 27 ページの『iSeries メニューおよび表示画面を使用した操作の実行』で説明したユーザー表示画面にも、同じ制限があります。

以下の統合ファイル・システム・コマンドは、QOPT ファイル・システムではサポートされていません。

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

QOPT ファイル・システムでの統合ファイル・システム API の使用

「ルート (/)」ファイル・システムでは、次の場合を除いて、 54 ページの『API を使用した操作の実行』にリストしてあるすべての C 言語 API を使用することができます。

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

NetWare ファイル・システム (QNetWare)

QNetWare ファイル・システムは、 Novell NetWare 4.10 または 4.11 を実行するローカルまたはリモートの iSeries 統合 xSeries サーバー上のデータ、または Novell NetWare 3.12、4.10、4.11、または 5.0 を実行するスタンドアロン PC サーバーへのアクセスを提供します。

さらに、

- NetWare ディレクトリー・サービス (NDS) オブジェクトへのアクセスを提供します。
- ストリーム・ファイルに保管されるデータをサポートします。

- NetWare ファイル・システムのローカル・ネーム・スペースへの動的マウントを提供します。

注: QNetWare ファイル・システムは、システムに NetWare Enhanced Integration for iSeries 400、 BOSS オプション 25 がシステム上にインストールされている場合にのみ使用可能です。インストール後の最初の IPL 時に、/QNetWare ディレクトリーとそのサブディレクトリーが、統合ファイル・システム・ディレクトリー構造の一部となります。

QNetWare ファイル・システムについての詳細は、以下のトピックを参照してください。

- NetWare ファイル・システムのマウント
- QNetWare ディレクトリー構造
- 統合ファイル・システム・インターフェースを介した QNetWare の使用

NetWare ファイル・システムのマウント

Novell NetWare サーバー上にある NetWare ファイル・システムは、「ルート (/)」、QOpenSys、および他のファイル・システム上にマウントでき、アクセスをより容易にし、 /QNetWare ディレクトリー下よりも良い状態で実行できるようにします。さらに NetWare ファイル・システムのマウントは、読み取り書き込み可能なファイル・システムを読み取り専用としてマウントするなど、マウント・ファイル・システムの追加 (ADDMFS) コマンドのオプションを利用するためにも使用できます。

NetWare ファイル・システムは、NDS パスを使用して、または NetWare パスを SERVER/VOLUME:directory/directory の形式で指定することによって、マウントできます。たとえば、サーバー Dreyfuss 上のボリューム Nest にあるディレクトリー doorway をマウントするには、次の構文を使用します。

```
DREYFUSS/NEST:doorway
```

このパス構文は、NetWare MAP コマンド構文に非常によく似ています。 NDS パスを使用して、NetWare ボリュームへのパスを指定できますが、それら自体をマウントすることはできません。

QNetWare ディレクトリー構造

/QNetWare ディレクトリー構造は、次のように複数の区別されたファイル・システムを表します。

- この構造は次の形式で、ネットワークの Novell NetWare サーバーとボリュームを示します。

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

  拡張子 .SVR は、Novell NetWare サーバーを表すために使用されます。
- サーバー下のボリュームが統合ファイル・システムのメニュー、コマンド、または API のいずれかでアクセスされる場合、 NetWare ボリュームのルート・ディレクトリーは、 /QNetWare の下の VOLUME ディレクトリーに自動的にマウントされます。
- QNetWare は次の形式で、ネットワーク上の NDS ツリーを表します。

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

  拡張子 .TRE、.C、.O、.OU、および .CN は、それぞれ NDS ツリー、国、組織、組織内単位、および共通名を表します。 Novell NetWare ボリュームが、ボリューム・オブジェクトまたはボリューム・オブジェクトへの別名を介した NDS パスを通してアクセスされる場合、そのルート・ディレクトリーも自動的に NDS オブジェクトにマウントされます。

統合ファイル・システム・インターフェースを介した QNetWare の使用

QNetWare ファイル・システムは、 OS/400 ファイル・サーバーまたは統合ファイル・システムのコマンド、ユーザー表示画面、および API を使用して、統合ファイル・システム・インターフェースによりアクセスすることができます。次のような考慮事項、制限事項、および依存性があります。

QNetWare ファイル・システムでの権限および所有権

QNetWare のファイルとディレクトリーは、Novell NetWare サーバーにより保管され、管理されます。コマンドおよび API を使用して、所有者またはユーザーの権限を検索または設定する場合、 QNetWare は NetWare ユーザーをユーザーの名前に基づいて iSeries サーバー・ユーザーにマップさせます。 NetWare 名が 10 文字を超え、対応する iSeries サーバー・ユーザーが存在しない場合、権限はマップされません。マップできない所有者は、自動的にユーザー・プロファイル QDFTOWN にマップされます。ユーザーの権限は、WRKAUT および CHGAUT コマンドを使用して表示および変更できます。権限がサーバーとの間で転送されると、それらの権限は iSeries サーバー権限にマップされます。

QNetWare ファイル・システムでの監査

Novell NetWare がファイルおよびディレクトリーの監査をサポートしても、 QNetWare ファイル・システムはこれらのオブジェクトの監査値を変更することはできません。したがって、CHGAUD コマンドはサポートされません。

QNetWare ファイル・システムでのファイルおよびディレクトリー

QNetWare ファイル・システムは、コマンドまたは API に入力されたファイルまたはディレクトリーの大文字小文字の区別を保存しません。すべての名前は、NetWare サーバーに送信されるときに大文字に設定されます。 Novell NetWare も、DOS、 OS/2、Apple Macintosh、および NFS などの複数のプラットフォームのネーム・スペースをサポートします。 QNetWare ファイル・システムは、DOS ネーム・スペースをサポートするだけです。 DOS ネーム・スペースは、すべての Novell NetWare ボリュームで必要であるため、すべてのファイルおよびディレクトリーは QNetWare ファイル・システムに表示されます。

QNetWare ファイル・システムでの NDS オブジェクト

QNetWare ファイル・システムは、NDS 名の表示を大文字でも小文字でもサポートします。

QNetWare ファイル・システムでのリンク

QNetWare ファイル・システムでは、 1 つのオブジェクトにつき 1 つのリンクのみがサポートされています。 QNetWare では、シンボリック・リンクを作成、保管することはできません。しかし、シンボリック・リンクは、 QNetWare のファイルまたはディレクトリーを指す「ルート (/)」または QOpenSys ディレクトリーで作成できます。

QNetWare ファイル・システムでの統合ファイル・システム・コマンドおよび表示画面の使用

28 ページの『CL コマンドを使用した操作の実行』にリストしてあるコマンドは、次の場合を除いて、QNetWare ファイル・システムで使用することができます。

ADDLINK

| APYJRNCHG

CHGAUD

CHGPGP

CHKIN

CHKOUT

| ENDJRN

| SNDJRN
| STRJRN
WRKOBJOWN
WRKOBJPGP

上記のコマンドに加え、次のコマンドは NDS オブジェクト、サーバー、またはボリュームに対して使用することができません。

CHGOWN
CPYFRMSTMF
CPYTOSTMF
CRTDIR

QNetWare ファイル・システムでの統合ファイル・システム API の使用

54 ページの『API を使用した操作の実行』にリストしてある C 言語関数は、次の API を除き、QNetWare ファイル・システムで使用することができます。

givedescriptor()
link()
| QjoEndJournal()
| QjoRetrieveJournalInformation()
| QJORJIDI()
| QJOSJRNE()
| QjoStartJournal()
readlink()
symlink()
takedescriptor()

上記の API に加え、次の API は NDS オブジェクト、サーバー、またはボリュームに対して使用することができません。

chmod()
chown()
create()
fchmod()
fchown()
fcntl()
ftruncate()
lseek()
mkdir()
read()
readv()
unmask()
write()
writev()

Windows NT Server ファイル・システム (QNTC)

QNTC ファイル・システムは、 Windows NT 4.0 サーバーまたはそれ以降のスタンドアロン・サーバーを実行する、ローカルまたはリモートの iSeries 統合 xSeries サーバーに保管されている、データおよびオブジェクトへのアクセスを提供します。このファイル・システムを使用すると、 Windows NT クライアントと同じデータを iSeries サーバー・アプリケーションで使用できます。それは、ストリーム・ファイル内にデータを保管します。

QNTC ファイル・システムは基本 OS/400 オペレーティング・システムの一部です。 QNTC ファイル・システムを使用するには、 TCP/IP Connectivity Utilities for iSeries 400 (部品番号: 5769-TC1) をインストールしておく必要があります。 iSeries 400 Integration with Windows NT Server (オペレーティング・システムのオプション 29) は、 /QNTC にアクセスするためにインストールされている必要はありません。

QNTC についての詳細は、『統合ファイル・システム・インターフェースを介した QNTC の使用』を参照してください。

統合ファイル・システム・インターフェースを介した QNTC の使用

OS/400 ファイル・サーバーまたは統合ファイル・システムのコマンド、ユーザー表示画面、および API を使用して、統合ファイル・システム・インターフェースにより QNTC ファイル・システムにアクセスすることができます。次のような考慮事項および制限事項があります。

