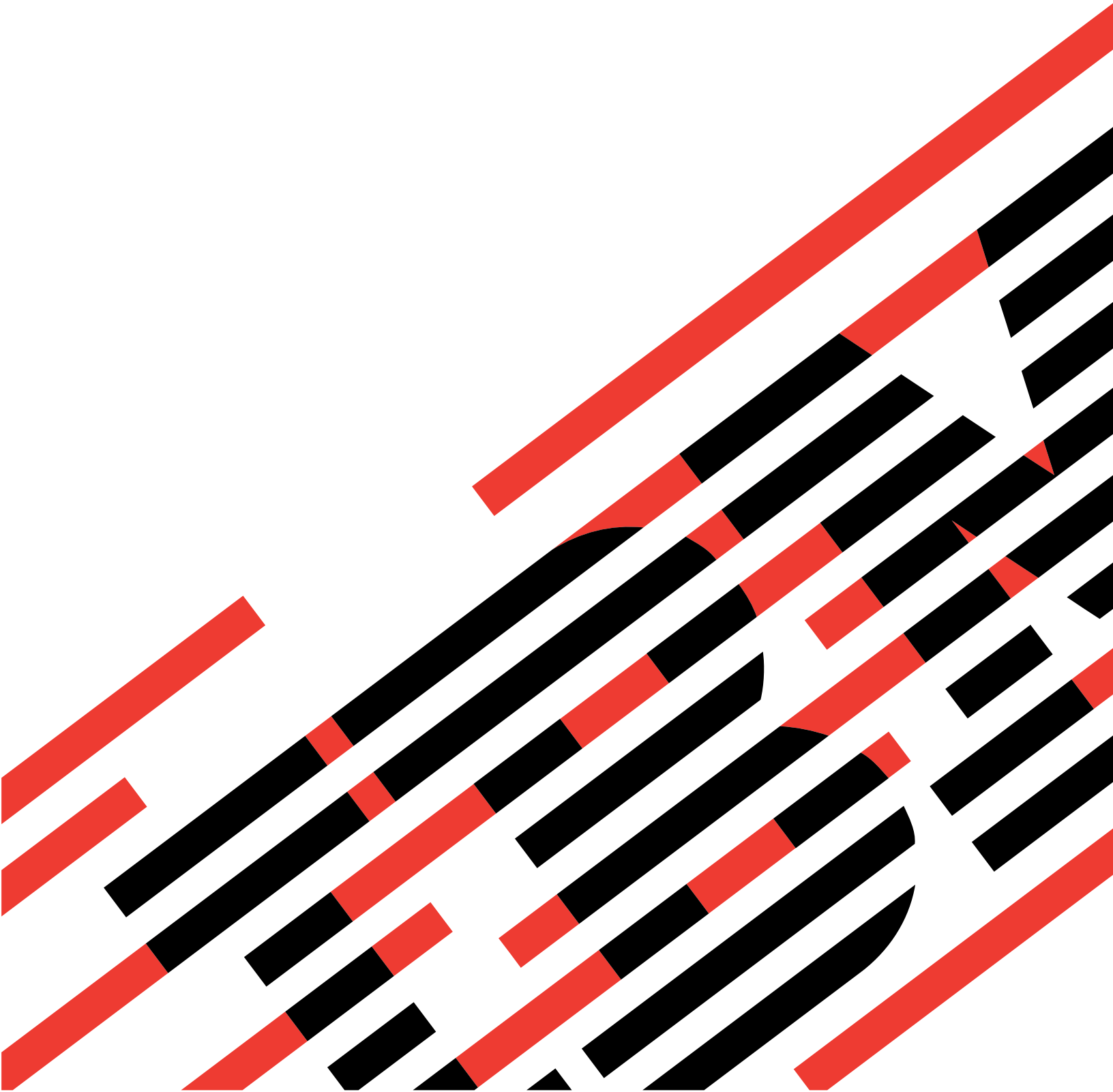
IBM

eserver

iSeries

統合ファイル・システム

バージョン 5 リリース 3



IBM

@server

iSeries

統合ファイル・システム

バージョン 5 リリース 3

ご注意！

本書および本書で紹介する製品をご使用になる前に、 135 ページの『特記事項』に記載されている情報をお読みください。

| 本書は、IBM OS/400 (プロダクト番号 5722-SS1) のバージョン 5、リリース 3、モディフィケーション 0 に適用されます。また、改訂版で断りがない限り、それ以降のすべてのリリースおよびモディフィケーションに適用されます。このバージョンは、すべての RISC モデルで稼働するとは限りません。また CISC モデルでは稼働しません。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

`http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： iSeries
Integrated File System
Version 5 Release 3

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2005.8

この文書では、平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、平成角ゴシック体™W7

目次

統合ファイル・システム

統合ファイル・システムは OS/400® の一部であり、パーソナル・コンピューターや 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およびストレージ管理をサポートします。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。

OS/400 での統合ファイル・システムに関する詳細情報については、以下のトピックを参照してください。

V5R3 の新機能
以前のリリースからのこのトピックの変更点を確認してください。

印刷可能な PDF
この情報の PDF バージョンを表示または印刷する方法。

統合ファイル・システムの概要
統合ファイル・システムの基本について説明します。

概念
統合ファイル・システムに関するさまざまな概念を示します。

ファイル・システムの処理
OS/400 に存在するさまざまなファイル・システムを管理する方法について説明します。

統合ファイル・システムへのアクセス
統合ファイル・システムにアクセスするさまざまなオプションについて説明します。メニューと表示画面の使用、CL コマンド、API、 iSeries™ ネットサーバー、 iSeries ナビゲーター、およびファイル転送プログラムの使用について説明します。

***TYPE1 から *TYPE2 へのディレクトリーの変換**
システムが *TYPE1 ディレクトリーから *TYPE2 ディレクトリーに自動変換する方法を説明します。

オブジェクトのジャーナル処理
統合ファイル・システム・オブジェクトのジャーナル処理を開始、終了、および変更する方法について説明します。

プログラミング・サポート
データのコピー、API の使用、その他のサポート機能の使用方法について説明します。

iSeries ナビゲーターを使用したファイルおよびフォルダーの処理
ファイルのチェックインとチェックアウト、フォルダーの作成と除去など、ファイルとフォルダーに対する操作について説明します。

統合ファイル・システムについての関連情報
統合ファイル・システムに関する追加のトピックが、 iSeries Information Center およびインターネットで提供されています。

注: 法律上の重要な情報について、 2 ページの『コードに関する特記事項』をご覧ください。

コードに関する特記事項

本書には、プログラミングの例が含まれています。

IBM® は、お客様に、すべてのプログラム・コードのサンプルを使用することができる非独占的な著作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。従って 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、いかなる保証も適用されません。商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任の保証の適用も一切ありません。

V5R3 の新機能

今回のリリースでは、以下の点が追加または更新されています。

統合ファイル・システム API

統合ファイル・システムでは、以下の API に関する機能およびサポートが追加されました。

- ファイルのクリア (fclear())
- ファイルのクリア (ラージ・ファイル使用可能) (fclear64())
- スキャン・シグニチャーの変更 (QP0LCHSG)
- 参照オブジェクトの検索 (QP0LRRO)
- スキャン・シグニチャーの検索 (QP0LRTSG)

統合ファイル・システム CL コマンド

V5R3 では、以下の CL コマンドに関する機能およびサポートが統合ファイル・システムに追加されました。

- ジャーナル・オブジェクトの変更 (CHGJRNOBJ) コマンド
- ディレクトリー情報の出力 (PRTDIRINF) コマンド
- ディレクトリー情報の検索 (RTVDIRINF) コマンド

PRTDIRINF コマンドと RTVDIRINF コマンドの出力については、 72 ページの『RTVDIRINF および PRTDIRINF コマンドの出力の処理』を参照してください。

2 つのデバイス・ドライバー

「ルート」(/) ファイル・システムの /dev/xti ディレクトリーの下に、 udp および tcp の 2 つのデバイス・ドライバーが格納されるようになりました。以下のトピックは、これらのデバイス・ドライバーについて説明しています。

- 32 ページの『「ルート」(/) ファイル・システムでの UDP および TCP デバイス』

QNTC ファイル・システム用の Kerberos 構成

| 以前のリリースでは、 QNTC がパスワードを使用してリモート・サーバーとの間で認証を行いました。パスワード管理を単純化するために、 QNTC は新しく Kerberos サーバー・チケットを使用してリモート・サーバーとの間で認証するようになりました。 QNTC ファイル・システムで Kerberos を使用するよう構成する方法について、詳しくは、 57 ページの『QNTC ファイル・システムでの Kerberos の使用可能化』を参照してください。

***TYPE2 ディレクトリーへの自動変換**

V5R3 がインストールされるとすぐに、システムは、現在 *TYPE1 ディレクトリーをサポートしているファイル・システムをすべて *TYPE2 ディレクトリーに自動変換し始めます。詳しくは、 90 ページの『*TYPE1 から *TYPE2 へのディレクトリーの変換』を参照してください。

スキャンのサポート

iSeries に新しく追加されたオプションの機能を使用して、統合ファイル・システム・オブジェクトをスキャンできるようになりました。スキャンは、オブジェクトのオープンまたはクローズ操作時に行われます。ウィルス検索など、さまざまな目的でオブジェクトをスキャンすることができます。また、実際の要件に応じて、スキャンがいつ行うかを選択できます。以下のトピックには、スキャン・サポートに関する追加情報、およびスキャンを実行するための要件が説明されています。

- 20 ページの『スキャンのサポート』

トピックの印刷

本書の PDF 版を表示またはダウンロードするには、『統合ファイル・システム』 (約 1,184 KB) を選択します。

PDF ファイルの保存

表示または印刷のために PDF をワークステーションに保存するには、以下のようにします。

1. ブラウザーで PDF を右マウス・ボタン・クリックする (上部のリンクを右マウス・ボタン・クリック)。
2. Internet Explorer を使用している場合は、「**対象をファイルに保存...**」をクリックする。 Netscape Communicator を使用している場合は、「**リンクを名前を付けて保存...**」をクリックする。
3. PDF を保存したいディレクトリーに進む。
4. 「**保存**」をクリックする。

Adobe Acrobat Reader のダウンロード

これらの PDF を表示または印刷するには、Adobe Acrobat Reader が必要です。このアプリケーションは Adobe Web サイト (www.adobe.com/products/acrobat/readstep.html) からダウンロードできます。

統合ファイル・システムの概要

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し、サーバーでの使用方法を紹介します。

- 4 ページの『統合ファイル・システムとは』
- 4 ページの『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します。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・システムは 11 のファイル・システムからなり、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。詳しくは、『ファイル・システムの処理』を参照してください。

統合ファイル・システムの主な機能は、次のとおりです。

- ストリーム・ファイルへは長い連続ストリングのデータを入れることができます。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。
- 階層ディレクトリー構造では木の枝に実がつくようにオブジェクトを編成されます。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。
- 共通インターフェース。この共通インターフェースにより、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することができます。
- ご使用のサーバー、iSeries の統合 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。さらに、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。

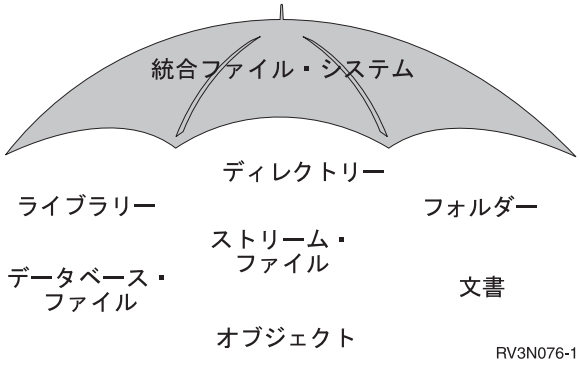


図 1. iSeries サーバーに保管される全情報の構造

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、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) をサポートすることにより、OS/400 の広範なデータ管理機能をさらに拡張します。

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。また、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、 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 (レコード単位のデータベース・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って管理しながら、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。

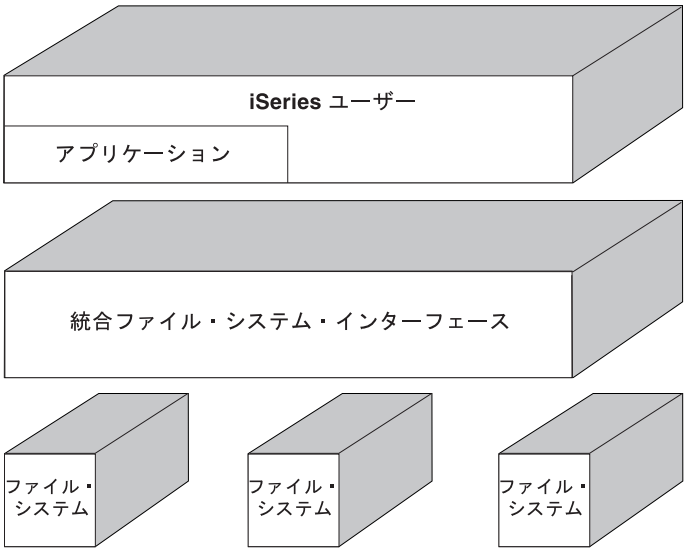


図 2. 別々のファイル・システムへの共通インターフェース

- 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。
- オブジェクト名および関連するオブジェクト情報の継続性を、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。

概念

このセクションでは、統合ファイル・システムの概念について説明します。ご使用のサーバーで統合ファイル・システムを利用する方法について、以下のトピックをご覧ください。

- 6 ページの『ディレクトリー』
- 12 ページの『リンク』
- 16 ページの『パス名』
- 17 ページの『ストリーム・ファイル』
- 18 ページの『名前の継続性』
- 19 ページの『拡張属性』
- 20 ページの『スキャンのサポート』

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される、特殊なオブジェクトです。各ディレクトリーには、それに属するオブジェクトのリストが入っています。リストには、他のディレクトリーが含まれる場合もあります。

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、それらの枝に付い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内の 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。そのほか、ファイルを一度だけ保管し、リンクを使って複数のパスによってアクセスするなど、 UNIX システムの標準機能も提供します。

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。

- 7 ページの『現行ディレクトリー』
- 8 ページの『ホーム・ディレクトリー』
- 8 ページの『提供されたディレクトリー』
- 10 ページの『*TYPE2 ディレクトリー』