QNTC ファイル・システムでの権限および所有権

QNTC ファイル・システムは、ファイルまたはディレクトリーの所有権の概念をサポートしていません。コマンドまたは API を使用して、 QNTC に保管されているファイルの所有権を変更しようとしても、失敗となります。 QDFTOWN というシステム・ユーザー・プロファイルが、 QNTC のすべてのファイルおよびディレクトリーを所有しています。

NT サーバー・ファイルおよびディレクトリーへの権限は、 Windows NT サーバーから管理されます。 QNTC は WRKAUT および CHGAUT コマンドをサポートしません。

QNTC ファイル・システムでの大文字小文字の区別

QNTC ファイル・システムは、オブジェクト名の大文字と小文字を、入力された状態で保持しますが、名前の大文字と小文字の区別はしません。オブジェクト名の検索は、大文字と小文字のどちらで行っても同じです。

QNTC ファイル・システムでのパス名

- パス名は斜線で始まり、255 文字までの長さにすることができます。
- パス名は大文字小文字を区別します。
- パスは、ファイル・システム名、Windows NT サーバー名、共用名、ディレクトリー名とサブディレクトリー名、およびオブジェクト名で構成されます。パス名の形式は、次のとおりです。

```
    /QNTC/Servername/Sharename/Directory/ . . . /Object
(QNTC is a required part of the path name.)
```

- サーバー名は最高 15 文字までです。パスの一部でなければなりません。
- 共用名は最高 12 文字までです。
- 共用名の後のパス名の各構成要素は、255 文字までの長さにすることができます。
- QNTC では、通常は 130 レベルの階層が使用できます。パス名のすべての構成要素が階層レベルとして含まれていれば、ディレクトリー階層は、132 レベルまでの深さにすることができます。
- 名前は Unicode CCSID で保管されます。

- ローカル・サブネットの各機能 Windows NT サーバーは、 /QNTC の下にディレクトリーとして自動的に表示されます。ディレクトリーの作成 (MKDIR) コマンド (29 ページの表 2を参照) または mkdir() API (54 ページの『API を使用した操作の実行』を参照) を使用して、ローカル・サブネットの外側に Windows NT サーバーを追加します。

QNTC ファイル・システムでのリンク

QNTC ファイル・システムでは、 1 つのオブジェクトにつき 1 つのリンクのみがサポートされています。 QNTC でシンボリック・リンクを作成したり保管したりすることはできません。「ルート (/)」または QOpenSys ファイル・システムからシンボリック・リンクを使用して、 QNTC のデータにアクセスすることができます。

リンクについては、 19 ページの『リンク』を参照してください。

QNTC ファイル・システムでの統合ファイル・システム・コマンドおよび表示画面の使用

| 28 ページの『CL コマンドを使用した操作の実行』にリストしてあるコマンドは、次の場合を除いて、
| QNTC ファイル・システムで使用することができます。

ADDLNK
| APYJRNCHG
CHGOWN
CHGAUT
CHGPGP
CHKIN
CHKOUT
DSPAUT
| ENDJRN
RST
SAV
| SNDJRNE
| STRJRN
WRKAUT
WRKOBJOWN
WRKOBJPGP

27 ページの『iSeries メニューおよび表示画面を使用した操作の実行』で説明したユーザー表示画面にも、同じ制限があります。

QNTC ファイル・システムでの MKDIR コマンドの使用

ディレクトリーの作成 (MKDIR) コマンドは、サーバー・ディレクトリーを /QNTC ディレクトリーに追加するのに使用します。ローカル・サブネット中のすべての機能 Windows NT サーバーは自動的に作成されます。ローカル・サブネット外のこれらの Windows NT サーバーは、 MKDIR コマンドまたは mkdir() API を使用して追加しなければなりません。次はその一例です。

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

これで、NTSRV1 サーバーを QNTC ファイル・システム・ディレクトリー構造に追加し、そのサーバー上でのファイルとディレクトリーのアクセスを可能にします。

また、TCP/IP アドレスを使用してディレクトリー構造に新しいサーバーを追加することもできます。次はその一例です。

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

これで、サーバーを QNTC ファイル・システム・ディレクトリー構造に追加します。

注: mkdir() API または MKDIR コマンドを使用して、ディレクトリー構造にディレクトリーを追加する場合、これらのディレクトリーは IPL を経ると見えなくなります。 MKDIR コマンドまたは mkdir() API を、すべてのシステム IPL 後に再発行する必要があります。

QNTC ファイル・システムでの統合ファイル・システム API の使用

54 ページの『API を使用した操作の実行』にリストしてある C 言語関数は、次のものを除き、 QNTC ファイル・システムで使用することができます。

- chmod()、fchmod()、utime()、および umask() 関数は、 QNTC のオブジェクトには効力がありませんが、使用してもエラーは起こりません。
- QNTC ファイル・システムは、次の関数をサポートしていません。

  chown()

  fchown()

  givedescriptor()

  link()

  QjoEndJournal()

  QjoRetrieveJournalInformation()

  QJORJIDI()

  QJOSJRNE()

  QjoStartJournal()

  Qp0lGetPathFromFileID()

  readlink()

  symlink()

  takedescriptor()

OS/400 ファイル・サーバー・ファイル・システム (QFileSvr.400)

OS/400 ファイル・サーバー ファイル・システムにより、リモートの iSeries サーバーに常駐する他のファイル・システムに透過的なアクセスができます。階層ディレクトリー構造を介してアクセスされます。

QFileSvr.400 ファイル・システムは、ユーザーの代わりにクライアントとしてファイル要求を実行するものです。 QFileSvr.400 はターゲット・システムの OS/400 ファイル・サーバーと対話して、実際のファイル操作を実行します。

QFileSvr.400 についての詳細は、『統合ファイル・システム・インターフェースを介した QFileSvr.400 の使用』を参照してください。

統合ファイル・システム・インターフェースを介した QFileSvr.400 の使用

QFileSvr.400 ファイル・システムは、OS/400 ファイル・サーバーまたは統合ファイル・システムのコマンド、ユーザー表示画面、および API を使用して、統合ファイル・システム・インターフェースによりアクセスすることができます。これらの統合ファイル・システム・インターフェースを使用する際には、次に挙げる考慮事項および制限事項に注意してください。

注: QFileSvr.400 ファイル・システムの特性は、ターゲット・サーバー上でアクセスしようとしているファイル・システムの特性によって決まります。

OS/400 ファイル・サーバー ファイル・システムでの大文字小文字の区別

英字の大・小文字については、第 1 レベル・ディレクトリーの場合、ターゲット・システムの実際の「ルート (/)」 ディレクトリーを表すので、 QFileSvr.400 ファイル・システムではオブジェクト名の入力に使用されたものがそのまま使用されます。ただし、QFileSvr.400 が名前を検索するときには、大文字と小文字を区別しません。

その他のすべてのディレクトリーの場合、大文字と小文字の区別はアクセスしようとしている特定のファイル・システムによって異なります。 QFileSvr.400 では、ファイル要求が OS/400 ファイル・サーバーに送られたときに入力されたのと同じオブジェクト名の大・小文字を使用します。

OS/400 ファイル・サーバー ファイル・システムでのパス名

- パス名の形式は、次のとおりです。

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

  第 1 レベル・ディレクトリー (上記の例では RemoteLocationName) は、以下の両方を表します。
  - 通信の接続を確立するために使用されるターゲット・サーバーの名前。ターゲット・サーバー名は、次のどちらでもかまいません。
    - TCP/IP ホストの名前 (たとえば、beowulf.newyork.corp.com)
    - SNA LU 6.2 の名前 (たとえば、appn.newyork)
  - ターゲット・サーバーの「ルート (/)」ディレクトリー

  このため、統合ファイル・システム・インターフェースを使用して、第 1 レベル・ディレクトリーが作成されるとき、指定されている属性はすべて無視されます。

  注: 第 1 レベル・ディレクトリーは、IPL 後は保持されません。つまり、IPL を実行した場合には、そのたびに第 1 レベル・ディレクトリーを作成し直さなければなりません。
- パス名の各構成要素は、255 文字までの長さにすることができます。全パス名は、16 メガバイトまでの長さにすることができます。

  注: オブジェクトが常駐するファイル・システムによって、コンポーネントの長さとパス名の長さが、QFileSvr.400 で認められる最大長より小さくなるよう、制限されることがあります。
- ディレクトリー階層の深さについては、プログラム、システム、およびアクセス中のファイル・システムに制限される他は、制限されません。
- 名前に使用されている文字は、名前が保管されるときに UCS2 のレベル 1 形式に変換されます (24 ページの『名前の継続性』を参照)。

OS/400 ファイル・サーバー ファイル・システムでの通信

- ターゲット・サーバー上のファイル・サーバーとの TCP 接続は、ターゲット・サーバーの QSERVER サブシステムが活動状態のときだけ確立できます。

- SNA LU 6.2 接続は、使用中でないローカル制御セッションがある場合 (LU 6.2 接続で使用するために特別に確立されたセッションの場合など) のみに試行されます。 LU 6.2 接続の確立時には、QFileSvr.400 ファイル・システムは BLANK モードを使用します。ターゲット・システムでは、QPWFSERV というジョブが QSERVER サブシステムに対して実行依頼されます。このジョブのユーザー・プロファイルは、 BLANK モードの通信項目によって定義されます。 LU6.2 通信についての詳細は、APPC プログラミング  を参照してください。
- TCP を通信プロトコルとして使用するファイル・サーバー要求は、その要求を発行しているジョブのコンテキスト内で実行されます。また、SNA を通信プロトコルとして使用するファイル・サーバー要求は、 OS/400 システム・ジョブの Q400FILSVR によって実行されます。
- ターゲット・サーバーとの接続がまだ確立されていない場合、 QFileSvr.400 ファイル・システムは、第 1 レベル・ディレクトリーを TCP/IP ホスト名として処理します。 QFileSvr.400 ファイル・システムは以下のステップをすべて実行し、ターゲット・サーバーとの通信を確立します。
  1. リモート・ロケーション名を IP アドレスへ解決します。
  2. 変換された IP アドレスを使用して、ホスト・サーバーの、ウェルノウン・ポート 449 上のサーバー・マップ・プログラムに接続します。次に、そのサーバー・マッパーにサービス名 『as-file』 を照会します。照会の結果、次のどちらかになります。
     - 『as-file』 がターゲット・サーバーのサービス・テーブルにある場合、サーバー・マップ・プログラムは OS/400 ファイル・サーバー・デーモンが listen しているポートを戻します。
     - サーバー・マッパーがターゲット・サーバーで活動状態になっていない場合は、 『as-file』 のデフォルト・ポート番号 (8473) が使用されます。

     次に、QFileSvr.400 ファイル・システムは、ターゲット・サーバー上の OS/400 ファイル・サーバー・デーモンと TCP 接続を確立しようとします。接続が確立されると、QFileSvr.400 は、要求と応答をファイル・サーバーと交換します。 QSERVER サブシステム内では、QPWFSERVSO 事前開始要求が接続を制御します。個々の事前開始ジョブは、それぞれのユーザー・プロファイルのもとで実行されます。
  3. リモート・ロケーション名が IP アドレスに変換されない場合、第 1 レベル・ディレクトリーが SNA LU 6.2 名と想定されます。それから、OS/400 ファイル・サーバーとの APPC 接続の確立が試行されます。
- QFileSvr.400 ファイル・システムは、定期的 (2 時間ごと) に、使用中でない接続がある (たとえば、その接続と関連付けられたオープン・ファイルがない) か、そしてそれらの接続が 2 時間の間に、何も活動しなかったかを判別する検査を行います。そのような接続が検出された場合は、その接続は終了されます。
- QFileSvr.400 ファイル・システムはループを検出できません。次のパス名は、ループの例です。

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

  上の例で、Remote1 はローカル・システムを表します。ループを含むパス名が指定されると、QFileSvr.400 ファイル・システムは短時間の経過後にエラーを戻します。このエラーでは、タイムアウトが発生したことが示されます。

  QFileSvr.400 ファイル・システムは、 SNA を介して通信するときに、既存の空きセッションを使用します。これは、モードを開始し、QFileSvr.400 にセッションを確立して、リモート通信システムに正常に接続するのに必要なことです。

OS/400 ファイル・サーバー ファイル・システムでのセキュリティーおよびオブジェクト権限

両方のシステムに Kerberos が構成されており、ユーザーが Kerberos に認証した場合、 Kerberos を使って、ターゲット iSeries サーバー上に存在するファイル・システムを認証することができます。 Kerberos による認証が失敗する場合、ユーザー ID およびパスワードを使って、アクセスを検査することができます。

注: ターゲット・サーバーがアクセスを検査した後で、発券許可証またはサーバー・チケットの有効期限が切れた場合、ターゲット・サーバーへの接続が終了するまでその有効期限は反映されません。 Kerberos についての詳細は、iSeries Information Center にある 『ネットワーク認証サービス』のトピックを参照してください。

- 認証するために Kerberos が使用されない場合、ターゲットの iSeries サーバーに常駐するファイル・システムにアクセスするためには、ローカル・サーバーのユーザー ID およびパスワードに一致する、ターゲット・サーバーのユーザー ID およびパスワードがなければなりません。

  注: ターゲット・サーバーがアクセスを検査したあとで、ローカル・サーバーまたはターゲット・サーバーのパスワードが変更された場合、ターゲット・サーバーとの接続が終了するまでその変更は反映されません。ただし、ローカル・サーバーのユーザー・プロファイルが削除され、同じユーザー ID で別のユーザー・プロファイルが作成された場合には遅延はありません。この場合、QFileSvr.400 ファイル・システムは、ターゲット・サーバーへのアクセス権があるかどうかを検査します。

- オブジェクト権限は、ターゲット・サーバーに存在するユーザー・プロファイルに基づいています。つまり、ターゲット・サーバーのファイル・システムにあるオブジェクトにアクセスできるのは、ターゲット・サーバーのユーザー・プロファイルに、そのオブジェクトに対する適切な権限がある場合に限られます。

OS/400 ファイル・サーバー ファイル・システムでのリンク

QFileSvr.400 ファイル・システムでは、 1 つのオブジェクトにつき 1 つのリンクのみがサポートされています。 QFileSvr.400 では、シンボリック・リンクを作成、保管することはできません。ただし、QFileSvr.400 のファイルには、「ルート (/)」、QOpenSys、またはユーザー定義のファイル・システムから、シンボリック・リンクを使用してアクセスすることができます。

リンクについては、 19 ページの『リンク』を参照してください。

OS/400 ファイル・サーバー ファイル・システムでの統合ファイル・システム・コマンドおよび表示画面の使用

28 ページの『CL コマンドを使用した操作の実行』にリストしてあるコマンドは、次の場合を除いて、QFileSvr.400 ファイル・システムで使用することができます。

ADDLNK
APYJRNCHG
CHGAUT
CHGOWN
DSPAUT
ENDJRN
RST
SAV
SNDJRNE

| STRJRN
WRKOBJOWN
WRKOBJPGP

27 ページの『iSeries メニューおよび表示画面を使用した操作の実行』で説明したユーザー表示画面にも、同じ制限があります。

OS/400 ファイル・サーバー ファイル・システムでの統合ファイル・システム API の使用

54 ページの『API を使用した操作の実行』 にリストしてある C 言語関数は、次のものを除き、QFileSvr.400 ファイル・システムで使用することができます。

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

ネットワーク・ファイル・システム (NFS)

NFS ファイル・システムは、ユーザーに、リモート NFS サーバーに保管されるデータとオブジェクトへのアクセスを提供します。 NFS サーバーからネットワーク・ファイル・システムをエクスポートしてから、 NFS クライアントに動的にマウントすることができます。

さらに、ネットワーク・ファイル・システムを介して、ローカルにマウントされたファイルには、リモート・サーバー上でマウントされたディレクトリー、またはファイル・システムの機能、特定、制限、および依存性があります。マウント・ファイル・システム上の操作はローカルには実行されません。要求はサーバーへの接続を介して流れ、サーバー上のファイル・システムのタイプの要件、および制限に従う必要があります。

NFS についての詳細は、『統合ファイル・システム・インターフェースを介した NFS ファイル・システムの使用』を参照してください。

統合ファイル・システム・インターフェースを介した NFS ファイル・システムの使用

ネットワーク・ファイル・システムは、統合ファイル・システム・インターフェースを介してアクセス可能であり、次の考慮事項および制限があります。

ネットワーク・ファイル・システムの特性

NFS を介してマウントされるファイル・システムの特性は、サーバーからマウントされた、ファイル・システムのタイプに依存しています。ローカル・ディレクトリーまたはファイル・システムにあるように見えるもので実行される要求は、実際には NFS 接続を介してサーバー上で操作しているということを意識しておくのは重要です。

このクライアント / サーバー関係は複雑です。たとえば、クライアントの「ルート (/)」ディレクトリーの分岐の最上部で、サーバーから QDLS ファイル・システムをマウントした場合を考慮してみます。マウントされたファイル・システムがローカル・ディレクトリーの拡張子に見えても、実際には QDLS ファイル・システムとして機能し、実行します。

NFS を介してマウントされたファイル・システムのこの関係を意識しておくことは、要求をローカルに、かつサーバー接続を介して処理するために重要です。ローカル・レベルで正しく処理するコマンドが、サーバーからマウントされたディレクトリー上で作動することを意味していないためです。クライアントでマウントされた各ディレクトリーは、サーバー・ファイル・システムのプロパティーおよび特性を持っています。