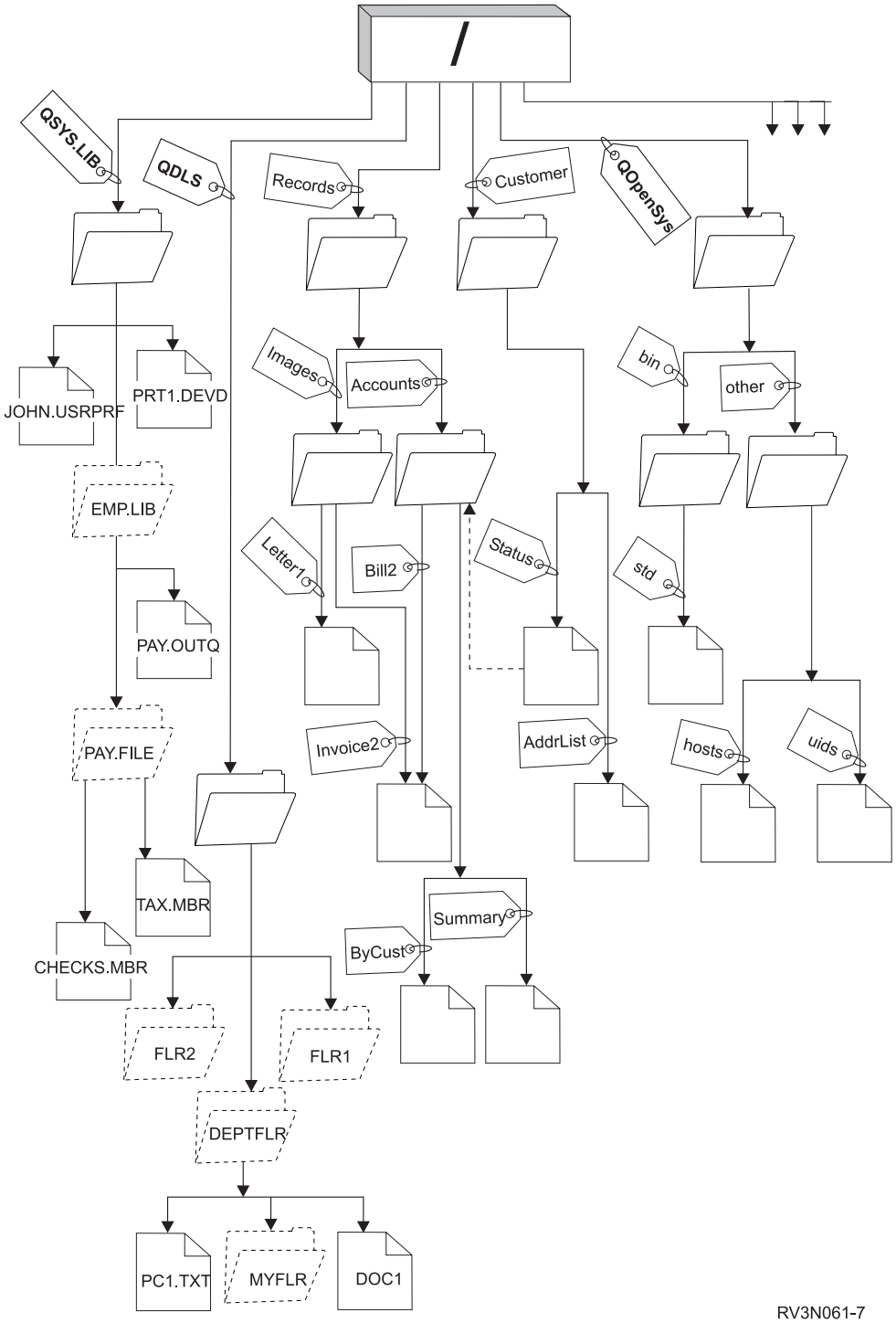


図 3. ファイル・システムおよびオブジェクトは、統合ファイル・システムのディレクトリー・ツリーの枝である

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に対する操作を要求するとき、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

リー内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。

ホーム・ディレクトリー

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・プロファイルで指定されています。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。

通常、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のディレクトリーを現行ディレクトリーとして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・ディレクトリーが、個々のスレッドのホーム・ディレクトリーにもなります。これは、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。ただし、ジョブの変更 (QWTCHGJB) API を使用すれば、スレッドで使用されるホーム・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 (API)』のトピックを参照してください。

提供されたディレクトリー

システムの再始動時に、以下のディレクトリーが存在しない場合、統合ファイル・システムによって作成されます。

/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、ユーザーまたはアプリケーションによって除去されない限り、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 (87 ページの『API を使用したアクセス』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。

/tmp ディレクトリーは、「名前変更およびリンク解除の制限」属性がオンになった状態で提供されるようになりました。属性の変更コマンド CHGATR を使用して、この属性を変更できます。ユーザーはこの属性をオフにすることができますが、システムが次に再始動するとき、自動的に再びオンになります。この属性がオンであるとき、 /tmp は以下のような制限を実施します。

- このディレクトリー内のオブジェクトに対する名前変更とリンク解除が制限されます。オブジェクトをこのディレクトリー内にリンクすることができますが、操作を実行しようとするユーザーに関して以下の 1 つまたは複数が該当しない限り、名前変更やリンク解除を行うことができません。
  1. ユーザーがオブジェクトの所有者である
  2. ユーザーがディレクトリーの所有者である
  3. ユーザーが特殊権限 *ALLOBJ を持っている。

/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。通常、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に設定します (たとえば /home/john)。詳しくは、8 ページの『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が入るサブディレクトリーがあります。通常、/usr 内のファイルは頻繁には変更されません。

/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、システムによって提供されます。

/QIBM/ProdData

/QIBM/ProdData ディレクトリーは、ライセンス・プログラム・データ用のシステム・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・データ用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・システム用のシステム・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・システム用のシステム・ディレクトリーで、ライセンス・プログラムのデータ用に使用されます。

/QOpenSys/QIBM/UserData

/QOpenSys/QIBM/UserData ディレクトリーは、 QOpenSys ファイル・システム用のシステム・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・データ用に使用されます。

/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・データ用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。

/dev /dev ディレクトリーには、さまざまなシステム・ファイルとディレクトリーが入ります。

/dev/xti

/dev/xti ディレクトリーには、 UDP および TCP デバイス・ドライバーが格納されます。詳しくは、 32 ページの『「ルート」(/) ファイル・システムでの UDP および TCP デバイス』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
- (通常は) より少ない補助記憶域スペース

*TYPE2 ディレクトリーは *TYPE1 ディレクトリーと比べて、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 ディレクトリーを完全に回復するには、 記憶域の再利用 (RCLSTG) コマンドを使用する必要があるかもしれません。

*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 リンクに比べて、最大で 1,000,000 個のリンクを持つことができます。つまり、ストリーム・ファイルへのハード・リンクが最大で 1,000,000 個まで可能であり、 *TYPE2 ディレクトリーに最大で 999,998 個までのサブディレクトリーを含めることができます。
3. iSeries ナビゲーターを使用すると、 *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、エントリーのリストが自動的に 2 進数の順序でソートされます。
4. 一部の新機能 (たとえば統合ファイル・システムのスキャン・サポート) は、 *TYPE2 ディレクトリー内のオブジェクトに対してのみ実行することができます。

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 (平均して) 10 % 大きくなります。

ご使用のシステム上で *TYPE2 ディレクトリーを設定するには、いくつかの方法があります。

- OS/400 V5R2 以降が事前インストールされる新規 iSeries サーバーには、 *TYPE2 ディレクトリーが含まれます。 ASP 1 から 32 内の「ルート」(/)、QOpenSys、および UDFS の変換は必要ありません。
- iSeries サーバーに OS/400 V5R2 以降をスクラッチ・インストールすると、 *TYPE2 ディレクトリーが含まれます。 ASP 1 から 32 内の「ルート」(/)、QOpenSys、および UDFS の変換は必要ありません。
- V5R1 または V5R2 の変換ユーティリティーを使用してファイル・システムを変換します。
- 独立補助記憶域プール (ASP) 内の UDFS がまだ *TYPE2 フォーマットに変換されていない場合、OS/400 V5R2 以降がインストールされたシステムで初めて独立 ASP をオンに変更したときに UDFS が変換されます。
- まだ *TYPE1 を使用している独立 ASP 上の UDFS を除いて、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、システムによって自動的に変換されます。この変換は V5R3 のインストール後に開始されますが、システム・アクティビティーには大きな影響を与えません。追加情報については、 90 ページの『*TYPE1 から *TYPE2 へのディレクトリーの変換』を参照してください。

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、以下の 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 以降でサポートされていますが、通常の *TYPE2 ディレクトリー・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 または V5R2 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、 90 ページの『*TYPE1 から *TYPE2 へのディレクトリーの変換』を参照してください。

OS/400 V5R1 または V5R2 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、 V5R2、およびそれ以降で *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、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 または V5R2 をご使用の場合、適切な変換ユーティリティーを使用して、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できます。 V5R3 がインストールされるとすぐに、*TYPE2 ディレクトリーをサポートするようまだ変換されていないすべてのファイル・システムに関して、

*TYPE2 ディレクトリーへの自動変換が開始されます。この自動変換を実行したくない場合には、 V5R3 のインストール前に *TYPE2 ディレクトリー・フォーマットへの変換を行うことができます。詳しくは、90 ページの『*TYPE1 から *TYPE2 へのディレクトリーの変換』を参照してください。

V5R2 で *TYPE2 ディレクトリー・サポートを使用可能にするには、V5R2 iSeries Information Center のディレクトリーの変換 (CVTDIR) コマンドを使用します。

V5R1 での *TYPE2 ディレクトリー・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 の *TYPE2 ディレクトリーに関する詳細な資料については、情報 APAR II13161 を参照してください。 APAR にアクセスするには、以下のいずれかの方法を使用します。

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.ibm.com/eserver/iseries/support/supporthome.nsf/document/10000045  Web サイトに移動して、情報 APAR をオンライン表示します。「**問題解決 (Problem Solving)**」→「**テクニカル・データベース (Technical Databases)**」→「**許可プログラム分析報告書 (Authorized Program Analysis Reports (APAR))**」→「**V5R1 APAR**」→「**APAR 番号 II13161 (APAR number II13161)**」を選択します。

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関連付けです。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指定して、サーバーに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、オブジェクトのことを、サーバーで識別するための名前をもつファイルのようなものと考えることができます。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、ディレクトリー・パスによってオブジェクトを識別す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、リンクに付けられます。

オブジェクトではなく、リンクに名前が付けられるという観念に慣れてくると、以前には考えられなかった多くの使用法が見えてきます。 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 (8 ページの『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・システムにまたがって設定できるものや、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があります。

リンクには、 13 ページの『ハード・リンク』と 14 ページの『シンボリック・リンク』の 2 種類があります。プログラムでパス名を使用するとき、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選択できます。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、各項目ごとに 2 つの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します。	遅い。シンボリック・リンクにはオブジェクトへのパス名が含まれていて、オブジェクトを検出するために、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・リンクを設定するには、オブジェクトが存在しなければなりません。	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・リンクによって参照されていても削除できます。
静的オブジェクト (属性が変更されない場合)	速い。静的オブジェクトの場合、パフォーマンスに影響する最大の要素はネーム・レゾリューションです。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・システムにまたがって設定できません。	制限なし。シンボリック・リンクは、複数のファイル・システムにまたがって設定できます。

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』

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。 (ファイルをディレクトリーにコピーするなどの方法で) ディレクトリー内にオブジェクトが作成されると、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複数のリンクを同じ名前にすることはできませんが、異なるディレクトリーからの複数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 つのオブジェクトへの複数のハード・リンクを設定することができます (同じディレクトリーから、または異なるディレクトリーから)。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。ディレクトリーと別の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。

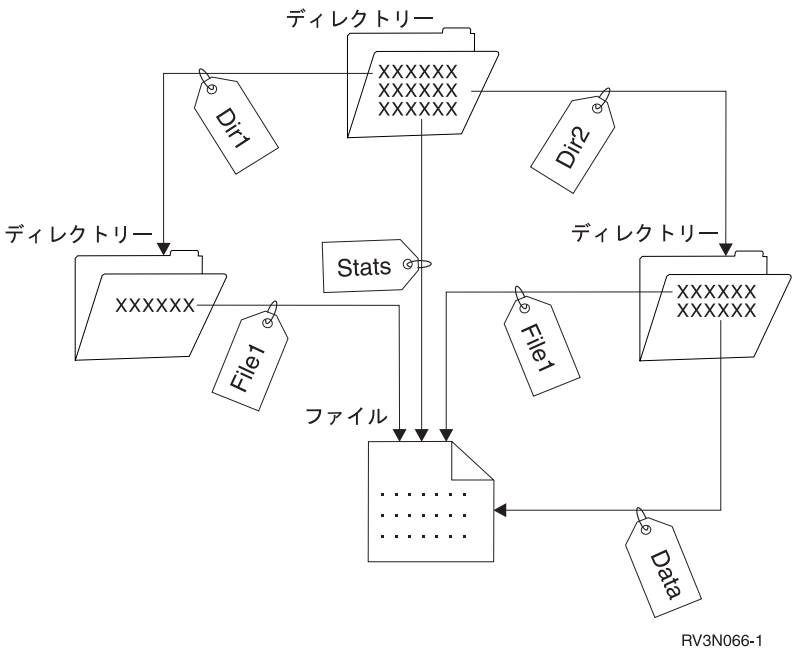


図4. 各ハード・リンクを定義するディレクトリー項目

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 つ残っている限り、オブジェクトの存在に影響を与えることなくハード・リンクを除去することができます。最後のハード・リンクが除去されるとき、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、その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各アプリケーションは、そのアプリケーションがオブジェクトをクローズするまでオブジェクトを使用できます。すべてのアプリケーションがオブジェクトを使用し終わると、そのオブジェクトはサーバーから削除されます。最後のハード・リンクが除去されたあとに、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・システムにまたが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・リンクが提供するパス名をたどり、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。

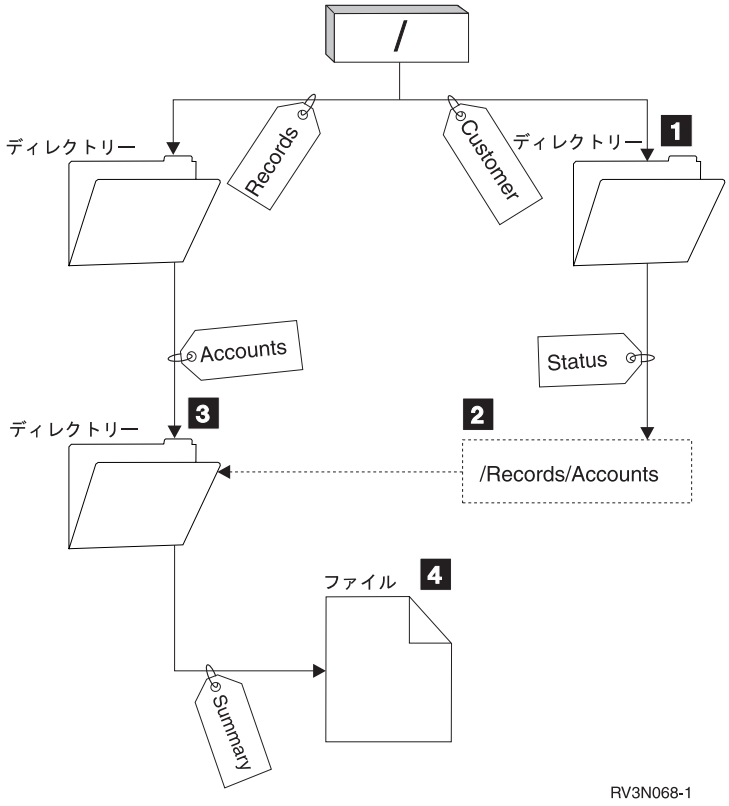


図 5. シンボリック・リンクの使用例

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 を持つシンボリック・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・タイプ *SYMLNK の) オブジェクトであるため、リンク先のオブジェクトがなくても存在することができます。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、シンボリック・リンクを使用します。

また、シンボリック・リンクは、複数のファイル・システムにまたが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ませんが、「ルート」(/) または QOpenSys ファイル・システム内にシンボリック・リンクを作成すれば、以下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。

13 ページの表 1 も参照してください。

パス名

パス名 (一部のシステムでは **pathname** と呼ばれる) は、オブジェクトを見つける方法をサーバーに指示します。パス名の形式は、ディレクトリー名のあとにオブジェクト名が続きます。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、スラッシュの代わりに円記号 (¥) を使用できます。

パス名を指定する方法には、次の 2 つがあります。

- **絶対パス名**は、最上位レベル、つまり「ルート」ディレクトリー (/ と示される) から始まります。たとえば、/ ディレクトリーから Smith というファイルへのパスを考えます。

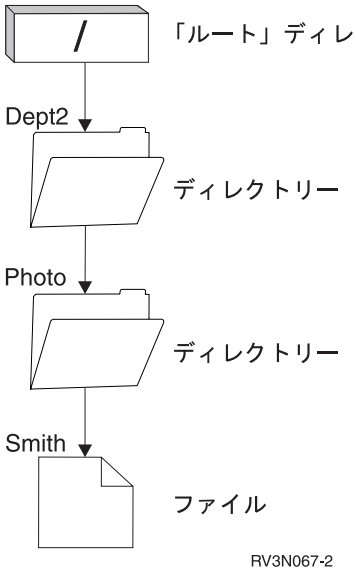


図 6. パス名の構成要素

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。

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に保管される文書は、ストリーム・ファイルです。 PC ファイルや UNIX システムのファイルもまた、ストリーム・ファイルの例です。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 *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 つまたは複数のフィールドに事前に区分されています。