ネットワーク・ファイル・システム内のサーバーおよびクライアントのバリエーション

ネットワーク・ファイル・システムの機能および特性に影響を及ぼす可能性のある、クライアント / サーバー接続には 3 つの主な可能性があります。

1. ユーザーが iSeries サーバーからファイル・システムをクライアントにマウントする。
2. ユーザーが UNIX サーバーからファイル・システムをクライアントにマウントする。
3. ユーザーが非 iSeries、非 UNIX サーバーからファイル・システムをクライアントにマウントする。

| 最初のシナリオでは、マウントされたファイル・システムはクライアント上で、 iSeries サーバーで動作するのと類似の方法で動作します。 ただし、ネットワーク・ファイル・システムと、サービスされているファイル・システムの両方の特性を考慮する必要があります。たとえば、サーバーからクライアントに QDLS ファイル・システムをマウントする場合、このファイル・システムには QDLS ファイル・システムの特性および制限があります。たとえば、QDLS ファイル・システムでは、パス名コンポーネントは 8 文字に 3 文字の拡張子を加えたものに制限されます。ただし、マウントされたファイル・システムには、NFS の特性および制限もあります。たとえば、NFS オブジェクトの監査値を変更するために CHGAUD コマンドを使用することはできません。

2 番目のシナリオでは、UNIX サーバーからマウントされたファイル・システムが、 iSeries QOpenSys ファイル・システムに非常に類似した動作をすることを意識することが重要です。 QOpenSys ファイル・システムの詳細については、 71 ページの『オープン・システム・ファイル・システム (QOpenSys)』を参照してください。

3 番目のシナリオでは、サーバーのオペレーティング・システムに関連したファイル・システムの資料を復習する必要があります。

ネットワーク・ファイル・システムでのリンク

一般的には、ネットワーク・ファイル・システムでは、 1 つのオブジェクトに複数のハード・リンクを設定することができます。シンボリック・リンクは、完全にサポートされています。シンボリック・リンクは、ネットワーク・ファイル・システムから、別のファイル・システムのオブジェクトへリンクするために、使用することができます。複数のハード・リンクおよびシンボリック・リンクの機能は、完全に NFS とマウントされているファイル・システムに依存しています。

リンクについては、 19 ページの『リンク』を参照してください。

ネットワーク・ファイル・システムでの統合ファイル・システム・コマンドの使用

| ネットワーク・ファイル・システムでは、次の場合を除いて、 28 ページの『CL コマンドを使用した操作の実行』 にリストしてあるすべてのコマンドと、 27 ページの『iSeries メニューおよび表示画面を使用した操作の実行』 に説明してある表示画面を使用することができます。

| • APYJRNCHG
| • CHGAUD
| • CHGATR
| • CHGAUT
| • CHGOWN
| • CHGPGP
| • CHKIN
| • CHKOUT
| • ENDJRN
| • SNDJRNE
| • STRJRN

ネットワーク・ファイル・システムおよび他の一般のマウント・ファイル・システムに特有の CL コマンドがいくつかあります。しかし、マルチスレッド可能プロセスでこれらのコマンドを使用するのは、安全性を考えると避けた方がよい場合があります。次の表で、これらのコマンドを説明します。ネットワーク・ファイル・システムに特に関連したコマンドおよび表示画面の完全な説明については、 OS/400 ネットワーク・ファイル・システム・サポート を参照してください。

表 7. ネットワーク・ファイル・システムの CL コマンド

コマンド	説明
ADDMFS	マウント・ファイル・システムの追加。ローカル・クライアント・ディレクトリー上に、エクスポートされたリモート・サーバー・ファイル・システムを配置する。
CHGNFSEXP	ネットワーク・ファイル・システム・エクスポートの変更。ネットワーク・ファイル・システム・クライアントへエクスポートされる、ファイル・システムのエクスポート・テーブルへ、ディレクトリー・ツリーを追加または除去する。
DSPMFSINF	マウント・ファイル・システム情報の表示。マウント・ファイル・システムについての情報を表示する。
ENDNFSSVR	ネットワーク・ファイル・システムの終了。サーバー上の 1 つまたはすべてのネットワーク・ファイル・システム・デーモンを終了する。
EXPORTFS	ファイル・システムのエクスポート。ネットワーク・ファイル・システム・クライアントへエクスポートされる、ファイル・システムのエクスポート・テーブルへ、ディレクトリー・ツリーを追加または除去する。
MOUNT	ファイル・システムのマウント。ローカル・クライアント・ディレクトリー上に、エクスポートされたリモート・サーバー・ファイル・システムを配置する。このコマンドは、ADDMFS コマンドの別名である。
RLSIFSLCK	統合ファイル・システム・ロックの解除。クライアントによって、またはオブジェクト上で保留された、ネットワーク・ファイル・システムのバイト範囲のロックをすべて解除する。
RMVMFS	マウント・ファイル・システムの除去。ローカル・クライアント・ネーム・スペースから、エクスポートされたリモート・サーバー・ファイル・システムを除去する。

表 7. ネットワーク・ファイル・システムの CL コマンド (続き)

コマンド	説明
STRNFSSVR	ネットワーク・ファイル・システムの開始。サーバー上の 1 つまたはすべてのネットワーク・ファイル・システム・デーモンを開始する。
UNMOUNT	ファイル・システムのアンマウント。ローカル・クライアント・ネーム・スペースから、エクスポートされたリモート・サーバー・ファイル・システムを除去する。このコマンドは、RMVMFS コマンドの別名である。

注: ネットワーク・ファイル・システムは、コマンドがこのシステムで使用される前にマウントする必要があります。

ネットワーク・ファイル・システムでの統合ファイル・システム API の使用

ネットワーク・ファイル・システムでは、次の場合を除いて、 54 ページの『API を使用した操作の実行』にリストしてあるすべての C 言語関数を使用することができます。

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

ネットワーク・ファイル・システムに特に関連した C 言語関数の完全な説明については、 OS/400 ネットワーク・ファイル・システム・サポート を参照してください。

注: ネットワーク・ファイル・システムは、API がこのシステムで使用される前にマウントする必要があります。

第 7 章 統合ファイル・システム・オブジェクトのジャーナル処理サポート

ジャーナル処理の主な目的は、オブジェクトの最後の保管以降にそのオブジェクトに加えられた変更を回復できるようにすることです。

この情報は、ジャーナル管理の概要を簡潔に示し、統合ファイル・システム・オブジェクトのジャーナル処理についての考慮事項、および統合ファイル・システム・オブジェクトのジャーナル処理サポートについての説明を示しています。

以下のトピックでは、統合ファイル・システム・オブジェクトのジャーナル処理サポートについて紹介しています。

- 『ジャーナル管理』
- 『ジャーナル処理するオブジェクト』
- 102 ページの『ジャーナル処理される統合ファイル・システム・オブジェクト』
- 103 ページの『ジャーナル処理される操作』
- 104 ページの『ジャーナル項目についての特別な考慮事項』

| 統合ファイル・システム オブジェクトのジャーナル処理についての詳細は、 iSeries Information Center の
| 『ジャーナル管理』のトピックを参照してください。

ジャーナル管理

ジャーナル管理の主な目的は、オブジェクトの最後の保管以降にそのオブジェクトに加えられた変更を回復できるようにすることです。さらに、以下の目的でジャーナル管理を使用することもできます。

- システム上のオブジェクトについて起きる活動の監査証跡
- ジャーナル処理できないオブジェクトついて起きる活動の記録
- 活動時保管メディアから復元するときの回復時間の短縮
- アプリケーション・プログラムのテストの援助

ジャーナルを使用して、ジャーナル管理によって保護したいオブジェクトを定義することができます。オブジェクトのジャーナル処理についての詳細な考慮事項については、『ジャーナル処理するオブジェクト』を参照してください。統合ファイル・システムでは、ストリーム・ファイル、ディレクトリー、およびシンボリック・リンクをジャーナル処理することができます。「ルート (/)」、QOpenSys、および UDFS ファイル・システム内のオブジェクトだけがサポートされています。

ジャーナル処理するオブジェクト

特定の統合ファイル・システム・オブジェクトに対してジャーナル処理を行うかどうかを決める際には、以下の質問を検討してください。

- そのオブジェクトはどれほど変更されますか。次の保管操作までの間に大量の変更が加えられるオブジェクトについては、ジャーナル処理の検討対象になります。
- そのオブジェクトに対する変更を再構築することはどれほど困難ですか。記録に残らない多くの変更が加えられますか。たとえば、電話による受注項目に使用されるオブジェクトは、注文用紙によって郵送で受け付けた受注に使用されるオブジェクトと比較して、再構築がより困難であると考えられます。

- そのオブジェクト内の情報はどれほど重要ですか。そのオブジェクトを最後の保管操作の状態に復元する必要がある場合、変更を再構築する際の遅延は、ビジネスにどのような影響を与えますか。
- そのオブジェクトは、サーバー上の他のオブジェクトとどのように関連していますか。特定のオブジェクト内のデータが頻繁に変更されなくても、そのオブジェクトのデータは、サーバー上のさらに動的な他のオブジェクトにとって重要である場合があります。たとえば、多くのオブジェクトは顧客マスター・ファイルに依存しています。受注情報を再構築する場合、顧客マスター・ファイルには、最後の保管以降に加えられた新規顧客や与信枠の変更を組み込む必要があります。

ジャーナル処理される統合ファイル・システム・オブジェクト

一部の統合ファイル・システム・オブジェクト・タイプは、OS/400 ジャーナル処理サポートを使用してジャーナル処理することができます。サポートされているオブジェクト・タイプは、 ストリーム・ファイル、ディレクトリー、およびシンボリック・リンクです。これらのオブジェクト・タイプのジャーナル処理をサポートしているファイル・システムは、「ルート (/)」、QOpenSys、および UDFS だけです。統合ファイル・システム・オブジェクト、従来のシステム・インターフェース (CL コマンドまたは API)、または iSeries ナビゲーターのいずれかを使用してジャーナル処理することができます。 iSeries ナビゲーターによって、ジャーナル処理の開始とジャーナル処理の終了、およびジャーナル処理情報の表示を実行することができます。

注: メモリー・マップ・ストリーム・ファイル、および仮想ドライブ・ストレージ・スペース用の iSeries 統合 xSeries サーバー (IXS) によって使用されるストリーム・ファイルをジャーナル処理することはできません。

次のリストは、統合ファイル・システムでのジャーナル処理サポートを要約しています。

- 汎用コマンドおよび API の両方を使用して、サポートされているオブジェクト・タイプに対するジャーナル操作を実行することができます。これらのインターフェースは通常、オブジェクトの識別情報を、パス名、ファイル ID、またはその両方の形式で受け入れます。
- ジャーナル処理の開始、ジャーナル処理の終了、およびジャーナル処理済み変更の適用を含むいくつかのジャーナル操作コマンドは、統合ファイル・システム・オブジェクトのサブツリー全体に対して実行することができます。オプションで、オブジェクト名に対するワイルド・カード・パターンを使用できる、組み込みリストおよび除外リストを使用することもできます。たとえば、ジャーナル処理の開始コマンドを使用して、ツリー `"/MyCompany"` 内で、パターン `"*.data"` に一致し、かつパターン `"A*.data"` と `"B*.data"` に一致するものは除外したすべてのオブジェクトを開始するように指定することができます。
- ディレクトリーに対するジャーナル処理サポートには、リンクの追加、リンクの除去、オブジェクトの作成、オブジェクトの名前変更、およびディレクトリー内でのオブジェクトの移動などのディレクトリー操作が含まれます。

  ジャーナル処理されるディレクトリーでは、ディレクトリーの現行のジャーナル状態をサブツリー内の新規オブジェクトに継承させる、属性の設定をサポートしています。ジャーナル処理されるディレクトリーについてこの属性がオンになっていると、 (ハード・リンクを追加するか、オブジェクトを名前変更または移動させることによって) そのディレクトリーに作成されているまたはリンクされているすべてのストリーム・ファイル、ディレクトリー、およびシンボリック・リンクについて、システムは自動的にジャーナル処理を開始します。

  注: オブジェクトのジャーナル処理を終了してから、現在同じディレクトリー内に存在するそのオブジェクトを名前変更する場合、ディレクトリーで「現在のジャーナル処理状態の継承」属性がオンであっても、そのオブジェクトのジャーナル処理は開始されません。

- オブジェクト名および完全パス名は、統合ファイル・システム・オブジェクトのいくつかのジャーナル項目内に含まれています。オブジェクト名およびパス名は、各国語サポート (NLS) に対応しています。
- システムが異常終了した場合、ジャーナル処理されている統合ファイル・システム・オブジェクトには、システムの初期プログラム・ロード (IPL) 回復が備えられています。
- write() および writev() API によってサポートされている最大書き込み制限は、2 GB -1 です。RCVSIZOPT (*MAXOPT2) が指定されている場合の最大ジャーナル項目サイズは、4,000,000,000 です。指定されていない場合、最大ジャーナル項目サイズは、15,761,440 バイトです。ストリーム・ファイルをジャーナル処理していて、15,761,440 バイトを超える書き込みがある場合、 *MAXOPT2 サポートを使用してエラーの発生を防ぐことができます。

統合ファイル・システム・オブジェクトのジャーナル処理についての詳細は、 iSeries Information Center にある『ジャーナル管理』を参照してください。

さまざまなジャーナル項目のレイアウトに関する情報については、メンバー QSYSINC/H (QP0LJRNL) に同梱されている C 言語組み込みファイル qp0ljrnl.h 内に、統合ファイル・システム・ジャーナル項目固有のデータ内容および形式の詳細が示されています。

| 統合ファイル・システム・オブジェクトに対して設定されるすべてのジャーナル項目の完全なリストについ
| ては、 iSeries Information Center にある『ジャーナル・コード・ファインダー』を参照してください。

ジャーナル処理される操作

以下の操作がジャーナル処理されるのは、その操作が使用するオブジェクトまたはリンクのタイプが、同様にジャーナル処理可能なタイプである場合だけです。

- オブジェクトの作成
- 既存のオブジェクトへのリンクの追加
- リンクの解除
- リンクの名前変更
| • ファイル ID の名前変更
- ディレクトリー内外へのリンクの移動

ジャーナル処理される以下の操作は、ストリーム・ファイルに固有のものです。

- データの書き込み
- ファイルの切り捨て / 拡張
- ファイル・データの強制
- ストレージを解放して保管

ジャーナル処理される以下の操作は、ジャーナル処理されるすべてのオブジェクト・タイプに適用されます。

- 属性の変更 (権限や所有権などのセキュリティー変更を含む)
- オープン
- クローズ
- ジャーナル処理の開始
- ジャーナル処理の終了
- ジャーナル処理済み変更適用 (APYJRNCHG) コマンドの開始

- ジャーナル処理済み変更適用 (APYJRNCHG) コマンドの終了
- 保管
- 復元

統合ファイル・システム・オブジェクトのジャーナル処理についての詳細は、 iSeries Information Center にある『ジャーナル管理』のトピックを参照してください。統合ファイル・システム・オブジェクトに対して設定されるすべてのジャーナル項目の完全なリストについては、**『システム管理』**のトピックの 『ジャーナル・コード・ファインダー』を参照してください。

ジャーナル項目についての特別な考慮事項

ジャーナル処理される統合ファイル・システム操作の多くは、コミットメント制御を内部で使用して、操作中に実行される複数の機能から単一のトランザクションを形成します。 コミットメント制御サイクルに Commit ジャーナル項目 (ジャーナル・コード C、タイプ CM) がなければ、これらのジャーナル処理操作を完了したと考えないでください。ジャーナル処理操作でコミットメント制御サイクルに Rollback ジャーナル項目 (ジャーナル・コード C、タイプ RB) を含むものは失敗した操作であり、それらの中のジャーナル項目は再実行または複製しないでください。

ジャーナル処理される統合ファイル・システム項目で、この方法でコミットメント制御を使用するもの (ジャーナル・コード B) には、以下の項目が含まれます。

- AA － 監査値の変更
- B0 － 作成の開始
- B1 － 要約の作成
- B2 － リンクの追加
- B3 － 名前変更 / 移動
- B4 － リンク解除 (親ディレクトリー)
- B5 － リンク解除 (リンク)
- FA － 属性の変更
- JT － ジャーナル処理の開始 (継承ジャーナル処理属性が Yes であるディレクトリー内での操作によって、ジャーナル処理が開始する場合のみ)
- OA － 権限の変更
- OG － オブジェクト 1 次グループの変更
- OO － オブジェクト所有者の変更

いくつかの統合ファイル・システム・ジャーナル項目には、項目が要約項目であるかどうかを示す固有のデータ・フィールドがあります。要約項目タイプを送信する操作は、2 つの同じ項目タイプをジャーナルに送信します。最初の項目には、項目固有のデータのサブセットが含まれています。 2 番目の項目には、項目固有のデータの全体が含まれており、それが要約項目であることが示されます。オブジェクトを複製する、または操作を再実行する、もしくはその両方を行うプログラムが使用するのは、通常は要約項目だけです。

ジャーナル処理されるディレクトリー内での作成操作では、 B1 ジャーナル項目 (要約の作成) は要約項目と見なされます。

一部のジャーナル操作では、操作に対して逆方向に関連しているジャーナル項目を送信する必要があります。たとえば、B4 ジャーナル項目 (リンクの解除) を含むコミットメント制御サイクルには、 B2 ジャーナル項目 (リンクの追加) も含まれることがあります。このタイプのシナリオが生じるのは、Rollback ジャーナル項目 (C － RB) となる操作だけです。