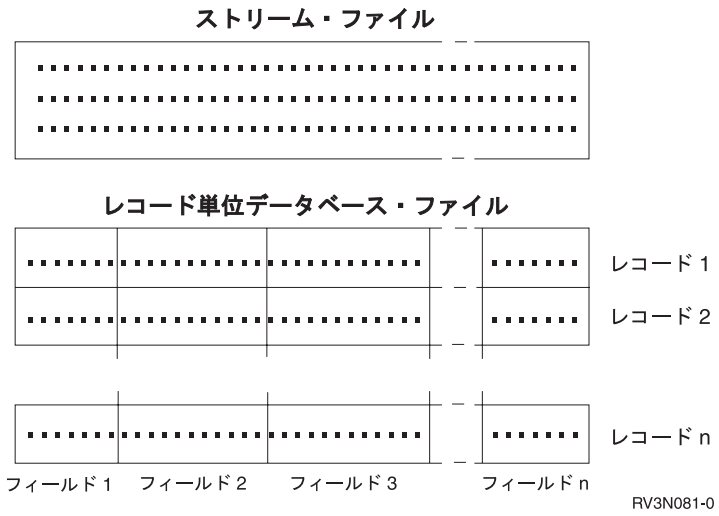


図 7. ストリーム・ファイルとレコード単位ファイルの比較

ストリーム・ファイルとレコード単位ファイルの構造は異なります。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、ファイルと対話するためにアプリケーションをどのように作成すべきか、それぞれのタイプのファイルをアプリケーションでどのように最適に使用でき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、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。

| 各ファイルは、2 つのフォーマット (*TYPE1 ストリーム・ファイルまたは *TYPE2 ストリーム・ファイル) のいずれかになります。ファイルが作成されたリリース、ユーザー定義ファイル・システムでファイルが作成されたかどうか、そのファイル・システムにどんな値が指定されたかによって、ファイル・フォーマットは異なります。

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。

- 101 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』
- 18 ページの『*TYPE1 のストリーム・ファイル』
- 18 ページの『*TYPE2 のストリーム・ファイル』

*TYPE1 のストリーム・ファイル

*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です。ファイルの最小サイズは 4096 バイトです。 *TYPE1 ストリーム・ファイルの最大オブジェクト・サイズは、約 256 ギガバイトです。

*TYPE2 のストリーム・ファイル

*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・アクセスが可能で、 OS/400 のバージョン 4 リリース 4 で新たに登場しました。 V5R3 では、「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムの場合、 *TYPE2 ストリーム・ファイルの最大オブジェクト・サイズが約 1 テラバイトになりました。それ以外の場合は、最大サイズは約 256 ギガバイトです。さらに、メモリー・マッピングが可能になり、属性の指定によって主記憶域割り振りを最適化できるようになりました。 V4R4 以降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 (ただし、ファイル・フォーマット *TYPE1 と指定されたユーザー定義ファイル・システムで作成されるファイルを除きます)。

注: 256 ギガバイトより大きいファイルは、 V5R3 より前のシステムでは保管または復元できません。

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ようにするシステム・サポートを利用できます。この場合、異なる文字エンコード・スキーム (コード・ページ) を使用する複数の iSeries サーバーおよび接続されたデバイスにまたがってファイル・システムを使用する場合にも、名前が維持されます。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る文字形式を、該当するコード・ページの適切な文字表記に変換します。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。

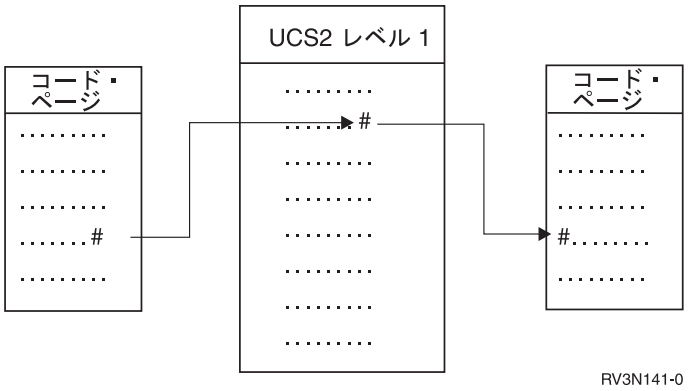


図 8. エンコード・スキーム間での同じ文字の保証

このサポートにより、さまざまなコード・ページを使用するデバイスが、サーバーと容易に対話することができます。たとえば、PC が iSeries サーバーとは異なるコード・ページを使用する場合でも、 PC ユーザーは同じファイル名でサーバーのファイルにアクセスできます。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実施されます。もちろん、デバイスが使用するコード・ページには、名前に使われている文字が含まれなければなりません。

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します。 EA は、それを表す名前、および値で構成されます。値は、テキスト、2 進データ、その他のタイプのデータです。

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、よく理解し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、その他の情報です。

.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した 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。

EA を設定できるかどうかは、ファイル・システムによって、およびオブジェクトによって異なります。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 の 3 つの事前定義 EA をサポートします。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」(/)、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・リンクに任意の種類の EA を付けることができます。また、EA が設定されないものが存在してもかまいません。

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、アプリケーションやユーザーが他の方法で EA にアクセスまたは変更することはできません。ただし、 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 2 つの CL コマンドは例外で、これらは拡張属性をユーザーに提示します。

また、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使用して、 QDLS の一部のオブジェクトに関連した EA を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、 46 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 48 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 PC が OS/2® または Windows を介して iSeries サーバーに接続されている場合、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 を照会および設定することができます。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」**を選択します)。

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。
- 名前の競合の可能性を最小限に抑えるために、 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
```

スキャンのサポート

iSeries には、統合ファイル・システム・オブジェクトをスキャンする機能が新しく追加されました。 iSeries の柔軟性を高めるこの新機能によって、ユーザーはさまざまな項目をスキャンし、スキャンをいつ行うかを決定し、スキャン結果に応じてどんなアクションを実行するかを決定できます。スキャンの詳細については、以下のトピックを参照してください。

- 21 ページの『関連するシステム値』
- 23 ページの『スキャンの実行理由』
- 『スキャンの例』

このサポートに関連して、以下の 2 つの出口点が新たに提供されます。

- QIBM_QP0L_SCAN_OPEN - オープン時の統合ファイル・システム・スキャン出口プログラム

  この出口点では、一定条件のもとで統合ファイル・システム・オブジェクトがオープンされるとき、オープン時の統合ファイル・システム・スキャン出口プログラムが呼び出されて、スキャン処理が実行されます。
- QIBM_QP0L_SCAN_CLOSE - クローズ時の統合ファイル・システム・スキャン出口プログラム

  この出口点では、一定条件のもとで統合ファイル・システム・オブジェクトがクローズされるとき、クローズ時の統合ファイル・システム・スキャン出口プログラムが呼び出されて、スキャン処理が実行されます。

オブジェクトをスキャン対象とするかどうか設定する方法については、 127 ページの『オブジェクトをスキャンするかどうかの設定』を参照してください。

注: スキャン対象となるオブジェクトは、 *TYPE2 ディレクトリーに完全に変換済みのファイル・システム内のオブジェクトだけです。

スキャンの例

以下の例は、出口プログラムによるスキャンの目的を示しています。

- ウィルス

  出口プログラムはウィルスをスキャンすることができます。ファイル内にウィルスが見つかった場合、アンチウィルス・プログラムは、問題の修復、ウィルスの隔離など、適切な処理を行うことができます。 iSeries 自体がウィルスに感染することは考えられないため、このスキャンの目的は、サーバー間でのウィルスの波及を抑止することです。
- ファイルがオープンされた時間を調べるための呼び出し

  ファイルがいつオープンされたかを検出するために、スキャンを使用することもできます。このようなスキャンによって、特定のファイルがアクセスされた日時を追跡することができます。特定のユーザーの振る舞いを追跡したい場合などに、これが役立ちます。

| システム値がどのように設定されているか、およびスキャン環境がどのように確立されたかに応じて、 2
| つの異なる時点でスキャンが実行されます。以下のリストは、2 つの異なる時点を示しています。

| 1. 実行時スキャン

| 実行時スキャンは、日常のアクティビティー中に 1 つまたは複数のファイルをスキャンすることです。
| これによって、ファイルにアクセスするたびにファイルの保全性が保障されます。日常のアクティビテ
| ィー中にスキャンすれば、スキャン基準に関してファイルが常に最新の状態に保たれます。

| **実行時にウィルスをスキャンする例**

| ある水曜日に、統合ファイル・システム上のファイルに PC からアクセスするとします。オープン出口
| プログラムが登録されていて、「ルート」(/)、QOpenSys、UDFS ファイル・システム内のファイルをス
| キャンするよう QSCANFS システム値が設定されているため、ファイルが PC からオープンされると
| きにスキャンされます。スキャンの結果、1 つのウィルスが検出され、アンチウィルス出口プログラム
| が問題の修復を行います。この出口プログラムがプログラムを修復したため、ファイルはもはやウィル
| スに感染しておらず、これにアクセスする PC が感染したり、ウィルスがさらに波及することはありま
| せん。

| ここで、アクセス時にウィルスをスキャンする代わりに、実行時スキャンを行わないよう設定したとし
| ます。感染したファイルに PC からアクセスすると、ウィルスはその PC に波及します。実行時スキャ
| ンを行うことによって、ウィルスが PC に広がる前に検出することができます。

| この方式の主な欠点は、スキャンを実行するためにリソース時間が必要であることです。ファイルにア
| クセスしようとするユーザーは、スキャンが完了した後で、そのファイルを使用することができます。
| システムは、すべてのアクセス時ではなく、必要な場合にのみスキャンを実行しようとします。詳しく
| は、 QIBM_QP0L_SCAN_OPEN および QIBM_QP0L_SCAN_CLOSE を参照してください。

| 2. 一括スキャンの手動設定

| 多数の項目を同時にスキャンしたい場合に、このオプションを使用します。この場合、サーバーがほと
| んど稼働しない週末にスキャンを実行するよう設定できます。こうすれば、日常のアクティビティーで
| ファイルにアクセスするとき、影響はほとんどありません。スキャンはオフラインで実行され、一括ス
| キャン後に変更されなかったファイルにアクセスするとき、再スキャンは必要ありません。このため、
| 実行時スキャンのオーバーヘッドが削減される可能性があります。

| 関連するシステム値

| この新しいスキャン機能に関連して、 2 つのシステム値が新たに追加されました。この 2 つのシステム値
| を使用して、ご使用のサーバーに適したスキャン環境を設定することができます。以下のリストには、2 つ
| のシステム値の名前と、それぞれに関する説明が示されています。これらのシステム値、および制御オプシ
| ョンについては、 iSeries ナビゲーターで説明されています。同等の文字ベースのインターフェース値が、
| iSeries ナビゲーター名の後の括弧内に示されます。たとえばシステム値 QSCANFSCTL の場合、 iSeries
| ナビゲーターの制御オプション「ファイル・サーバーのみを介したアクセスのスキャン (Scan accesses
| through file servers only)」を選択すると、文字ベースの制御オプション *FSVRONLY を指定した場合と同
| じ結果になります。

| 名前と説明は以下のとおりです。

| 1. ルート (/)、QOpenSys、およびユーザー定義ファイル・システムをスキャンするために、登録済み出口
| プログラムを使用する (QSCANFS)

| このシステム値を使用すれば、ファイル・システムをスキャンするかどうかを指定できます。「ルー
| ト」(/)、QOpenSys、およびユーザー定義ファイル・システム内のオブジェクトだけがスキャンされます

(ファイル・システムがすでに完全に変換済みの場合)。この値は、統合ファイル・システムのいずれかのスキャン関連出口点に登録された出口プログラムによって、オブジェクトがスキャンされるかどうかを指定します。

デフォルト値は、何らかの出口プログラムが登録済みの場合、オブジェクトがスキャンされます。

2. スキャン制御 (QSCANFSCTL)

このシステム値には、デフォルト制御オプションを使用するか、特定の制御オプションを指定することができます。 iSeries ナビゲーターのシステム値によって指定されるさまざまな制御オプションの概要については、以下を参照してください。

- ファイル・サーバーのみを介したアクセスのスキャン (Scan accesses through file servers only) - (*FSVRONLY を指定)

  ファイル・サーバーから iSeries にアクセスした場合に限り、スキャンが実行されます。このオプションを選択しない場合、すべてのアクセスがスキャン対象となります。

- 出口プログラムの障害時に要求が失敗 (Fail request if exit program fails) - (*ERRFAIL を指定)

  出口プログラムの呼び出し時にエラーが発生した場合、その出口プログラムを起動した要求または操作が失敗します。このオプションを選択しない場合、システムは障害が起きた出口プログラムをスキップして、オブジェクトはスキャンされなかったかのように扱われます。

- 書き込みアクセス・アップグレードの実行 (Perform write access upgrades) - (*NOWRTUPG を指定しない)

  出口プログラムに渡されるスキャン記述子に書き込みアクセスが含まれるようにするために、アクセス・アップグレードが実行されます。このオプションを選択しない場合、システムは書き込みアクセス・アップグレードを試行しません。

  *NOWRTUPG を指定した場合、出口プログラムに渡されるスキャン記述子に書き込みアクセスが含まれるようにするために、システムはアクセス・アップグレードを試行**しません**。 *NOWRTUPG が指定されない場合、システムは書き込みアクセス・アップグレードを試行します。

- 「オブジェクト変更時のみ」属性を使用してスキャンを制御する (Use 'only when objects have changed' attribute to control scan) - (*USEOCOATR を指定)

  「オブジェクト変更時のみ」属性 (オブジェクトが変更された場合にのみオブジェクトをスキャンする) が使用されます。このオプションを選択しない場合、この属性は使用されずオブジェクトは、変更された後およびスキャン・ソフトウェアが更新を示したときにスキャンされます。

- クローズ中にスキャンが失敗した場合、クローズ要求が失敗 (Fail close request if scan fails during close) - (*NOFAILCLO を指定しない)

  オブジェクトのクローズ処理中にスキャンが失敗した場合、クローズ要求が失敗します。このオプションを選択しない場合、クローズ要求は失敗しません。これを選択しない場合、この値は「出口プログラムの障害時に要求が失敗」の指定をオーバーライドします。

  *NOFAILCLO が指定されている場合、クローズ処理の一部であるオブジェクト・スキャンが失敗した場合でも、システムは、スキャン失敗の通知を出してクローズ要求を失敗**させません**。

- オブジェクト復元後の次のアクセスでスキャンを実行 (Scan on next access after object has been restored) - (*NOPOSTRST を指定しない)

オブジェクトの復元後、それに対してスキャンが行われます。「オブジェクトをスキャンしない (the object will not be scanned)」属性が指定されている場合、オブジェクトは復元後に一度だけスキャンされます。「オブジェクト変更時のみ (object change only)」属性が指定されている場合、オブジェクトは復元後にスキャンされます。

オブジェクトの復元中に *NOPOSTRST が指定された場合、オブジェクトが復元されても、スキャンは実行されません。オブジェクト属性が「オブジェクトをスキャンしない (the object will not be scanned)」である場合、そのオブジェクトは一度もスキャンされません。オブジェクト属性が「オブジェクト変更時のみ (object change only)」である場合、オブジェクトの復元後、変更されたときだけスキャンされます。

この 2 つのシステム値に関する詳細については、 QSCANFS または QSCANFSCTL を参照してください。

スキャンの実行理由

さまざまな理由で、スキャンを実行することができます。以下のリストは、どんなときに、どんな理由でスキャンを実行できるかを示しています。

- 『オブジェクトの変更』
- 『シグニチャーの変更』
- 24 ページの『異なる CCSID』
- 25 ページの『保管操作中に』
- 25 ページの『オブジェクト保全性の検査』

オブジェクトの現在のスキャン状況と属性を確認するには、オブジェクト・リンクの処理 (WRKLNK) コマンド、 属性の取得 (Qp0lGetAttr()) API、または iSeries ナビゲーターの「プロパティー」ページを使用できます。

オブジェクトの変更: オブジェクトの変更後にそのオブジェクトにアクセスしたとき、スキャンを実行することができます。通常、変更されるのはオブジェクトのデータ部分です。オブジェクトの変更の例には、オブジェクトへの直接書き込み、メモリー・マッピングによる書き込み、オブジェクトの切り捨て、オブジェクトの消去などがあります。オブジェクトの CCSID 属性が変更された場合もまた、次のアクセス時にスキャンが起動します。

シグニチャーの変更: オブジェクトにアクセスしたとき、グローバル・シグニチャーがそのオブジェクトのシグニチャーと異なる場合にスキャンを実行することができます。グローバルまたは独立 ASP グループのシグニチャーは、スキャン関連出口プログラムに関連付けられたソフトウェアのレベルを表します。詳しくは、 QIBM_QP0L_SCAN_OPEN を参照してください。オブジェクト・シグニチャーは、オブジェクトが最後にスキャンされたときのグローバルまたは独立 ASP シグニチャーを表します。オブジェクトが独立 ASP グループに含まれない場合、オブジェクト・シグニチャーはグローバル・スキャン・シグニチャーと比較されます。オブジェクトが独立 ASP に含まれる場合、オブジェクト・シグニチャーは関連する独立 ASP グループ・スキャン・シグニチャーと比較されます。

注: 以下の例では、語句スキャン・キーとスキャン・キー・シグニチャーが使用されます。スキャン・キーは、スキャン・ソフトウェアの 1 つのセットを識別する方法です。たとえば、特定の製造元のセットを指定します。スキャン・キー・シグニチャーを使用すれば、スキャン・ソフトウェアのセットが提供するサポート・レベルを識別することができます。たとえば、ウィルス定義のセットを識別します。詳しくは、QIBM_QP0L_SCAN_OPEN を参照してください。

以下の例では、オブジェクトが独立 ASP グループに含まれず、スキャンが実行されます。

1. QIBM_QP0L_SCAN_OPEN 出口点に出口プログラムが登録されています。スキャン・キーとスキャン・キー・シグニチャーが以下のように指定されるとします。

   スキャン・キー:　XXXXXX
   スキャン・キー・シグニチャー: 0000000000

   グローバル・スキャン・シグニチャーは 0000 で、更新されていません。
2. その後、 QIBM_QP0L_SCAN_CLOSE 出口点に 1 つの出口プログラムが登録されます。スキャン・キーとスキャン・キー・シグニチャーが以下のように指定されるとします。

   スキャン・キー:　XXXXXX
   スキャン・キー・シグニチャー: 1111111111

   その後、グローバル・スキャン・シグニチャーが 0001 に更新されます。
3. 次に、現在のオブジェクト・シグニチャーが 0000 であるファイルがオープンされます。出口プログラムが存在し、グローバル・スキャン・シグニチャーが (0000 から 0001 へ) 更新されているため、スキャンが開始されます。スキャンが正常に完了すると、ファイル・シグニチャーが 0001 に更新されます。
4. 別のユーザーによってファイルがオープンされた場合、オブジェクト・シグニチャーとグローバル・シグニチャーが一致するため、再スキャンされません。

以下の例では、出口プログラムが再スキャンを実行しようとします。

1. 新種のウィルスを対象とするスキャン・サポートがシステムに追加されました。スキャン・シグニチャーの変更 (QP0LCHSG) API が呼び出されて、スキャン・キーのスキャン・キー・シグニチャーが更新されます。スキャン・キーとスキャン・キー・シグニチャーが以下のように指定されるとします。

   スキャン・キー:　XXXXXX
   スキャン・キー・シグニチャー: 2222222222

   その後、グローバル・スキャン・キー・シグニチャーが 0002 に更新されます。
2. 以前にスキャンされたファイルがオープンされると、シグニチャーが異なるため、再スキャンが実行されます。

続いて、オブジェクトが独立 ASP グループに含まれる場合の例を示します。

1. 独立 ASP 内のファイルが初めてオープンされるとき、独立 ASP がオンになります。最初のファイルがオープンされるとき、独立 ASP のスキャン・キー・リストがシステムのスキャン・キー・リストと比較されます。独立 ASP のスキャン・キー・リストは存在しないため、両者は異なっています。この場合、独立 ASP のスキャン・キー・リストがグローバル・スキャン・キー・リストを取得します。その後、独立 ASP スキャン・キー・リストのスキャン・キーは XXXXXX に、スキャン・キー・シグニチャーは 2222222222 になります。その結果、独立 ASP のスキャン・シグニチャーが 0001 に変更されます。独立 ASP に含まれる、現在のオブジェクト・シグニチャーが 0000 であるファイルがオープンされると、独立 ASP のスキャン・シグニチャー 0001 と比較され、両者が異なるためにファイルがスキャンされます。正常にスキャンされると、ファイル・シグニチャーが 0001 に更新されます。

注: オブジェクトの属性が「オブジェクト変更時のみ (object change only)」で、かつ *USEOCOATR システム値が指定されている場合を除いて、シグニチャーの変更が原因でスキャンが起動されます。

異なる CCSID: 以前にスキャンされたときと異なる CCSID にオブジェクトが関連付けられている場合、スキャンを起動することができます。

| たとえば、CCSID 819 でデータが保管されたファイルが、 CCSID 1200 でオープンされ、正常にスキャンされたとします。ファイルのデータが変更されない限り、そのファイルを CCSID 1200 でオープンしても、スキャンは起動されません。しかし、そのファイルが別の CCSID (たとえば 37) でオープンされた場合、その CCSID 37 に関してスキャンが起動されます。そのスキャンもまた正常に実行された場合、CCSID 1200 および 37 を使用する後続のすべてのアクセスは、トリガーをさらに起動しません。

| システムに保管されるデータを最小化するために、 2 つの CCSID および 1 つのバイナリー標識のみが保持されます。多数の異なる CCSID を使って同じオブジェクトに頻繁にアクセスする場合には、これが原因で、多数のスキャンが実行される可能性があります。

| **保管操作中に:** これは、スキャンを実行するもう 1 つの例です。オブジェクトが保管されるときにスキャンを要求することができます。 SAV コマンドに新しく追加された SCAN パラメーターを使用して、ファイルの保管時にスキャンするかどうかを指定できます。さらに、以前にそのオブジェクトのスキャンが失敗した場合、または保管中にスキャンが失敗した場合に、オブジェクトを保管しないよう要求することができます。こうすれば、スキャンが失敗したファイルがメディアに書き込まれたり、他のシステムに伝送されるのを防ぐことができます。

| **注:** この場合、オブジェクトが復元されたときに、スキャン済みとマークされるわけではありません。オブジェクトが復元されるときには常に、スキャン状況の履歴全体が消去されます。

| **オブジェクト保全性の検査:** 最後に、オブジェクト保全性の検査 (CHKOBJITG) コマンドの SCANFS パラメーターの値が *YES に指定されている場合に、スキャンを要求することができます。ファイルをオープンせずに内容を判別したい場合には、この方法が便利です。 SCANFS (*Status) を指定した場合、以前にスキャンが失敗したすべてのオブジェクトに関して、スキャン障害違反がログに記録されます。

ファイル・システムの処理

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へのアクセスを提供します。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。構造と規則という観点から見ると、ライブラリーを介してデータベース・ファイルその他のさまざまなオブジェクト・タイプにアクセスする 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、フォルダー構造を介して文書 (実際にはストリーム・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・サポートが別々のファイル・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また、別個のファイル・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、 26 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」(/)
- オープン・システム・ファイル・システム (QOpenSys)
- ユーザー定義ファイル・システム (UDFS)
- ライブラリー・ファイル・システム (QSYS.LIB)

- 独立 ASP QSYS.LIB
- 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)
- NetWare ファイル・システム (QNetWare)
- iSeries NetClient ファイル・システム (QNTC)
- OS/400 ファイル・サーバー・ファイル・システム (QFileSvr.400)
- ネットワーク・ファイル・システム (NFS)

共通のインターフェースを使用すれば、任意のファイル・システムと対話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。提供されているコマンド、メニューと表示画面、およびアプリケーション・プログラム・インターフェース (API) は、この共通インターフェースを介してファイル・システムと対話することを可能にします。

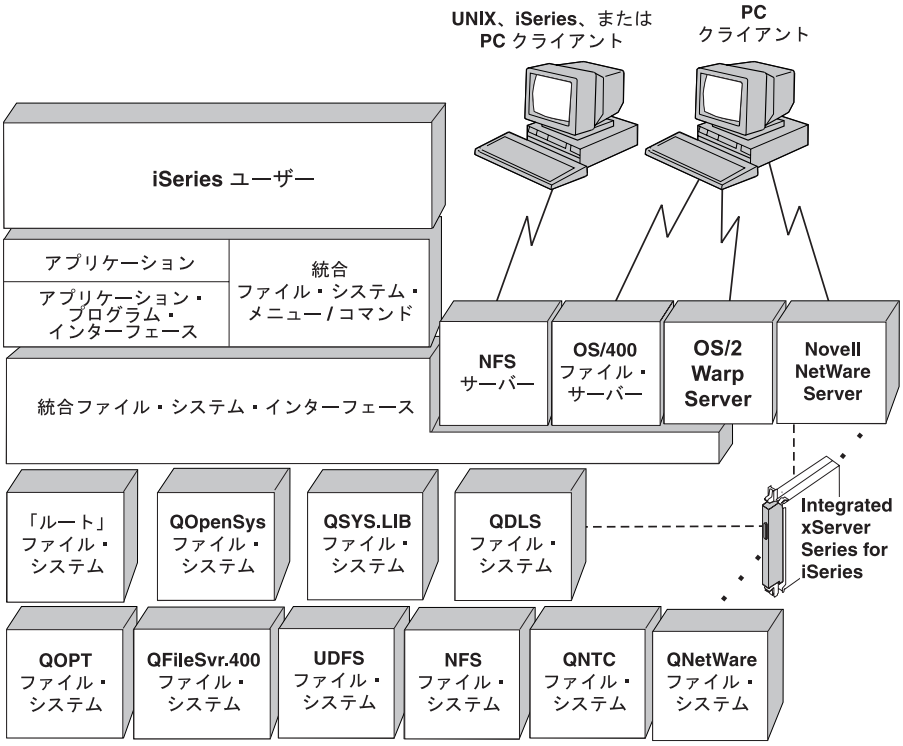


図 9. ファイル・システム、ファイル・サーバー、および統合ファイル・システム・インターフェース

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

ファイル・システムの比較

27 ページの表 2 と 29 ページの表 3 は、各ファイル・システムの機能と制限事項の要約です。

表 2. 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
ファイル・サイズの制限	T2=1 TB、T1=256 GB	T2=1 TB、T1=256 GB	データベース・ファイルのサイズ	4 GB	不定 [17]
OfficeVision® との統合 (たとえば、ファイルをメールで送信できる)	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 から 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	0 [18]
ファイル・システム内のディレクトリー階層の最大レベル	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
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 2. 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、やや遅くなります。
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。詳しくは、 40 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。詳しくは、 43 ページの『独立 ASP QSYS.LIB』を参照してください。
5. 詳しくは、 46 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。詳しくは、 40 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 から 3 文字です。詳しくは、 46 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 の 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. 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を操作する場合、統合ファイル・システム API はスレッド・セーフです。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に対してこれらの API が実行されると、複数のスレッドがジョブで実行されている場合には、 API が失敗します。
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します。 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、および 999,998 個のサブディレクトリーという制限があります。 *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・システムの両方を表します。
17. アクセス対象のシステムに応じて異なります。
18. QNTC は拡張属性をサポートしません。

略語

T1 = *TYPE1 *STMF

T2 = *TYPE2 *STMF

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

表 3. ¥ (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
ファイル・サイズの制限	4 GB	4 GB	T2=1 TB、T1=256 GB	不定 [16]	
OfficeVision との統合 (たとえば、ファイルをメールで送信できる)	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大レベル	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	Yes	Yes	Yes	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 3. ¥ (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す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. 詳しくは、 48 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. アクセス先のファイル・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、やや遅くなります。
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、マルチスレッド可能なプロセスでアクセスされるとき、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。
16. アクセス対象のシステムに応じて異なります。

略語

T1 = *TYPE1 *STMF

T2 = *TYPE2 *STMF

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

「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・サポートおよび階層ディレクトリー構造を利用しています。「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。
- 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。
- 統合ファイル・システムのスキャン関連出口点を使用して、オブジェクトのスキャンをサポートします。

「ルート」(/) ファイル・システムは、 /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・システムのデバイス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示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・システムにアクセスするには、 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 API を使用して、統合ファイル・システム・インターフェースを介します。

- 『「ルート」(/) 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・システムでのパス名』
- 『「ルート」(/) ファイル・システムでのリンク』
- 32 ページの『「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』
- 32 ページの『「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』
- 32 ページの『「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』
- 32 ページの『「ルート」(/) ファイル・システムでの UDP および TCP デバイス』

「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、オブジェクト名が入力されたときと同じ大文字と小文字を保持しますが、サーバーが名前を検索するときには、大文字と小文字が区別されません。

「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
```

- パス名の各構成要素の長さは、最大で 255 文字まで可能です (QSYS.LIB または QDLS ファイル・システムよりもかなり長くすることができます)。全パス名の最大長は非常に長く、最大 16 メガバイトまで可能です。
- 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 (18 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、 12 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、 67 ページの『CL コマンドを使用したアクセス』にリストしてあるすべてのコマンドと、 66 ページの『メニューおよび表示画面を使用したアクセス』に説明されている表示画面を使用できます。ただし、マルチスレッド可能プロセスでは、これらのコマンドの使用を避けた方が安全かもしれません。

「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、 106 ページの『API を使用した操作の実行』にリストされたすべての API を使用できます。スレッド・セーフ方式など、API に関する追加情報は、 API を参照してください。

「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・システムでは、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、詳しくは 95 ページの『オブジェクトのジャーナル処理』を参照してください。

「ルート」(/) ファイル・システムでの UDP および TCP デバイス: 「ルート」(/) ファイル・システムの /dev/xti ディレクトリーの下に、 udp および tcp の 2 つのデバイス・ドライバーが格納されるようになりました。この 2 つのドライバーはどちらも文字特殊ファイル (*CHRSF) で、初期プログラム・ロード (IPL) 時に作成されます。 udp および tcp デバイス・ドライバーは、 UDP および TCP トランスポート・プロバイダーへの接続をオープンするために使用されます。これらのドライバーはどちらもユーザー・デバイスになり、新しいデバイス・メジャー番号が割り当てられます。さらに、どちらもクローン・オープンが可能です。つまり、それぞれのオープン時にデバイスの固有のインスタンスが取得されます。これらのデバイスの使用は、 PASE (Portable Application Solutions Environment) でのみサポートされます。以下の表 4 は、作成されるオブジェクトと、それらのプロパティーを示しています。

表 4. デバイス・ドライバー・オブジェクトとプロパティー

パス名	タイプ	メジャー	マイナー	所有者	所有者のデータ権限	グループ	グループのデータ権限	パブリック (一般ユーザー) のデータ権限
/dev/xti	*DIR	該当せず	該当せず	QSYS	*RWX	なし	*RX	*RX
/dev/xti/tcp	*CHRSF	クローン	TCP	QSYS	*RW	なし	*RW	*RW
/dev/xti/udp	*CHRSF	クローン	UDP	QSYS	*RW	なし	*RW	*RW

オープン・システム・ファイル・システム (QOpenSys)

QOpenSys ファイル・システムには、 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。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。

さらに、

- UNIX システムと同様の階層ディレクトリー構造を介してアクセスされます。
- 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。
- 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。
- 統合ファイル・システムのスキャン関連出口点を使用して、オブジェクトのスキャンをサポートします。

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、唯一の違いは、 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を区別することです。

QOpenSys についての詳細は、 『QOpenSys の使用』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 API を使用して、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。

- 『QOpenSys ファイル・システムでの大文字小文字の区別』
- 『QOpenSys ファイル・システムでのパス名』
- 34 ページの『QOpenSys ファイル・システムでのリンク』
- 34 ページの『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』
- 34 ページの『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』
- 34 ページの『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、大文字と小文字が異なる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 (18 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、 12 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、 67 ページの『CL コマンドを使用したアクセス』にリストしてあるすべてのコマンドと、 66 ページの『メニューおよび表示画面を使用したアクセス』に説明されている表示画面を使用できます。ただし、マルチスレッド可能プロセスでは、これらのコマンドの使用を避けた方が安全かもしれません。

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: QOpenSys ファイル・システムでは、 106 ページの『API を使用した操作の実行』にリストされたすべての API を使用できます。スレッド・セーフ方式など、API に関する追加情報は、 API を参照してください。

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・システムでは、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、 95 ページの『オブジェクトのジャーナル処理』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任意の補助記憶域プール (ASP)、または独立補助記憶域プール (ASP) に常駐します。ユーザー自身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、このファイル・システムは、