このシナリオは、次の 2 つの理由で生じることがあります。

1. 操作が失敗する直前であり、エラー・パスのクリーンアップのためにその項目が内部的に必要であったため。
2. 操作がシステム障害によって中断されて、それに続く IPL の際に、その項目を送信する必要のある回復が実行されて、中断された操作をロールバックするため。

付録 A. トランスポート独立リモート・プロシージャー・コール

Sun Microsystems 社によって開発されたリモート・プロシージャー・コール (RPC) は、クライアント・アプリケーションをサーバー機構から容易に分離し、分散させます。これには、外部データ表示または XDR と呼ばれるデータ表示の標準が含まれており、複数のタイプのマシンが転送データにアクセスできるようにします。トランスポート独立 RPC (TI-RPC) は、RPC の最新バージョンです。ネットワーク層で使用される、基礎になるプロトコルを分離する方法を提供し、プロトコル間のさらにシームレスな遷移を提供します。現在 iSeries サーバーで使用可能なプロトコルは、TCP と UDP だけです。

分散アプリケーションのネットワーク全体での開発は、RPC の使用時にはシームレスな作業です。主なターゲットは、ユーザー・インターフェースやデータ検索の分散をさらに重視したアプリケーションです。

ネットワークの選択

以下の API は、アプリケーションの実行の際のトランスポートを選択するための手段を提供します。

これらの API では、*STMF /etc/netconfig ファイルがシステム上に存在していなければなりません。netconfig ファイルが /etc ディレクトリーにない場合、ユーザーはファイルを /QIBM/ProdData/OS400/RPC ディレクトリーからコピーする必要があります。 netconfig ファイルは常に、/QIBM/ProdData/OS400/RPC ディレクトリーにあります。

API	説明
endnetconfig()	netconfig ファイルに保管されるレコードへのポインターを解放する。
freenetconfigent()	呼び出しから getnetconfigent() 関数へ戻される netconfig 構造を解放する。
getnetconfig()	netconfig ファイルの現行レコードへのポインターを戻し、そのポインターを次のレコードを指すようにする。
getnetconfigent()	入力 netid に対応する netconfig 構造へのポインターを戻す。
setnetconfig()	レコード・ポインターを netconfig ファイルの最初の項目に初期設定する。getnetconfig() 関数を最初に使用する前に、setnetconfig() 関数を使用する必要がある。 setnetconfig() 関数は、固有のハンドル (netconfig ファイルに保管されるレコードへのポインター) が getnetconfig() 関数に使用されるように戻す。

名前からアドレスへの変換

以下の API により、アプリケーションは、トランスポート独立の方法で、サービスまたは指定されたホストのアドレスを獲得できます。

API	説明
netdir_free()	名前からアドレスへの変換 API により割り振られる構造を解放する。
netdir_getbyaddr()	アドレスをホスト名およびサービス名にマッピングする。
netdir_getbyname()	サービス・パラメーターで指定されるホスト名とサービス名を、 netconfig 構造で識別されるトランスポートと整合性のある一連のアドレスにマッピングする。
netdir_options()	ブロードキャスト・アドレスおよび TCP と UDP の予約済みポート機能など、トランスポートに特定の機能へのインターフェースを提供する。

API	説明
netdir_sperror()	名前からアドレスへの変換 API の 1 つが失敗した理由を説明する通知メッセージを発行する。
taddr2uaddr()	トランスポート特定 (ローカル) アドレスを、トランスポート独立 (汎用) アドレスに変換する。
uaddr2taddr()	トランスポート独立 (汎用) アドレスを、トランスポート特定 (ローカル) アドレス (netbuf 構造) に変換する。

外部データ表示 (XDR)

以下の API により、リモート・プロシージャー・コール (RPC) アプリケーションは、各種のホストのバイト順序または構造レイアウト規則に関係なく、任意のデータ構造を扱うことができます。

API	説明
xdr_array()	可変長配列とそれに対応する、外部表示間の変換を行うフィルター・プリミティブ。この関数は、配列の各要素をエンコードまたはデコードするために呼び出される。
xdr_bool()	ブール (C 整数) とその外部表示間の変換を行うフィルター・プリミティブ。データのエンコード時に、このフィルターは 1 または 0 のどちらかの値を生成する。
xdr_bytes()	カウントされたバイト配列とその外部表示間の変換を行うフィルター・プリミティブ。この関数は、配列要素のサイズが 1 となっており、各要素の外部記述が組み込み型である総称配列のサブセットを扱う。バイト・シーケンスの長さは、明示的に符号なし整数で指定される。バイト・シーケンスは、ヌル文字で終了することはない。各バイトの外部表示は、その内部表示と同じである。
xdr_char()	C 言語の文字とその外部表示間の変換を行うフィルター・プリミティブ。
xdr_double()	C 言語の倍精度数とその外部表示間の変換を行うフィルター・プリミティブ。
xdr_double_char()	C 言語の 2 バイト文字とその外部表示間の変換を行うフィルター・プリミティブ。
xdr_enum()	C 言語の列挙型 (enum) とその外部表示間の変換を行うフィルター・プリミティブ。
xdr_free()	渡されるポインターにより指されるオブジェクトを再帰的に解放する。
xdr_float()	C 言語の浮動小数点数 (正規化された単一浮動小数点数) と、その外部表示間の変換を行うフィルター・プリミティブ。
xdr_int()	C 言語の整数とその外部表示間の変換を行うフィルター・プリミティブ。
xdr_long()	C 言語の長整数とその外部表示間の変換を行うフィルター・プリミティブ。
xdr_netobj()	可変長の不透明データとその外部表示間の変換を行うフィルター・プリミティブ。
xdr_opaque()	固定長の不透明データとその外部表示間の変換を行うフィルター・プリミティブ。
xdr_pointer()	構造内で追跡するポインターを提供し、ヌル・ポインターをシリアル化する。2 分木やリンク・リストなどの再帰的データ構造を表すことができる。

API	説明
xdr_reference()	構造内で追跡するポインターを提供するフィルター・プリミティブ。このプリミティブにより、別の構造により参照される、ある構造内のポインターのシリアル化、シリアル化解除、および解放を行うことができる。 xdr_reference() 関数は、シリアル化中にヌル・ポインターに特別な意味を付けることはなく、ヌル・ポインターのアドレスを渡すとメモリー・エラーが起きる場合がある。したがって、プログラマーは 2 相判別共用体を使ってデータを記述する必要がある。一方はポインターが有効な場合に使用され、他方はポインターがヌルの場合に使用される。
xdr_short()	C 言語の短整数とその外部表示間の変換を行うフィルター・プリミティブ。
xdr_string()	C 言語のストリングとそれに対応する外部表示間の変換を行うフィルター・プリミティブ。
xdr_u_char()	無符号 C 言語文字とその外部表示間の変換を行うフィルター・プリミティブ。
xdr_u_int()	C 言語の無符号整数とその外部表示間の変換を行うフィルター・プリミティブ。
xdr_u_long()	C 言語の無符号長整数とその外部表示間の変換を行うフィルター・プリミティブ。
xdr_u_short()	C 言語の無符号短整数とその外部表示間の変換を行うフィルター・プリミティブ。
xdr_union()	判別 C 共用体とそれに対応する外部表示間の変換を行うフィルター・プリミティブ。
xdr_vector()	固定長配列とそれに対応する外部表示間の変換を行うフィルター・プリミティブ。
xdr_void()	パラメーターなし。パラメーターを必要とする他の RPC 関数に渡されるが、データの伝送はしない。
xdr_wrapstring()	xdr_string(xdr, sp, maxuint) API を呼び出すプリミティブ。maxuint は無符号整数の最大値。 RPC パッケージは 2 つの XDR 関数の最大値をパラメーターとして渡し、 xdr_string() 関数は 3 つを必要とするので、xdr_wrapstring() が役立つ。

認証

以下の API は、トランスポート独立リモート・プロシージャー・コール (TI-RPC) アプリケーションに認証を提供します。

API	説明
auth_destroy()	auth パラメーターに指される認証情報構造を破棄する。
authnone_create()	各リモート・プロシージャー・コールを使ってヌル認証情報を渡すデフォルトの RPC 認証を作成し、戻す。
authsys_create()	認証情報を含む RPC 認証ハンドルを作成し、戻す。

トランスポート独立 RPC (TI-RPC)

以下の API は、アプリケーションを特定のトランスポート機能から分離することによって、分散アプリケーション開発環境を提供します。これによってトランスポートが使いやすくなります。

TI-RPC 単純化 API

以下の単純化 API は、使用されるトランスポートのタイプを指定します。このレベルを使用するアプリケーションは、明示的にハンドルを作成する必要はありません。

API	説明
rpc_call()	指定されたシステム上でリモート・プロシージャーを呼び出す。
rpc_reg()	RPC サービス・パッケージでプロシージャーを登録する。