- 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。
- 統合ファイル・システムのスキャン関連出口点を使用して、オブジェクトのスキャンをサポートします。

UDFS の作成に関する詳細は、 CRTUDFS を参照してください。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。さらに、UDFS の作成時に、以下のような追加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 内に格納され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によって、 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が判別されます。

| • オブジェクトスキャンの作成属性。 UDFS で作成されるオブジェクトに対するスキャン属性を定義します。この属性の設定については、 127 ページの『オブジェクトをスキャンするかどうかの設定』を参照してください。
| • 「名前変更およびリンク解除の制限」属性
| • UDFS の監査値
| • 異なるストリーム・ファイル・フォーマット (*TYPE1 と *TYPE2)。この 2 つのストリーム・ファイル・フォーマットについては、 17 ページの『ストリーム・ファイル』を参照してください。

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に、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 *FIFO 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・システム汎用コマンド、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、**マウント**および**アンマウント**の 2 つの状態でのみ存在します。 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能になります。

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。ディレクトリーに 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能になります。 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 に UDFS をマウントすると、 /home/JON/payroll ファイルはアクセス不能になり、 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 をアンマウントした後、 /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能になります。

注: 独立 ASP 上の UDFS 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 にアクセスするには、 OS/400 ファイル・サーバーを使用して、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・インターフェースを介します。統合ファイル・システム・インターフェースを使用する際、以下の考慮事項および制限事項に注意してください。

- 36 ページの『統合ファイル・システム UDFS での大文字小文字の区別』

- 『統合ファイル・システム UDFS でのパス名』
- 37 ページの『統合ファイル・システム UDFS でのリンク』
- 37 ページの『UDFS での統合ファイル・システム・コマンドの使用』
- 38 ページの『UDFS での統合ファイル・システム API の使用』
- 38 ページの『UDFS のグラフィカル・ユーザー・インターフェース』
- 38 ページの『統合ファイル・システム UDFS の作成』
- 38 ページの『統合ファイル・システム UDFS の削除』
- 38 ページの『統合ファイル・システム UDFS の表示』
- 38 ページの『統合ファイル・システム UDFS のマウント』
- 38 ページの『統合ファイル・システム UDFS のアンマウント』
- 39 ページの『統合ファイル・システム UDFS の保管および復元』
- 39 ページの『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』
- 39 ページの『UDFS と独立補助記憶域プール (ASP)』

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、サーバーは検索時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した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、1 つの 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 (18 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 では、同じオブジェクトに対する複数のハード・リンクを設定することができ、シンボリック・リンクが完全にサポートされます。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、12 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、67 ページの『CL コマンドを使用したアクセス』にリストしてあるすべてのコマンドと、 66 ページの『メニューおよび表示画面を使用したアクセス』に説明されている表示画面を使用できます。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 5. 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。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、ローカル・クライアント・ディレクトリーに入れる。
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。
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。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、ローカル・クライアント・ディレクトリーに入れる。このコマンドは、ADDMFS コマンドの別名です。
RMVMFS	マウント・ファイル・システムの除去。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、ローカル・クライアント・ネーム・スペースから除去する。
UNMOUNT	ファイル・システムのアンマウント。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、ローカル・クライアント・ネーム・スペースから除去する。このコマンドは、RMVMFS コマンドの別名です。

注: UDFS 上に保管されたオブジェクトに対して統合ファイル・システム・コマンドを実行する前に、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、 106 ページの『API を使用した操作の実行』にリストされたすべての API を使用できます。

注: UDFS 上に保管されたオブジェクトに対して統合ファイル・システム API を実行する前に、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。基本的なタスクには、次のものが含まれます。

- 『統合ファイル・システム UDFS の作成』
- 『統合ファイル・システム UDFS のマウント』
- 『統合ファイル・システム UDFS 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。任意の ASP または独立 ASP 内に UDFS を作成することができます。

さらに、UDFS に関して以下の項目を指定できます。

- 大文字小文字の区別
- UDFS 内に作成されるオブジェクトをスキャンするかどうか
- UDFS 内に作成されるオブジェクトの監査値
- 制限された名前変更およびリンク解除の属性値

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済みの UDFS と、その中のすべてのオブジェクトを削除します。 UDFS がマウントされている場合、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、およびマウントされているかどうかを表示します。マウント・ファイル・システムの情報の表示 (DSPMFSINF) コマンドもまた、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・システム内のオブジェクトを、統合ファイル・システムのネーム・スペースからアクセスできるようにします。 UDFS をマウントするには、 ADDMFS コマンドの 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 UDFS 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能にします。いったん UDFS がアンマウントされると、 UDFS 内のオブジェクトに個別にアクセスできなくなります。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能にします。コマンド使用時にファイル・システム内のいずれかのオブジェクト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になります。 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、/dev/qasp02/jenn.udfs という 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能になります。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 UDFS 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、復元することができます。保管コマンド (SAV) によって UDFS 内のオブジェクトを保管できます。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 内のオブジェクトだけでなく、 UDFS 属性を正しく保管するには、UDFS をアンマウントしてください。

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・システム内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、 95 ページの『オブジェクトのジャーナル処理』を参照してください。

UDFS と独立補助記憶域プール (ASP): 独立 ASP をオンにすると、「ルート」(/) ファイル・システム内部が以下のように変更されます。

- /dev ディレクトリーの中に、独立 ASP 用のディレクトリーが作成されます。このディレクトリーの名前は、その ASP に関連した装置記述の名前に一致します。オンに変更する要求が出される前にこのディレクトリーがすでに存在する場合、ディレクトリーが空でなければ、変更は実行されますが、 ASP 上で UDFS の操作を行うことはできません。この場合、独立 ASP をオフに変更した後、ディレクトリーを名前変更するか、ディレクトリーの内容を削除して、オンに変更する要求を再び試行してください。
- /dev/asp_name ディレクトリー内には、独立 ASP に存在するすべての UDFS に関連したブロック特殊ファイル・オブジェクトがあります。システム提供のデフォルト UDFS が必ず 1 つ存在します。デフォルト UDFS のブロック特殊ファイルのパスは、 /dev/asp_name/QDEFAULT.UDFS です。
- デフォルト UDFS は、 /asp_name ディレクトリーに上書きマウントされます。オンに変更する要求の前に、 /asp_name ディレクトリーがすでに存在する必要はありません。ただし、すでに存在する場合、ディレクトリーの中身は空でなければなりません。空でない場合、ASP はオンに変更されますが、デフォルト UDFS はマウントされません。このような場合には、ディレクトリーを名前変更するか、ディレクトリーの内容を削除します。その後、オフに変更して再びオンに変更するか、 MOUNT コマンドを使用してデフォルト UDFS をマウントします。
- 独立 ASP が 1 次または 2 次 ASP である場合、デフォルト UDFS が正常にマウントされると、 1 つの追加のファイル・システムがマウントされます。独立 ASP の QSYS.LIB ファイル・システムが /asp_name/QSYS.LIB の上に上書きマウントされます。詳しくは、 43 ページの『独立 ASP QSYS.LIB』を参照してください。

注: デフォルト UDFS と別個に、このファイル・システムをマウントまたはアンマウントすることはできません。このファイル・システムは常に自動的にマウントまたはアンマウントされます。

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・ファイルへのアクセスを提供するとともに、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に対する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、 DDS などの機能を使用してアプリケーションを開発するプログラマーは、 QSYS.LIB ファイル・システムを使用しました。コマンド、メニュー、および表示画面を使って出力待ち行列を操作するシステム・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うシステム管理者もまた、 QSYS.LIB ファイル・システムを使用しました。

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・インターフェースを介して QSYS.LIB にアクセスできません。

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・システムにアクセスするには、 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 API を使用して、統合ファイル・システム・インターフェースを介します。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、次の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』
- 41 ページの『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』
- 41 ページの『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』
- 41 ページの『QSYS.LIB ファイル・システムでの保管ファイルのサポート』
- 41 ページの『QSYS.LIB ファイル・システムでの大文字小文字の区別』
- 42 ページの『QSYS.LIB ファイル・システムでのパス名』
- 42 ページの『QSYS.LIB ファイル・システムでのリンク』
- 42 ページの『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』
- 43 ページの『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』

QSYS.LIB ファイル・システムの QPWFSERVER 権限リスト: QPWFSERVER は権限リスト (オブジェクト・タイプ *AUTL) です。これは、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・システム内のすべてのオブジェクトに関して、追加のアクセス要件を提供します。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、一般ユーザーがリモート・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の他に、外部記述の物理ファイルがあります。
- 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 つのジョブにのみ与えられます。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・オブジェクトに対する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、保管ファイル・オブジェクトに対する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とき、そのファイルの他のオープン・インスタンスは許可されません。保管ファイルでは、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**。ただし、ファイルの複数のインスタンスを読み取り用にオープンするようなジョブが存在しない場合に限ります。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、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・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。大文字と小文字のどちらでオブジェクト名を検索しても、結果は同じです。

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、引用符で囲んで名前を検索する場合、引用符で囲まれた文字の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、固有のオブジェクトを識別するためには、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、または保管できません。

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じです。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は可能です。

リンクについては、 12 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: QSYS.LIB ファイル・システムでは、 67 ページの『CL コマンドを使用したアクセス』にリストされているコマンドを使用できます。ただし、以下を除きます。

- ADDLNK コマンドは、 QSYS.LIB のオブジェクトへの シンボリック・リンクを作成する場合にのみ使用できます。
- 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に対して使用できません。

66 ページの『メニューおよび表示画面を使用したアクセス』で説明されているのと同じ制限が、ユーザー表示画面に適用されます。

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: QSYS.LIB ファイル・システムでは、 106 ページの『API を使用した操作の実行』にリストされている API を使用できます。ただし、以下を除きます。

- 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 のオブジェクトへのリンクを設定する場合にのみ使用できます。
- データベース物理ファイルに対しては、 QjoStartJournal() および QjoEndJournal() API を使用できません。

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・ファイルへのアクセスを提供するとともに、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、すべてのユーザー・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に対する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・システムに対するアクセスは、 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 を使用して、統合ファイル・システム・インターフェースを介して行われます。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、次の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』
- 44 ページの『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』
- 44 ページの『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』
- 44 ページの『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』
- 44 ページの『独立 ASP QSYS.LIB ファイル・システムでの大文字小文字の区別』
- 45 ページの『独立 ASP QSYS.LIB ファイル・システムでのパス名』
- 45 ページの『独立 ASP QSYS.LIB ファイル・システムでのリンク』
- 45 ページの『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』
- 46 ページの『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』

独立 ASP QSYS.LIB ファイル・システムの QPWFSERVER 権限リスト: QPWFSERVER は権限リスト (オブジェクト・タイプ *AUTL) です。これは、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・システム内のすべてのオブジェクトに関して、追加のアクセス要件を提供します。この権限リストで指定された権限は、独立 ASP 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、一般ユーザーがリモート・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の他に、外部記述の物理ファイルがあります。
- 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 つのジョブにのみ与えられます。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・オブジェクトに対する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、保管ファイル・オブジェクトに対する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とき、そのファイルの他のオープン・インスタンスは許可されません。保管ファイルでは、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**。ただし、ファイルの複数のインスタンスを読み取り用にオープンするようなジョブが存在しない場合に限ります。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、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・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。大文字と小文字のどちらでオブジェクト名を検索しても、結果は同じです。

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、引用符で囲んで名前を検索する場合、引用符で囲まれた文字の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、固有のオブジェクトを識別するためには、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、または保管できません。

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じです。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、 12 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:
独立 ASP QSYS.LIB ファイル・システムでは、 67 ページの『CL コマンドを使用したアクセス』にリストされているコマンドを使用できます。ただし、以下を除きます。

- ADDLNK コマンドは、独立 ASP QSYS.LIB のオブジェクトへの シンボリック・リンクを作成する場合にのみ使用できます。
- 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に対して使用できません。
- MOV コマンドを使用して、独立 ASP QSYS.LIB ファイル・システム内のライブラリーを基本補助記憶域プール (ASP) に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 パラメーターで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 ではサポートされません。

66 ページの『メニューおよび表示画面を使用したアクセス』で説明されているのと同じ制限が、ユーザー表示画面に適用されます。

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: 独立 ASP QSYS.LIB ファイル・システムでは、 106 ページの『API を使用した操作の実行』にリストされている API を使用できます。ただし、以下を除きます。

- 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 のオブジェクトへのリンクを設定する場合にのみ使用できます。
- データベース物理ファイルに対しては、 QjoStartJournal() および QjoEndJournal() API を使用できません。

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、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・システムにアクセスするには、 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 API を使用して、統合ファイル・システム・インターフェースを介します。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、次の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・システムの詳細については、以下のトピックを参照してください。

- 『QDLS ファイル・システムでの統合ファイル・システムおよび HFS』
- 47 ページの『QDLS ファイル・システムでのユーザー登録』
- 47 ページの『QDLS ファイル・システムでの大文字小文字の区別』
- 47 ページの『QDLS ファイル・システムでのパス名』
- 47 ページの『QDLS ファイル・システムでのリンク』
- 48 ページの『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』
- 48 ページの『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』

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を操作するには、文書ライブラリー・オブジェクト (DLO) CL コマンドだけでなく、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 を使用することができます。統合ファイル・システムが統合言語環境 (ILE) 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 を使用すると、ディレクトリー拡張属性 (*ディレクトリー項目属性* ともいう) にアクセスしてこれを変更することができます。 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ことに注意してください。

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、以下のように、1 つの名前だけで構成されるか、

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

  あるいは、以下の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、または保管できません。

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リンクが設定されているかのように扱います。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は可能です。

リンクについては、 12 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: QDLS ファイル・システムでは、 67 ページの『CL コマンドを使用したアクセス』にリストされているコマンドを使用できます。ただし、以下を除きます。

- 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 のオブジェクトへのリンクを設定する場合にのみ使用できます。
- CHKIN および CHKOUT コマンドは、ファイルに対して使用できますが、ディレクトリーに対しては使用できません。
- APYJRNCHG、CHGJRNOBJ、ENDJRN、SNDJRNE、および STRJRN コマンドはサポートされていません。

66 ページの『メニューおよび表示画面を使用したアクセス』で説明されているのと同じ制限が、ユーザー表示画面に適用されます。

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: QDLS ファイル・システムでは、 106 ページの『API を使用した操作の実行』にリストされている API を使用できます。ただし、以下を除きます。

- 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 のオブジェクトへのリンクを設定する場合にのみ使用できます。
- 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・システムは、光メディアに保管されたストリーム・データへのアクセスを提供します。

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・システムを介した QOPT の使用

QOPT ファイル・システムへのアクセスは、 PC 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・システムを介して行ないます。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、次の考慮事項および制限事項に注意してください。

QOPT ファイル・システムの詳細については、以下のトピッ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・システムおよび HFS』
- 『QOPT ファイル・システムでの大文字小文字の区別』
- 『QOPT ファイル・システムでのパス名』
- 50 ページの『QOPT ファイル・システムでのリンク』
- 50 ページの『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』
- 50 ページの『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』

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を操作するには、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 を使用できます。統合ファイル・システムが統合言語環境 (ILE) 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 を使用すれば、ディレクトリー拡張属性 (*ディレクトリー項目属性* ともいう) にアクセスしてこれを変更したり、保留されている光ファイルを処理することができます。 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ことに注意してください。

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: QOPT 内にファイルまたはディレクトリーを作成するとき、光メディアのフォーマットに応じて、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は区別されません。

QOPT ファイル・システムでのパス名:

- パス名はスラッシュ (/) 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 /QOPT を指定するか、パス名の中に 1 つまたは複数のディレクトリーや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' から 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、または保管できません。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できます。

リンクについては、 12 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: QOPT ファイル・システムでは、 67 ページの『CL コマンドを使用したアクセス』にリストされているほとんどのコマンドを使用できます。ただし、QOPT ファイル・システムにはいくつかの例外があります。マルチスレッド可能なプロセスでこれらの CL コマンドを使用することは、安全でない可能性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。 66 ページの『メニューおよび表示画面を使用したアクセス』で説明されているのと同じ制限が、ユーザー表示画面に適用されます。

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
- CHGJRNOBJ
- 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、 106 ページの『API を使用した操作の実行』にリストされているすべての API をスレッド・セーフ方式で使用できます。ただし、以下を除きます。

- 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 の統合 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、 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 が終わると、 /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して表示されます。

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することができます。これによって、アクセスがより容易になり、/QNetWare ディレクトリーよりもパフォーマンスが向上します。さらに、NetWare ファイル・システムをマウントすれば、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できます。

NetWare ファイル・システムをマウントするには、 NDS パスを使用するか、 NetWare パスを SERVER/VOLUME:directory/directory の形式で指定します。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、次のような別個の複数のファイル・システムを表します。

- この構造は、以下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 サーバーを表します。
- 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 のいずれかを介して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、以下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 は NDS ツリーを表します。 .C は国、.O は組織、.OU は組織単位、 .CN は共通名をそれぞれ表します。 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・システムにアクセスするには、 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 API を使用して、統合ファイル・システム・インターフェースを介します。次のような考慮事項、制限事項、および相互関係があります。

QNetWare ファイル・システムの詳細については、以下を参照してください。

- 『QNetWare ファイル・システムでの権限および所有権』
- 『QNetWare ファイル・システムでの監査』
- 『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・システムでの NDS オブジェクト』
- 『QNetWare ファイル・システムでのリンク』
- 53 ページの『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』
- 53 ページの『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』

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 を使って所有者またはユーザーの権限を検索または設定するとき、 QNetWare はユーザーの名前に基づいて NetWare ユーザーを iSeries サーバー・ユーザーにマップします。 NetWare 名が 10 文字を超える場合、または対応する iSeries サーバー・ユーザーが存在しない場合には、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。 WRKAUT コマンドおよび CHGAUT コマンドを使用して、ユーザーの権限を表示および変更することができます。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 はファイルおよびディレクトリーの監査をサポートしますが、 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やディレクトリーの大文字小文字の区別を保持しません。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。さらに、Novell NetWare は、DOS、 OS/2、 Apple Macintosh、および NFS など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 ボリュームで必要とされるため、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、または保管できません。ただし、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、シンボリック・リンク作成することができます。

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: QNetWare ファイル・システムでは、 67 ページの『CL コマンドを使用したアクセス』にリストされているコマンドを使用できます。ただし、以下を除きます。

ADDLINK

APYJRNCHG

CHGAUD

| CHGJRNOBJ

CHGPGP

CHKIN

CHKOUT

ENDJRN

SNDJRNE

STRJRN

WRKOBJOWN

WRKOBJPGP

上記のコマンドに加えて、次のコマンドを 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はできません。

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: QNetWare ファイル・システムでは、 106 ページの『API を使用した操作の実行』にリストされている API を使用できます。ただし、以下の API を除きます。

givedescriptor()

link()

QjoEndJournal()

QjoRetrieveJournalInformation()

QJORJIDI()

QJOSJRNE()

QjoStartJournal()

readlink()

symlink()

takedescriptor()

上記の API に加え、次の API を 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はできません。

chmod()

chown()

create()

fchmod()

fchown()

fcntl()

ftruncate()

lseek()

mkdir()

read()

readv()

unmask()

write()

writev()

iSeries NetClient ファイル・システム (QNTC)

QNTC ファイル・システムは、 Windows NT 4.0 以降のサーバーまたはスタンドアロン・サーバーを実行する、ローカルまたはリモートの iSeries の統合 xSeries サーバーに保管されている、データや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、 iSeries サーバー・アプリケーションは、 Windows クライアントと同じデータを使用できます。

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 にアクセスするために、 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) が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

iSeries NetServer、 iSeries ナビゲーター、統合ファイル・システムのコマンド、ユーザー表示画面、または API を使用して、統合ファイル・システム・インターフェースを介して QNTC ファイル・システムにアクセスできます。次のような考慮事項および制限事項に注意してください。

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

- 『QNTC ファイル・システムでの権限および所有権』
- 55 ページの『QNTC ファイル・システムでの大文字小文字の区別』
- 55 ページの『QNTC ファイル・システムでのパス名』
- 55 ページの『QNTC ファイル・システムでのリンク』
- 55 ページの『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』
- 56 ページの『QNTC ファイル・システムでの MKDIR コマンドの使用』
- 56 ページの『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』
- 57 ページの『QNTC ファイル・システムでの Kerberos の使用可能化』

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します。 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 コマンドおよび 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、入力されたままの状態で保持しますが、名前の大文字と小文字を区別しません。大文字と小文字のどちらでオブジェクト名を検索しても、結果は同じです。

QNTC ファイル・システムでのパス名:

- パス名はスラッシュで始まり、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.)
```

- サーバー名の長さは最大 15 文字までが可能で、パスに含まれなければなりません。
- 共用名の長さは最大 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 レベルの階層を使用できます。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・サブネットで機能しているそれぞれの Windows NT サーバーは、 /QNTC の下のディレクトリーとして自動的に表示されます。ローカル・サブネットの外側に Windows ベースのサーバーを追加するには、ディレクトリーの作成 (MKDIR) コマンド (67 ページの表 7 を参照)、または mkdir() API (87 ページの『API を使用したアクセス』を参照) を使用します。

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 では、シンボリック・リンクを作成または保管できません。「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、 12 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: QNTC ファイル・システムでは、 67 ページの『CL コマンドを使用したアクセス』にリストされているコマンドを使用できます。ただし、以下を除きます。

ADDLNK

APYJRNCHG

| CHGJRNOBJ

CHGOWN

CHGAUT

CHGPGP

CHKIN

CHKOUT

DSPAUT

ENDJRN

| RST (統合 xSeries® サーバーで使用可能)

| SAV (統合 xSeries サーバーで使用可能)
SNDJRNE
STRJRN
WRKAUT
WRKOBJOWN
WRKOBJPGP

66 ページの『メニューおよび表示画面を使用したアクセス』で説明されているのと同じ制限が、ユーザー表示画面に適用されます。

QNTC ファイル・システムでの MKDIR コマンドの使用: 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には、ディレクトリーの作成 (MKDIR) コマンドを使用します。 iSeries NetServer のドメイン内およびローカル・サブネット内で機能しているすべての Windows サーバー用に、 QNTC ディレクトリーが自動的に作成されます。ローカル・サブネット外、または iSeries NetServer ドメイン外の Windows ベースのサーバーを追加するには、 MKDIR コマンドまたは mkdir() API を使用する必要があります。次はその一例です。

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

上記により、 NTSRV1 サーバーが QNTC ファイル・システム・ディレクトリー構造に追加され、そのサーバー上のファイルとディレクトリーにアクセスできるようになります。

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'
```

上記により、サーバーが QNTC ファイル・システム・ディレクトリー構造に追加されます。

注: iSeries NetServer for WINS を構成することによって、サブネット外のサーバー用のディレクトリーを自動的に作成できます。

注: ディレクトリー構造へのディレクトリーの追加に mkdir() API または MKDIR CL コマンドを使用した場合、これらのディレクトリーは IPL を行うと表示されなくなります。それぞれのシステム IPL 後に、 MKDIR コマンドまたは mkdir() API を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: QNTC ファイル・システムでは、106 ページの『API を使用した操作の実行』にリストされている API を使用できます。ただし、以下を除きます。

- chmod()、fchmod()、utime()、および umask() 関数は、 QNTC のオブジェクトに対して効力がありませんが、使用してもエラーは発生しません。
- QNTC ファイル・システムは、次の関数をサポートしません。

  chown()
  fchown()
  givedescriptor()
  link()
  QjoEndJournal()
  QjoRetrieveJournalInformation()
  QJORJIDI()

QJOSJRNE()

QjoStartJournal()

Qp0lGetPathFromFileID()

readlink()

symlink()

takedescriptor()

QNTC ファイル・システムでの Kerberos の使用可能化: QNTC では、 Kerberos V5 認証プロトコルをサポートする Windows ベースのサーバーに iSeries からアクセスすることができます。 LAN 管理機能タイプのパスワードを使ってサーバーを相互に認証する代わりに、 iSeries を適切に構成すれば、単一のログオン・トランザクションによって、サポートされる CIFS サーバーにアクセスできるようになりました。

Kerberos を QNTC ファイル・システムで使用可能にするには、まず以下を構成する必要があります。

- ネットワーク認証サービス
- エンタープライズ識別マッピング (EIM)

上記の項目を構成した後、 QNTC ファイル・システムで Kerberos を使用可能にすることができるようになります。ユーザーが QNTC の Kerberos サポートを利用できるようにするには、以下のようなステップを実行する必要があります。

- ユーザーの iSeries ユーザー・プロファイルで、ローカル・パスワード管理パラメーター LCLPWDMGT を ***NO** に設定する必要があります。 *NO と指定することで、ユーザーはシステムへのパスワードを持たなくなり、 5250 セッションにサインオンできなくなります。システムにアクセスする唯一の手段は、 Kerberos を使用可能なアプリケーション (たとえば、 iSeries ナビゲーター) です。

  *YES を指定した場合、パスワードはシステムによって管理され、ユーザーは Kerberos を使用せずに認証されます。
- Kerberos チケット、および iSeries ナビゲーター接続が必要です。
- ご使用の iSeries 用の Kerberos チケットは、転送可能 (forwardable) でなければなりません。チケットを転送可能にするには、以下のステップを実行してください。
  - Kerberos レルムの KDC 上の「Active Directory Users and Computers」ツールにアクセスします。
  - ユーザーを選択します。
  - サービス・プリンシパル名に対応する名前を選択します。
  - 「プロパティー (Properties)」を選択します。
  - 「アカウント (Account)」タブを選択します。
  - 「アカウント (Account)」オプションで、「アカウントは委任用に信頼される (Account is trusted for delegation)」を選択します。

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・システムは、リモート iSeries サーバーに常駐する他のファイル・システムへの透過的なアクセスを提供します。このファイル・システムには、階層ディレクトリー構造を介してアクセスします。

QFileSvr.400 ファイル・システムは、ユーザーの代わりにファイル要求を実行するクライアントのようなものと考えることができます。 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 の詳細については、『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・システムにアクセスするには、 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 API を使用して、統合ファイル・システム・インターフェースを介します。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、次の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・サーバー上のアクセス対象のファイル・システムの特性によって決まります。

QFileSvr.400 ファイル・システムの詳細について、以下のトピックを参照してください。

- 『OS/400 ファイル・サーバー ファイル・システムでの大文字小文字の区別』
- 『OS/400 ファイル・サーバー ファイル・システムでのパス名』
- 59 ページの『OS/400 ファイル・サーバー ファイル・システムでの通信』
- 60 ページの『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』
- 60 ページの『OS/400 ファイル・サーバー ファイル・システムでのリンク』
- 60 ページの『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』
- 61 ページの『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』

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: 第 1 レベル・ディレクトリーは、実際にはターゲット・システムの「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大文字小文字がそのまま保持されます。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対象の特定のファイル・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文字小文字を保持します。

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、次のいずれかになります。

    - 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、指定された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・システムによっては、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短く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、およびアクセス対象のファイル・システムの制限以外に制限はありません。
- 名前に使用される文字は、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 (18 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にのみ確立可能です。
- SNA LU 6.2 接続は、使用中でないローカル制御セッション (たとえば、 LU 6.2 接続用に特別に確立されたセッション) がある場合にのみ試行されます。 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・サーバーとの接続を確立します。
  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・マッパーに接続します。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表にある場合、サーバー・マッパーは 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) に検査を行い、使用中でない (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) 接続が存在するかどうか、それらの接続が 2 時間以内に何も活動しなかったかどうかを判別します。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。このエラーは、タイムアウトが発生したことを示します。

  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。 QFileSvr.400 がリモート通信システムに正常に接続するには、モードを開始してセッションを確立する必要があります。

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 に認証された場合、 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 による認証が失敗した場合、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。

- Kerberos を使って認証しない場合、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必要です。

  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、または保管できません。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、 12 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: QFileSvr.400 ファイル・システムでは、 67 ページの『CL コマンドを使用したアクセス』にリストされているコマンドを使用できます。ただし、以下を除きます。

ADDLNK

APYJRNCHG

CHGAUT

| CHGJRNOBJ

CHGOWN

DSPAUT

ENDJRN

RST

SAV

SNDJRNE

STRJRN

WRKOBJOWN

WRKOBJPGP

66 ページの『メニューおよび表示画面を使用したアクセス』で説明されているのと同じ制限が、ユーザー表示画面に適用されます。

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: QFileSvr.400 ファイル・システムでは、 106 ページの『API を使用した操作の実行』にリストされている API を使用できます。ただし、以下を除きます。

chown()

fchown()

| fclear()

| fclear64()

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、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ユーザーに提供します。 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た後、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・システムには、マウント元のリモート・サーバーのディレクトリーまたはファイル・システムの機能、特性、制限、および

相互関係が適用されます。マウント・ファイル・システムに対する操作はローカルには実行されません。要求のフローは接続を介してサーバーへ送られ、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・システムには、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でき、次の考慮事項および制限があります。

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

- 『ネットワーク・ファイル・システムの特性』
- 『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』
- 63 ページの『ネットワーク・ファイル・システムでのリンク』
- 63 ページの『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』
- 64 ページの『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ます。ローカル・ディレクトリーまたはローカル・ファイル・システムのように見えるものに対する要求は、実際には、NFS 接続を介してサーバー上で操作していることに注意してください。

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に、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えてください。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、動作します。

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、あるいはサーバー接続を介して処理するために重要です。コマンドがローカル・レベルで正しく動作したからといって、サーバーからマウントされたディレクトリー上で正しく作動するとは限りません。クライアント上に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: クライアント/サーバー接続には主に以下の 3 つがあり、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します。

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 のサーバーから、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、 iSeries サーバーで動作するのと似た方法でクライアント上で動作します。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適用されます。たとえば、QDLS ファイル・システムでは、パス名の構成要素は、8 文字に拡

張子 3 文字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適用されます。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 サーバー QOpenSys ファイル・システムに非常に似た動作をすることに注意してください。 QOpenSys ファイル・システムの詳細については、 32 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・システムの資料を再確認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・リンクを使用して、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のリンクを設定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・リンクを使用できるかどうかは、 NFS にマウントされるファイル・システムに完全に依存します。

リンクについては、 12 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、 67 ページの『CL コマンドを使用したアクセス』にリストしてあるすべてのコマンドと、 66 ページの『メニューおよび表示画面を使用したアクセス』に説明されている表示画面を使用できます。ただし、以下を除きます。

- APYJRNCHG
- CHGJRNOBJ
- 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。ただし、マルチスレッド可能プロセスでは、これらのコマンドの使用を避けた方が安全かもしれません。次の表に、これらのコマンドを説明します。ネットワーク・ファイル・システムに固有のコマンドおよび表示画面の詳細な説明については、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 6. 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。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、ローカル・クライアント・ディレクトリーに入れる。
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表に、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。
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表に、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。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、ローカル・クライアント・ディレクトリーに入れる。このコマンドは、ADDMFS コマンドの別名です。
RLSIFSLCK	統合ファイル・システムのロック解除。クライアントによって保持された、またはオブジェクトに対して保持された、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。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、ローカル・クライアント・ネーム・スペースから除去する。
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。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、ローカル・クライアント・ネーム・スペースから除去する。このコマンドは、RMVMFS コマンドの別名です。

注: ネットワーク・ファイル・システムに対してコマンドを使用する前に、ネットワーク・ファイル・システムがマウントされている必要があります。

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、 106 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 API を使用できます。ただし、以下を除きます。

- 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 言語関数の詳細説明については、 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・システムに対して API を使用する前に、ネットワーク・ファイル・システムがマウントされている必要があります。

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、および文書の処理に使用するすべてのユーザー・インターフェース (メニュー、コマンド、表示画面など) は、統合ファイル・システムの導入前と同じように操作可能です。ただし、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これらのインターフェースを使用することはできません。

統合ファイル・システムには、これとは別のユーザー・インターフェースのセットが提供されています。統合ファイル・システムのディレクトリーを介してアクセス可能なすべてのファイル・システム内のオブジェクトに対して、これらのインターフェースを使用できます。

メニューと表示画面を利用して、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のグラフィカル・ユーザー・インターフェースである 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。

メニューおよび表示画面を使用したアクセス
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できます。

CL コマンドを使用したアクセス
CL コマンドを使用すれば、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・システム内の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。