TI-RPC 最上位 API

以下の API により、アプリケーションはトランスポートのタイプを指定できます。

API	説明
clnt_call()	クライアントと関連したリモート・プロシージャーを呼び出す。
clnt_control()	クライアント・オブジェクトについての情報を変更する。
clnt_create()	総称クライアント・ハンドルを作成する。
clnt_destroy()	クライアントの RPC ハンドルを破棄する。
svc_create()	サーバー・ハンドルを作成する。
svc_destroy()	RPC サービス・トランスポート・ハンドルを破棄する。

TI-RPC 中間レベル API

以下の API は、最上位 API に類似していますが、ユーザー・アプリケーションがネットワーク選択 API を使用してトランスポート特定情報を選択します。

API	説明
clnt_tp_create()	クライアント・ハンドルを作成する。
svc_tp_create()	サーバー・ハンドルを作成する。

TI-RPC エキスパート・レベル API

以下の API により、アプリケーションは使用するトランスポートを指定できます。また、CLIENT および SVCXPRT ハンドルの詳細の制御のレベルを上げて提供します。これらの API は名前からアドレスへの変換 API を使用して提供される、追加の制御を持つ中間レベル API に類似しています。

API	説明
clnt_tli_create()	クライアント・ハンドルを作成する。
rpcb_getaddr()	サービスの汎用アドレスを検出する。
rpcb_set()	サーバー・アドレスを RPCbind で登録する。
rpcb_unset()	サーバーがそのアドレスを抹消するために使用する。
svc_reg()	プログラムとバージョンをディスパッチに関連付ける。
svc_tli_create()	サーバー・ハンドルを作成する。
svc_unreg()	svc_reg() によってアソシエーション・セットを削除する。

他の TI-RPC API

以下の API により、さまざまなアプリケーションが、単純化、最上位、中間レベル、およびエキスパート・レベル API と協同で作業することができます。

API	説明
clnt_freeres()	RPC または XDR システムが割り振るデータを解放する。
clnt_geterr()	クライアント・ハンドルからエラー構造を入手する。
svc_freeargs()	RPC または XDR システムが割り振るデータを解放する。
svc_getargs()	RPC 要求の引き数をデコードする。
svc_getrpccaller()	呼び出し元のネットワーク・アドレスを入手する。
svc_run()	RPC 要求が来るのを待機する。
svc_sendreply()	プロシージャー呼び出しの結果をリモート・クライアントに送信する。
svcerr_decode()	デコード・エラーについて情報をクライアントに送る。
svcerr_noproc()	プロシージャー番号エラーについて情報をクライアントに送る。
svcerr_systemerr()	システム・エラーについて情報をクライアントに送る。

付録 B. 統合ファイル・システムの C 関数を使用するプログラムの例

この簡単な C 言語プログラムは、さまざまな統合ファイル・システム関数の使用を示すものです。例に挙げたプログラムは、以下のような操作を行います。

1. **getuid()** 関数を使用して、実際のユーザー ID (uid) を判別します。
2. **getcwd()** 関数を使用して、現行ディレクトリーを判別します。
3. **open()** 関数を使用して、ファイルを作成します。所有者 (ファイルの作成者) に、ファイルの読み取り権限、書き込み権限、および実行権限を与えます。
4. **write()** 関数を使用して、ファイルにバイト・ストリングを書き込みます。オープン操作 (3) で提供されたファイル記述子がファイルを識別します。
5. **close()** 関数を使用して、ファイルをクローズします。
6. **mkdir()** 関数を使用して、現行ディレクトリーに新しいサブディレクトリーを作成します。所有者には、サブディレクトリーの読み取りアクセス権、書き込みアクセス権、および実行アクセス権が与えられます。
7. **chdir()** 関数を使用して、新しいサブディレクトリーを現行ディレクトリーにします。
8. **link()** 関数を使用して、以前に作成したファイル (3) にリンクを作成します。
9. **open()** 関数を使用して、読み取り専用としてファイルをオープンします。前に作成したリンク (8) によってファイルにアクセスできます。
10. **read()** 関数を使用して、ファイルからバイト・ストリングを読み取ります。オープン操作 (9) で提供されたファイル記述子がファイルを識別します。
11. **close()** 関数を使用して、ファイルをクローズします。
12. **unlink()** 関数を使用して、ファイルへのリンクを除去します。
13. **chdir()** 関数を使用して、新しいサブディレクトリーが作成された親ディレクトリーに、現行ディレクトリーを戻します。
14. **rmdir()** 関数を使用して、以前に作成したサブディレクトリー (6) を削除します。
15. **unlink()** 関数を使用して、以前に作成したファイル (3) を削除します。

注: このサンプル・プログラムは、実行されるジョブの CCSID が 37 であるシステムで正常に稼働します。統合ファイル・システム API には、ジョブの CCSID でエンコードされたオブジェクト、およびパス名がなければなりません。しかし、C コンパイラーは CCSID 37 の文字固定情報を保管します。互換性を完全にするには、オブジェクトやパス名のような文字固定情報を、 API をジョブの CCSID に渡す前に変換する必要があります。

コード例に関する特記事項を参照してください。

```
#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;
}
```

付録 C. 統合ファイル・システムの RPG コードの例

Code Snippets  には、統合ファイル・システムの RPG コードの例があります。この例を表示するには、以下のステップを実行してください。

1. 「検索」カテゴリーのドロップダウン・リストから**「ILE RPG ソース (ILE RPG Source)」**を選択します。
2. **「検索」**をクリックします。
3. リストをスクロールダウンして、**「IFS を RPG から使用する (Using IFS from RPG)」**が表示されるようにします。
4. **「IFS を RPG から使用するためのコード (Code for using IFS from RPG)」**をクリックします。

コード例に関する特記事項を参照してください。

参照文献

この参照文献リストでは、本書で説明した情報の背景または詳細が記載されている iSeries サーバー情報をリストしています。

- iSeries Information Center の**『プログラミング』**のカテゴリーに記載されている『制御言語』のトピックでは、 iSeries サーバーの制御言語 (CL) とそのコマンドについて説明しています。各コマンドの説明では、構文図、パラメーター、デフォルト値、キーワード、および例を使用しています。
- iSeries Information Center の『グローバリゼーション』のトピックでは、文字セットやコード・ページなどの各国語サポート (NLS) の概念について説明し、 iSeries サーバー NLS および多言語機能の評価、計画、使用に必要な情報を提供します。
- iSeries Information Center の**『プログラミング』**のカテゴリーに記載されている『API』のトピックでは、各 OS/400 API (統合ファイル・システム API を含む) について説明しています。
- iSeries Information Center の**『システム管理』**のカテゴリーに記載されている『ジャーナル管理』のトピックでは、 iSeries サーバー上でのシステム管理アクセス・パス保護 (SMAPP) のセットアップ、管理、およびトラブルシューティングについて説明しています。
- iSeries Information Center の**『データベース』**のカテゴリーに記載されている『コミットメント制御』のトピックでは、データベース・ファイルまたは統合ファイル・システム・ファイルなどのリソースへの変更のグループを作業論理単位として定義および処理する方法について説明します。
- OS/400 ネットワーク・ファイル・システム・サポート。この資料では、一連のリアル・ライフ・アプリケーションを通してネットワーク・ファイル・システムについて説明します。エクスポート、マウント、ファイル・ロック、およびセキュリティーに関する考慮事項が記載されています。この資料には、NFS を使用してセキュア・ネットワークのネーム・スペースを構成、開発する方法が記載されています。
- オプティカル・サポート。この資料は、OS/400 での IBM オプティカル・サポートについての、ユーザーの手引きおよび解説書として作成されています。この資料は、光ディスク・ライブラリー・データ・サーバーの概念を理解したり、光ディスク・ライブラリーの計画を立てたり、光ディスク・ライブラリー・データ・サーバーの管理や操作をしたり、光学式データ・サーバーの問題を解決したりするのに役立ちます。
- WebSphere Development Studio: ILE C/C++ Programmers Guide。この資料は、iSeries サーバー上で ILE C/400 プログラムを設計、編集、コンパイル、実行、およびデバッグするために必要な情報を説明しています。
- WebSphere Development Studio: C/C++ Language Reference。この資料は、ILE C/400 プログラムの構造、およびライブラリー関数とインクルード (ヘッダー) ファイルの詳細を説明しています。
- 機密保護解説書。この資料は、OS/400 セキュリティーに関する技術情報の詳細を説明しています。
- APPC プログラミング。この資料では、iSeries サーバーの拡張プログラム間通信機能 (APPC) サポートについて説明しています。APPC を使用するアプリケーション・プログラムの開発、および APPC の通信環境の定義の手引きを記載しています。
- バックアップおよび回復の手引き。この資料では、IBM iSeries サーバーのリカバリーおよび可用性オプションに関する一般情報を記載しています。

参照文献

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

アクセス・モード 62
エンコード・スキーム 24, 63
オープン・システム (QOpenSys) ファイル・システム
  説明 4
  特性および制限事項 71
オープン・ファイル記述子 62
オブジェクト
  複数のファイル・システムにわたるマイグレーション 37