API を使用したアクセス
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。

iSeries ナビゲーターを使用したアクセス
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。

iSeries ネットサーバーを使用したアクセス
iSeries ネットサーバーを介して、 Windows ユーザーは OS/400 の共用ディレクトリー・パスおよび共用出力待ち行列にアクセスすることができます。

ファイル転送プログラムを使用したアクセス
ファイル転送プログラムは、いくつかの異なるファイル・システムからファイルを転送します。さらに、QDLS ファイル・システム内のフォルダーと文書も転送します。

PC を使用したアクセス
iSeries に接続された PC を使用すれば、統合ファイル・システムに保管されたディレクトリーやオブジェクトと対話することができます。ディレクトリーとオブジェクトは、 PC 上に保管されているかのように表示されます。

メニューおよび表示画面を使用したアクセス

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できます。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** を押して次に進みます。
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・コマンドを使用して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
- オブジェクトの属性の表示と変更
- 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、 『ファイル・システムの処理』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アクセス

CL コマンドを使用したアクセス

統合ファイル・システムのメニューおよび表示画面から実行できる操作 (66 ページの『メニューおよび表示画面を使用したアクセス』を参照) はすべて、制御言語 (CL) コマンドを入力しても実行できます。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・システムに関連した CL コマンドについての詳細は、 34 ページの『ユーザー定義ファイル・システム (UDFS)』および 61 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実行するコマンドについては、 OS/2 と DOS のユーザーに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 7. 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。エクスポートされたリモート・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、ジャーナル対象オブジェクトが保管された後の変更内容を適用したり、特定の時点までの変更を適用する。	
CHGATR	属性の変更。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。または、ディレクトリーとその内容、およびすべてのサブディレクトリーの内容の属性が変更されたディレクトリー・ツリーの属性を変更する。	
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
CHGJRNOBJ[2]	ジャーナル・オブジェクトの変更。オブジェクトに対するジャーナル処理を終了および再始動せずに、 1 つのオブジェクトまたはオブジェクトのリストのジャーナル属性を変更する。	
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表に対して、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。	

表 7. 統合ファイル・システム・コマンド (続き)

コマンド	説明	別名
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。ユーザー定義ファイル・システムを作成する。	
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。	
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
PRTDIRINF	ディレクトリー情報の出力。ディレクトリー情報の検索 (RTVDIRINF) コマンドによって収集された、統合ファイル・システム内のオブジェクトに関するディレクトリー情報を出力する。	
RLSIFSLCK	統合ファイル・システムのロック解除。 NFS クライアントによって保持された、またはオブジェクトに対して保持されたすべてのバイト範囲ロックを解除する。	
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

表 7. 統合ファイル・システム・コマンド (続き)

コマンド	説明	別名
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。エクスポートされたリモート・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 (この変数は CL プログラムで使用される)。	
RTVDIRINF	ディレクトリー情報の検索。統合ファイル・システム内のオブジェクトの属性を収集する。	
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の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、オブジェクトを処理する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、オブジェクトを処理する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、オブジェクトを処理する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・システムでこれらのコマンドを使用するうえでの制限事項について、詳しくは以下のトピックを参照してください。

- 『ファイル・システムの処理』
- 『CL コマンドおよび表示画面のパス名規則』
- 『RTVDIRINF および PRTDIRINF コマンドの出力の処理』
- 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ってオブジェクトを操作するとき、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、現行ジョブの 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。表示画面にパス名を入力するときには、アポストロフィは任意指定です。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、最上位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わるように、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、次のようにします。

  ```
  '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、オブジェクトを格納するファイル・システムが大文字小文字を区別するかどうかによって異なります。また、オブジェクトを作成するか、検索するかによっても異なります。
- 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、 『ファイル・システムの処理』を参照してください。
- 次のように、パス名の先頭文字が / 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、パス名の先頭文字が / 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 (~) の後にユーザー名、その後にスラッシュ (または円記号) 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'
  ```

- 一部のコマンドでは、パス名の最後の構成要素の中でアスタリスク (*) または疑問符 (?) を使用して、名前のパターンを検索できます。 * は、* の位置に任意の文字 (何文字でもよい) が存在す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 つの文字が存在す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、次のようにします。

  ```
  '**.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* の形式でなければなりません。

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

  詳しくは、 40 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* の形式でなければなりません。たとえば、以下のようにします。

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

  詳しくは、 43 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)。

  たとえば、次のようにします。

  ```
  '"/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があるため、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持っています。

- 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 (87 ページの『API を使用したアクセス』を参照してください)。さらに、16 進 0 値をパス名に含めることはできません。

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。

RTVDIRINF および PRTDIRINF コマンドの出力の処理

RTVDIRINF コマンドは、統合ファイル・システム内のオブジェクトの属性を収集するために使用されます。収集された情報はデータベース・ファイル (表) に保管されます。ファイルには、INFFILEPFX パラメーターに指定された情報ファイル接頭部を使って名前が付けられます。表は、 INFLIB パラメーターによって指定されたライブラリー内に作成されます。データベース・ファイルや表にどのような名前が付けられるかについて、詳しくは RTVDIRINF を参照してください。

RTVDIRINF コマンドの結果として、 3 つの表が作成されます。 1 つの表はオブジェクト属性を保管します。もう 1 つはディレクトリー用で、最後の表は、オブジェクト属性を保管するのにどのファイルが使用されたかを判別します。

この 3 つの表の情報にアクセスする方法については、 86 ページの『RTVDIRINF のデータへのアクセス』を参照してください。これらのデータの利用方法についての詳細は、 86 ページの『RTVDIRINF のデータの使用』を参照してください。

表 8 は、オブジェクト属性を保管する表に含まれるフィールドを示しています。 INFFILEPFX パラメーターに *GEN を指定した場合、このコマンドが生成する固有の接頭部を使用してデータベース・ファイルが作成されます。この接頭部は QAEZD で始まり、その後に 4 桁が続きます。収集された情報を保管するために作成されるファイルの名前には、この接頭部が使用され、その後に文字 'D' (ディレクトリー情報を含むファイルの場合)、または文字 'O' (ディレクトリー内のオブジェクトに関する情報を含むファイルの場合) のいずれかが付きます。たとえば、*GEN を指定してこのコマンドを初めて実行した場合、ファイル QAEZD0001D および QAEZD0001O が、 INFLIB パラメーターで指定されたライブラリー内に作成されます。ユーザーは、このデータベースの命名に使用するファイル接頭部を指定できます (最大で 9 文字の長さ)。

表 8. QAEZDxxxxO (オブジェクト属性の保管)