[カ行]

階層ファイル・システム (HFS)
  QDLS ファイル・システム用 API の使用 83
  QOPT ファイル・システム用 API の使用 86
拡張属性
  各国語間での維持 25, 63
  説明 23
  命名規則 24
関数
  パス名規則 61
  プログラムの例 113
  C プログラムでの使用 49, 54
  ILE C/400 59
関連資料 121
許可 62
国別言語サポート 2, 24, 63
権限
  コマンド 28
  プログラムでの処理 62
  プログラムの例 113
  QFileSvr.400 ファイル・システムでの制限事項 96
  QNTC ファイル・システムでの制限事項 91
現行ディレクトリー 9
コード・ページ 2, 24, 63
コマンド
  使用 28
  パス名規則 31
  リスト 28

[サ行]

作業ディレクトリー 9
参照文献 121
ジャーナル処理
  開始 47
  終了 47
シンボリック・リンク
  使用の例 21
  説明 21
  ハード・リンクとの比較 23
ストリーム・ファイル
  使用の利点 3
  説明 3
  データベース・ファイルへの (からの) コピー 50
  プログラムでの使用 49
  プログラムの例 113
  利点 2
  レコード単位ファイルとの比較 3
  ILE C/400 での使用法の指示 59
セキュリティー
  コマンド 28
  プログラムでの処理 62
  QFileSvr.400 ファイル・システムでの制限事項 96
  QNTC ファイル・システムでの制限事項 91
絶対パス名 18
操作 (プログラム例) 113
相対パス名 19
ソケット 63

[タ行]

データベース・ファイル
  ストリーム・ファイルからの作成 50
  ストリーム・ファイルとの比較 3
  ストリーム・ファイルへの (からの) コピー 50
データ変換 64
ディレクトリー
  現行 9
  コマンド 28
  説明 6
  統合ファイル・システム 38
  プログラムの例 113
  ホーム 9
  メニューおよび表示画面 28
  利点 1
テキスト・オープン・ファイル・モード 64

統合ファイル・システム
コマンド
使用 28
パス名規則 31
リスト 28
使用の利点 1
説明 1
プログラミング・インターフェース
国別言語サポート 63
セキュリティー 62
パス名規則 61
プログラムの例 113
ポインターおよびファイル記述子 62
C プログラムでの使用 54
メニューおよび表示画面
使用 27
パス名規則 31
PC からの作業
ファイル・システムの表示方法 36
PC からの操作
ディレクトリーおよびオブジェクトとの対話 34
統合ファイル・システム・インターフェース 1, 2, 5
統合ファイル・システム・オブジェクトのジャーナル処理 101
独立 ASP QSYS.LIB
特性および制限事項 80
独立 ASP QSYS.LIB ファイル・システム
説明 5

[ナ行]

名前
エンコード・スキーム間における維持 24
各国語間での維持 63
独立 ASP QSYS.LIB ファイル・システムでの使用 81
QDLS ファイル・システムでの使用 84
QFileSvr.400 ファイル・システムでの使用 94
QNTC ファイル・システムでの使用 91
QOpenSys ファイル・システムでの使用 71
QOPT ファイル・システムでの使用 86
QSYS.LIB ファイル・システムでの使用 79
/ (ルート) ファイル・システムでの使用 70
ネットワーク・ファイル・システム 5
説明 5
特性および制限事項 97

[ハ行]

ハード・リンク
シンボリック・リンクとの比較 23
説明 20
バイナリー・オープン・ファイル・モード 64
パス名
コマンドまたは表示画面の規則 31
絶対パス名 18
説明 18
相対パス名 19
独立 ASP QSYS.LIB ファイル・システムでの使用 82
API の規則 61
QDLS ファイル・システムでの使用 84
QFileSvr.400 ファイル・システムでの使用 94
QNTC ファイル・システムでの使用 91
QOpenSys ファイル・システムでの使用 72
QOPT ファイル・システムでの使用 86
QSYS.LIB ファイル・システムでの使用 79
/ (ルート) ファイル・システムでの使用 70
光 (QOPT) ファイル・システム
説明 5
特性および制限事項 85
表示画面
使用 27
パス名規則 31
ファイル 113
オープン・モード 64
転送 34
メニューおよび表示画面 28
ファイル記述子 62
ファイル転送プロトコル 34
ファイル・サーバー 6
ファイル・サーバー I/O プロセッサー 6
ファイル・システム
インターフェース 5
オープン・システム (QOpenSys)
説明 4
特性および制限事項 71
オブジェクトのマイグレーション 37
説明 4
独立 ASP QSYS.LIB
説明 5
特性および制限事項 80
ネットワーク・ファイル・システム
説明 5
特性および制限事項 97
比較 65
光 (QOPT)
特性および制限事項 85
光ファイル・システム (QOPT)
説明 5
ファイルの転送 34
文書ライブラリー・サービス (QDLS)
説明 5
特性および制限事項 83

ファイル・システム *(続き)*
ユーザー定義ファイル・システム
説明 5
特性および制限事項 72
ライブラリー (QSYS.LIB)
説明 5
特性および制限事項 77
利点 2
ルート (/)
説明 4
特性および制限事項 69
NetWare ファイル・システム (QNetWare)
特性および制限事項 87
OS/400 ファイル・サーバー (QFileSvr.400)
特性および制限事項 93
QFileSvr.400 ファイル・システム (QFileSvr.400)
説明 5
QNetWare ファイル・システム
説明 5
QNTC サーバー
特性および制限事項 91
QNTC ファイル・システム
説明 5
フォルダー
QDLS ファイル・システム 5, 83
文書ライブラリー・サービス (QDLS) ファイル・システム
説明 5
特性および制限事項 83
変換
オブジェクト名 24, 63
データ 64
ホーム・ディレクトリー 9
ポインター 62
保管ファイル
独立 ASP QSYS.LIB ファイル・システムでの使用 81
QSYS.LIB ファイル・システムでの使用 78

［マ行］

マイグレーション、複数のファイル・システムにわたる 37
メニュー
使用 27
パス名規則 31
モード、アクセスの 62
モード、オープン・ファイルの 64
文字変換 2, 24, 63

［ヤ行］

ユーザー定義ファイル・システム 5
説明 5
特性および制限事項 72
ユーザー・インターフェース
コマンド 28
メニューおよび表示画面 27
PC からのビュー 36
ユーザー・スペース
独立 ASP QSYS.LIB ファイル・システムでの使用 81
QSYS.LIB ファイル・システムでの使用 78

［ラ行］

ライブラリー (QSYS.LIB) ファイル・システム
説明 5
特性および制限事項 77
リンク
コマンド 28
使用の利点 19
シンボリック 21
説明 19
独立 ASP QSYS.LIB ファイル・システムでの使用 82
ハード 20
比較 23
プログラムの例 113
メニューおよび表示画面 28
QDLS ファイル・システムでの使用 84
QFileSvr.400 ファイル・システムでの使用 96
QNTC ファイル・システムでの使用 92
QOpenSys ファイル・システムでの使用 72
QOPT ファイル・システムでの使用 87
QSYS.LIB ファイル・システムでの使用 79
/ (ルート) ファイル・システムでの使用 70
ルート (/) ファイル・システム
説明 4
特性および制限事項 69
例
シンボリック・リンクの使用 21
統合ファイル・システム API を使用するプログラム 113
パス名 18, 31, 61

A

API
パス名規則 61
プログラムの例 113
C プログラムでの使用 49, 54

API *(続き)*
ILE C/400 59

C

C 言語プログラム
例 113
ILE C/400 の関数 59
Client Access 35

F

FTP 34

I

ILE C/400
同等の API 54
ANSI 関数 59

L

LU 6.2、QFileSvr.400 ファイル・システムでの 94

N

NetServer 35
NetWare ファイル・システム (QNetWare)
特性および制限事項 87

O

OS/400 ファイル・サーバー 6
OS/400 ファイル・サーバー (QFileSvr.400) ファイル・システム
特性および制限事項 93

P

PC クライアント
統合ファイル・システムでの作業 34
ファイル・システムの表示方法 36
PC ファイル・サーバー 6

Q

QDLS ファイル・システム
説明 5
特性および制限事項 83
QFileSvr.400 5
QFileSvr.400 (QFileSvr.400) ファイル・システム
説明 5
QFileSvr.400 ファイル・システム
説明 5
特性および制限事項 93
QNetWare ファイル・システム 5
説明 5
特性および制限事項 87
QNTC ファイル・システム 5
説明 5
特性および制限事項 91
QOpenSys ファイル・システム
説明 4
特性および制限事項 71
QOPT 5
QOPT ファイル・システム
説明 5
特性および制限事項 85
QSYS.LIB ファイル・システム
説明 5
特性および制限事項 77

T

TCP/IP、QFileSvr.400 ファイル・システムでの 94

U

Unicode 24

W

Windows NT Server ファイル・システム (QNTC)
特性および制限事項 91

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12