フィールド名	フィールド・タイプ	フィールド記述
QEZDIRIDX	INTEGER	親ディレクトリー索引
QEZOBJNAM [1]	VARGRAPHIC (1024)	オブジェクト名。 [2]
QEZOBJLEN	INTEGER	オブジェクト名フィールドに含まれるバイト数。
QEZNMCCSID	INTEGER	オブジェクト名を表すのに使用される CCSID。
QEZREGION	GRAPHIC (2)	オブジェクト名の国を表す 2 文字の ID。 (照合シーケンスなど、操作の場所によって定義される操作に影響を与える。)
QEZLANGID	GRAPHIC (3)	オブジェクト名に使用されている言語を表す 3 文字の ID。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZMODE	INTEGER	ファイルのアクセス・モードとタイプ。モードの詳細については、『ファイルのオープン open() API』を参照してください。
QEZOBJTYPE [1]	GRAPHIC (10)	オブジェクト・タイプ。
QEZCCSID	INTEGER	データの CCSID およびオブジェクトの拡張属性。
QEZALCSIZE [1]	BIGINT	このオブジェクトに割り振られたバイト数。
QEZDTASIZE	BIGINT	このオブジェクト内のデータのサイズ (バイト)。このサイズには、オブジェクト・ヘッダーや、オブジェクトに関連した拡張属性のサイズは含まれません。
QEZEAS	BIGINT	このオブジェクトに関連した拡張属性の数。
QEZCEAS	BIGINT	このオブジェクトに関連した重要な拡張属性の数。
QEZEXTATRS	BIGINT	すべての拡張属性データの合計バイト数。
QEZCRTTIM	TIMESTAMP	オブジェクトが作成された日時。
QEZACCTIM	TIMESTAMP	オブジェクトのデータが最後にアクセスされた日時。
QEZCHGTIMA [1]	TIMESTAMP	オブジェクトの属性が最後に変更された日時。
QEZCHGTIMD	TIMESTAMP	オブジェクトのデータが最後に変更された日時。
QEZSTGFREE [1]	SMALLINT	オブジェクトのデータがオフラインで移動され、オンライン記憶域が解放されたかどうか。有効な値は次のとおりです。 0 - オブジェクトのデータはオフラインでありません。 1 - オブジェクトのデータはオフラインです。
QEZCHKOUT [1]	SMALLINT	オブジェクトがチェックアウトされたかどうかを示す標識。有効な値は次のとおりです。 0 - オブジェクトはチェックアウトされていません。 1 - オブジェクトはチェックアウトされています。
QEZCHKOWN	GRAPHIC (10)	オブジェクトをチェックアウトしたユーザー。チェックアウトされていない場合、このフィールドはブランクになります。
QEZCHKTIM	TIMESTAMP	オブジェクトがチェックアウトされた日時。オブジェクトがチェックアウトされていない場合、このフィールドにはデータが含まれません。
QEZLOCAL	SMALLINT	オブジェクトがローカルに保管されるか、それともリモート・システム上に保管されるか。オブジェクトをローカルに保管するかリモートに保管するかの決定は、対応するファイル・システムの規則に応じて異なります。ローカル標識とリモート標識のどちらも伝送しないファイル・システム内のオブジェクトは、リモートとして扱われます。有効な値は次のとおりです。 1 - オブジェクトのデータはローカルに保管されます。 2 - オブジェクトのデータはリモート・システム上に保管されます。
QEZOWN [1]	GRAPHIC (10)	オブジェクトの所有者であるユーザー・プロファイルの名前。または、以下の特殊値。 *NOUSRPRF - この特殊値はネットワーク・ファイル・システムによって使用され、リモート・オブジェクトのユーザー ID (UID) に一致する UID を持つユーザー・プロファイルが、ローカル iSeries サーバー上に存在しないことを示します。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZUID	INTEGER	システムの各ユーザーごとに、固有の数値であるユーザー識別番号 (UID) が必要です。
QEZOWNPGP	GRAPHIC (10)	オブジェクトの 1 次グループであるユーザー・プロファイルの名前。または、以下の特殊値。 *NONE - このオブジェクトには 1 次グループがありません。 *NOUSRPRF - この特殊値はネットワーク・ファイル・システムによって使用され、リモート・オブジェクトのグループ ID (GID) に一致する GID を持つユーザー・プロファイルがローカル・サーバー上に存在しないことを示します。
QEZGID	INTEGER	グループ・プロファイルは、固有の数値であるグループ識別番号 (GID) によって識別されます。
QEZAUTLST	GRAPHIC (10)	指定されたオブジェクトを保護するのに使用される権限リストの名前。値 *NONE は、オブジェクトの権限を決定するために権限リストが使用されないことを示します。
QEZASP	SMALLINT	ストレージが保管される補助記憶域プール。
QEZJRNSTS [1]	SMALLINT	オブジェクトの現在のジャーナル状況。このフィールドは、以下のいずれかの値になります。 0 (NOT_JOURNALED) - オブジェクトは現在、ジャーナル処理されていません。 1 (JOURNALED) - オブジェクトは現在、ジャーナル処理されています。
QEZJSUBTRE	SMALLINT	このフラグが戻されるとき、このオブジェクトは、統合ファイル・システムのジャーナル処理サブツリー・セマンティクスを持つディレクトリーです。 0 - オブジェクトは、サブツリー・セマンティクスでジャーナル処理されません。 1 - オブジェクトは、サブツリー・セマンティクスでジャーナル処理されます。このディレクトリーのサブツリー内に新しく作成されるオブジェクトは、このディレクトリーからジャーナル処理属性とオプションを継承します。
QEZJOPTENT	SMALLINT	ジャーナル処理がアクティブのとき、オプショナルと見なされる項目がジャーナル処理されます。オプション・ジャーナル項目のリストは、各オブジェクト・タイプに応じて異なります。 0 - オブジェクトの、オプション項目はジャーナル処理されません。 1 - オブジェクトの、オプション項目がジャーナル処理されます。
QEZJAFTERI	SMALLINT	ジャーナル処理がアクティブのとき、変更後のオブジェクトのイメージがジャーナル処理されます。 0 - オブジェクトの、変更後イメージはジャーナル処理されません。 1 - オブジェクトの、変更後イメージがジャーナル処理されます。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZJBEFORI	SMALLINT	ジャーナル処理がアクティブのとき、変更前のオブジェクトのイメージがジャーナル処理されます。 0 - オブジェクトの、変更前イメージはジャーナル処理されません。 1 - オブジェクトの、変更前イメージがジャーナル処理されます。
QEZJRNID	GRAPHIC (10)	このフィールドは、ジャーナル処理対象のオブジェクトに ID を関連付けます。さまざまなジャーナル処理関連コマンドおよび API でこの ID を使用できます。
QEZJRNNAM	GRAPHIC (10)	ジャーナル状況が JOURNALED の場合、このフィールドには、現在使用されているジャーナルの名前が入ります。ジャーナル状況が NOT_JOURNALED の場合、このフィールドには、このオブジェクトに関して最後に使用されていたジャーナルの名前が入ります。このオブジェクトがまだジャーナル処理されていない場合には、このフィールドのすべてのバイトが 2 進ゼロに設定されます。
QEZJRNLIB	GRAPHIC (10)	ジャーナル状況が JOURNALED の場合、このフィールドには、現在使用されているジャーナルを含むライブラリーの名前が入ります。ジャーナル状況が NOT_JOURNALED の場合、このフィールドには、最後に使用されたジャーナルを含むライブラリーの名前が入ります。このオブジェクトがまだジャーナル処理されていない場合には、このフィールドのすべてのバイトが 2 進ゼロに設定されます。
QEZJRNSTR	TIMESTAMP	オブジェクトのジャーナル処理が最後に開始された日時に対応する Epoch 以降の秒数。このオブジェクトがまだジャーナル処理されていない場合には、このフィールドが 2 進ゼロに設定されます。
QEZAUDT	GRAPHIC (10)	オブジェクトに関連した監査値。有効な値は次のとおりです。 *NONE - オブジェクトにアクセスするユーザーにかかわりなく、このオブジェクトが読み取りまたは変更されるとき、オブジェクトの監査は実行されません。 *USRPRF - 現在のユーザーが監査されている場合にのみ、このオブジェクトを監査します。このオブジェクトを監査するかどうか判別するために、現在のユーザーが検査されます。ユーザー・プロファイルを使用して、このオブジェクトの変更アクセスだけを監査するか、それとも変更アクセスと読み取りアクセスの両方を監査するかを指定できます。 *CHANGE - システムのすべてのユーザーによる、このオブジェクトへの変更アクセスをすべて監査します。 *ALL - システムのすべてのユーザーによる、このオブジェクトへのすべてのアクセスを監査します。すべてのアクセスは、読み取り操作または変更操作として定義されます。
QEZBLKSIZ	INTEGER	オブジェクトのブロック・サイズ。
QEZNLNK	INTEGER	オブジェクトへのハード・リンクの数。
QEZFILEID [1]	GRAPHIC (16)	オブジェクトのファイル ID。オブジェクトに関連付けられた識別子。Qp0lGetPathFromFileID() と、ファイル ID を指定すれば、オブジェクトのパス名を検索できます。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZFILEIDS	INTEGER	4 バイトからなるファイル ID。この数値は、ファイル・システム内のオブジェクトを一意的に識別します。この数値は、システム全体の中のオブジェクトを識別することはできません。
QEZGENID	BIGINT	ファイル ID に関連した生成 ID。
QEZFSID	BIGINT	オブジェクトが属するファイル・システム ID。この数値は、オブジェクトが属するファイル・システムを一意的に識別します。
QEZRDEV	BIGINT	オブジェクトがデバイス特殊ファイルを表す場合、それによって表される実際のデバイス。
QEZDOM	GRAPHIC (10)	オブジェクトのドメイン。有効な値は次のとおりです。 *SYSTEM - オブジェクトはシステム・ドメインに存在します。 *USER - オブジェクトはユーザー・ドメインに存在します。
QEZCRTAUD	GRAPHIC (10)	このディレクトリー内に作成されるオブジェクトに関連した監査値。有効な値は次のとおりです。 *NONE - オブジェクトにアクセスするユーザーにかかわりなく、このオブジェクトが読み取りまたは変更されるとき、オブジェクトの監査は実行されません。 *USRPRF - 現在のユーザーが監査されている場合にのみ、このオブジェクトを監査します。このオブジェクトを監査するかどうか判別するために、現在のユーザーが検査されます。ユーザー・プロファイルを使用して、このオブジェクトの変更アクセスだけを監査するか、それとも変更アクセスと読み取りアクセスの両方を監査するかを指定できます。 *CHANGE - システムのすべてのユーザーによる、このオブジェクトへの変更アクセスをすべて監査します。 *ALL - システムのすべてのユーザーによる、このオブジェクトへのすべてのアクセスを監査します。すべてのアクセスは、読み取り操作または変更操作として定義されます。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZSCN	GRAPHIC (1)	統合ファイル・システムのいずれかのスキャン関連出口点に出口プログラムが登録されている場合、オブジェクトがスキャンされるかどうかを指定します。詳しくは、 20 ページの『スキャンのサポート』を参照してください。 有効な値は次のとおりです。 'x40' (SCANNING_NO) - スキャン関連出口プログラムに記述された規則に従ってオブジェクトがスキャンされません。注: この属性を持つオブジェクトが復元されるときにファイル・システム・スキャン制御 (QSCANFSCTL) の値 *NOPOSTRST が指定されていない場合、オブジェクトは、復元後に少なくとも一度スキャンされます。 'x80' (SCANNING_YES) - オブジェクトが最後にスキャンされた以降にオブジェクトが変更された場合、またはスキャン・ソフトウェアがアップデートされた場合に、スキャン関連出口プログラムに記述された規則に従ってオブジェクトがスキャンされます。 'x20' (SCANNING_CHGONLY) - オブジェクトが最後にスキャンされた以降にオブジェクトが変更された場合に限り、スキャン関連出口プログラムに記述された規則に従ってオブジェクトがスキャンされます。スキャン・ソフトウェアがアップデートされた場合には、スキャンは実行されません。この属性は、ファイル・システム・スキャン制御 (QSCANFSCTL) システム値が *USEOCOATR に指定されている場合にのみ有効です。そうでない場合は、オブジェクトの属性が SCANNING_YES であるものとして扱われます。注: この属性を持つオブジェクトが復元されるときにファイル・システム・スキャン制御 (QSCANFSCTL) の値 *NOPOSTRST が指定されていない場合、オブジェクトは、復元後に少なくとも一度スキャンされます。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZINHSCN	GRAPHIC (1)	統合ファイル・システムのいずれかのスキャン関連出口点に出口プログラムが登録されている場合、ディレクトリー内に作成されるオブジェクトをスキャンするかどうかを指定します。詳しくは、 20 ページの『スキャンのサポート』を参照してください。 有効な値は次のとおりです。 x'40' - ディレクトリー内にオブジェクトが作成された後、スキャン関連出口プログラムに記述された規則に従ってオブジェクトがスキャンされません。注: この属性を持つオブジェクトが復元されるときにファイル・システム・スキャン制御 (QSCANFSCTL) の値 *NOPOSTRST が指定されていない場合、オブジェクトは、復元後に少なくとも一度スキャンされます。 x'80' - ディレクトリー内にオブジェクトが作成された後、オブジェクトが最後にスキャンされた以降にオブジェクトが変更された場合、またはスキャン・ソフトウェアがアップデートされた場合に、スキャン関連出口プログラムに記述された規則に従ってオブジェクトがスキャンされます。 x'20' - ディレクトリー内にオブジェクトが作成された後、オブジェクトが最後にスキャンされた以降にオブジェクトが変更された場合に限り、スキャン関連出口プログラムに記述された規則に従ってオブジェクトがスキャンされます。スキャン・ソフトウェアがアップデートされた場合には、スキャンは実行されません。この属性は、ファイル・システム・スキャン制御 (QSCANFSCTL) システム値が *USEOCOATR に指定されている場合にのみ有効です。そうでない場合は、オブジェクトの属性が SCANNING_YES であるものとして扱われます。注: この属性を持つオブジェクトが復元されるときにファイル・システム・スキャン制御 (QSCANFSCTL) の値 *NOPOSTRST が指定されていない場合、オブジェクトは、復元後に少なくとも一度スキャンされます。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZSSTATUS	GRAPHIC (1)	このオブジェクトに関連したスキャン状況。このフィールドは、以下のいずれかの値になります。 x'00' (SCAN_REQUIRED) - スキャン関連出口プログラムによってまだスキャンされていないため、あるいは、最後のスキャン以降にオブジェクト・データか CCSID が変更されたため、オブジェクトのスキャンが必要です。オブジェクト・データまたは CCSID が変更される例には、 (直接の、またはメモリー・マッピングを介した) オブジェクトへの書き込み、オブジェクトの切り捨て、オブジェクトのクリア、オブジェクトの CCSID 属性の変更などがあります。 x'01' (SCAN_SUCCESS) - オブジェクトはスキャン関連出口プログラムによってすでにスキャン済みで、その最後のスキャン要求の際、オブジェクトのスキャンが失敗しませんでした。 x'02' (SCAN_FAILURE) - オブジェクトはスキャン関連出口プログラムによってすでにスキャン済みです。その最後のスキャン要求の際、オブジェクトのスキャンが失敗し、操作が完了しませんでした。いったんオブジェクトが失敗済みとマークされると、オブジェクトが再びスキャンされるのは、オブジェクトのスキャン・シグニチャーがグローバル・スキャン・キー・シグニチャーまたは独立 ASP グループ・スキャン・キー・シグニチャーと異なるようになった時点です。このため、オブジェクトに対する処理を行う後続の要求は、スキャン失敗標識とともに失敗します。失敗する要求の例としては、オブジェクトのオープン、オブジェクトの CCSID の変更、オブジェクトのコピーなどがあります。 x'05' (SCAN_PENDING_CVN) - オブジェクトが *TYPE2 ディレクトリー内に含まれないため、ディレクトリーを変換するまではスキャンされません。 x'06' (SCAN_NOT_REQUIRED) - オブジェクトがスキャン対象外とマークされているため、オブジェクトのスキャンは必要ありません。
QEZSSIGDF	GRAPHIC (1)	スキャン・シグニチャーは、スキャン・ソフトウェアのサポート・レベルを示します。 オブジェクトが独立 ASP グループに含まれる場合、オブジェクト・スキャン・シグニチャーは関連する独立 ASP グループ・スキャン・シグニチャーと比較されます。オブジェクトが独立 ASP グループに含まれない場合、オブジェクト・スキャン・シグニチャーはグローバル・スキャン・シグニチャーの値と比較されます。このフィールドは、以下のいずれかの値になります。 x'00' - 比較されたシグニチャーは互いに異なりません。 x'01' - 比較されたシグニチャーが互いに異なります。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZSBINARY	GRAPHIC (1)	オブジェクトが以前にスキャンされたとき、バイナリー・モードでスキャンされたかどうかを示す。このフィールドは、以下のいずれかの値になります。 x'00' - オブジェクトはバイナリー・モードでスキャンされませんでした。 x'01' - オブジェクトはバイナリー・モードでスキャンされました。オブジェクトのスキャン状況が SCAN_SUCCESS であれば、オブジェクトはバイナリーで正常にスキャン済みです。オブジェクトのスキャン状況が SCAN_FAILURE であれば、オブジェクトのバイナリーでのスキャンが失敗しました。
QEZSCCSID1	INTEGER	オブジェクトが以前にスキャンされたとき、リストされた CCSID でスキャンされたかどうかを示す。オブジェクトのスキャン状況が SCAN_SUCCESS であれば、オブジェクトはこの CCSID で正常にスキャン済みです。オブジェクトのスキャン状況が SCAN_FAILURE であれば、この CCSID でのオブジェクト・スキャンが失敗しました。値 0 は、このフィールドが適用されないことを意味します。
QEZSCCSID2	INTEGER	オブジェクトが以前にスキャンされたとき、リストされた CCSID でスキャンされたかどうかを示す。オブジェクトのスキャン状況が SCAN_SUCCESS であれば、オブジェクトはこの CCSID で正常にスキャン済みです。オブジェクトのスキャン状況が SCAN_FAILURE であれば、このフィールドは 0 になります。値 0 は、このフィールドが適用されないことを意味します。
QEZUDATE	TIMESTAMP	オブジェクトが最後に使用された日付に対応する Epoch 以降の秒数。オブジェクトの作成時には、このフィールドはゼロです。 OS/400 タイプ用、またはオブジェクトが属するファイル・システム用に使用状況データが保持されない場合、このフィールドはゼロです。
QEZUDCOUNT	INTEGER	オブジェクトがこれまでに使用された日数。ファイル・システムによって、およびファイル・システムでサポートされる個々のオブジェクト・タイプによって、「使用」の意味が異なります。「使用」とは、ファイルのオープンまたはクローズを意味する場合もあれば、リンクの追加、名前変更、復元、オブジェクトのチェックアウトを指す場合もあります。このカウントは、オブジェクトが使用された一日ごとに増加し、 Qp0lSetAttr() API を呼び出すことによってゼロにリセットされます。
QEZURESET	INTEGER	使用日数カウントが最後にゼロ (0) にリセットされた日付に対応する Epoch 以降の秒数。使用日数カウントをゼロにリセットするために Qp0lSetAttr() API を呼び出すと、この日付は現在日付に設定されます。
QEZPRMLNK	SMALLINT	オブジェクトが複数の名前を持つ場合、このフィールドには、最初に検出された名前だけが示される。
QEZALWCKPW	SMALLINT	活動時保管チェックポイント処理中に、読み取りプログラムおよび書き込みプログラムとの間でストリーム・ファイル (*STMF) を共用できるかどうか。有効な値は次のとおりです。 0 - 読み取りプログラムとの間でのみ、オブジェクトを共用できます。 1 - 読み取りプログラムおよび書き込みプログラムとの間で、オブジェクトを共用できます。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZSIG [1]	SMALLINT	オブジェクトが OS/400 ディジタル署名を持っているかどうか。有効な値は次のとおりです。 0 - オブジェクトには OS/400 ディジタル署名がありません。 1 - オブジェクトには OS/400 ディジタル署名があります。
QEZSYSSIG	SMALLINT	システムが信頼するソースによってオブジェクトが署名されたかどうか。有効な値は次のとおりです。 0 - システムが信頼するソースによる署名は 1 つも存在しません。 1 - システムが信頼するソースによってオブジェクトが署名されています。オブジェクトに複数の署名が含まれる場合、少なくとも 1 つは、システムが信頼するソースによる署名です。
QEZMLTSIG	SMALLINT	オブジェクトに複数の OS/400 ディジタル署名が含まれるかどうか。有効な値は次のとおりです。 0 - オブジェクトにはただ 1 つのディジタル署名が含まれます。 1 - オブジェクトには複数のディジタル署名が含まれます。 QEZSYSSIG フィールドの値が 1 の場合、少なくとも 1 つの署名は、システムによって信頼されたソースからのものです。
QEZDSTGOPT	SMALLINT	このオプションは、指定したオブジェクト用の補助記憶域がシステムによってどのように割り振られるかを決定するために使用されます。このオプションは、「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システム内のストリーム・ファイルに関してのみ指定できます。 *TYPE1 バイト・ストリーム・ファイルに関しては、このオプションが無視されます。有効な値は次のとおりです。 0 - 補助記憶域は通常の方法で割り振られます。つまり、追加の補助記憶域が必要とされる場合、現在のスペース要件および予想される将来の要件を満たすために論理的サイズ範囲に分けて割り振られ、ディスク入出力操作の回数が最小化されます。 1 - 補助記憶域は、オブジェクトによるスペース使用量を最小化する方法で割り振られます。つまり、追加の補助記憶域が必要とされる場合、現在のスペース要件を満たすために小さなサイズの範囲に分けて割り振られます。多数の小さな範囲からなるオブジェクトにアクセスすることにより、そのオブジェクトに関するディスク入出力操作の回数が増加する可能性があります。 2 - システムは、使用されるスペースとディスク入出力操作の回数を比較考慮して、オブジェクト用の最適な補助記憶域割り振りを動的に決定します。たとえば、頻繁に読み取りおよび書き込みされるファイルが多数の小さな範囲からなる場合、ディスク入出力操作の回数を最小化するために、今後はより大きな範囲で補助記憶域が割り振られます。あるいは、頻繁に切り捨てられるファイルの場合、スペース使用量を最小化するために、今後はより小さな範囲で補助記憶域が割り振られます。さらに、このシステム、およびシステムの活動のために、ストリーム・ファイル・サイズ情報が保持されます。また、このファイル・サイズ情報は、他のオブジェクト・サイズと関連して、このオブジェクトの最適な補助記憶域割り振りを決定するうえでも役立ちます。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZMSTGOPT	SMALLINT	このオプションは、指定したオブジェクト用の主記憶域がシステムによってどのように割り振られ、使用されるかを決定します。このオプションは、「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システム内のストリーム・ファイルに関してのみ指定できます。有効な値は次のとおりです。 0 - 主記憶域は通常の方法で割り振られます。つまり、可能な限り大きな主記憶域が割り振られて使用されます。この場合、情報が主記憶域にキャッシュされるので、ディスク入出力操作の回数が最小限になります。 1 - 主記憶域は、オブジェクトによるスペース使用量を最小化する方法で割り振られます。つまり、可能な限り少ない主記憶域が割り振られて使用されます。この場合、少ない情報が主記憶域にキャッシュされるので、主記憶域の使用量が最小化されますが、ディスク入出力操作の回数は増えます。 2 - システムは、他のシステム活動や主記憶域の競合を考慮して、オブジェクト用の最適な主記憶域割り振りを動的に決定します。つまり、主記憶域の競合がほとんどない場合、ディスク入出力操作の回数を最小化するために、可能な限り大きな主記憶域が割り振られて使用されます。また、主記憶域の競合が大きい場合には、主記憶域の競合を最小化するために、より少ない主記憶域が割り振られて使用されます。このオプションは、記憶域プールのページング・オプションが *CALC の場合にのみ有効です。記憶域プールのページング・オプションが *FIXED である場合には、 STG_NORMAL と同じ動作になります。ファイル・サーバーを通してオブジェクトがアクセスされるときには、このオプションは影響を与えません。その代わり、STG_NORMAL と同じ動作になります。
QEZDIRTYP2	SMALLINT	指定されたディレクトリー・オブジェクトのフォーマット。有効な値は次のとおりです。 0 - ディレクトリー・フォーマットは *TYPE1 です。 1 - ディレクトリー・フォーマットは *TYPE2 です。 詳しくは、 10 ページの『*TYPE2 ディレクトリー』を参照してください。
QEZFILTYP2 [1]	SMALLINT	ストリーム・ファイル (*STMF) のフォーマット。有効な値は次のとおりです。 0 - ストリーム・ファイル・フォーマットは *TYPE1 です。 1 - ストリーム・ファイル・フォーマットは *TYPE2 です。 詳しくは、 17 ページの『ストリーム・ファイル』を参照してください。
QEZUDFTYP2	SMALLINT	ユーザー定義ファイル・システム内に作成されるストリーム・ファイル (*STMF) のデフォルト・ファイル・フォーマット。有効な値は次のとおりです。 0 - ストリーム・ファイル・フォーマットは *TYPE1 です。 1 - ストリーム・ファイル・フォーマットは *TYPE2 です。 詳しくは、 17 ページの『ストリーム・ファイル』を参照してください。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZNONSAV	SMALLINT	オブジェクトを保管できるかどうか。有効な値は次のとおりです。 0 - オブジェクトは保管されます。 1 - オブジェクトは保管されません。さらに、このオブジェクトがディレクトリーである場合、ディレクトリーのサブツリー内のオブジェクトは、保管対象オブジェクトとして明示的に指定されていない限り、いずれも保管されません。サブツリーには、すべてのサブディレクトリー、およびサブディレクトリー内のオブジェクトが含まれます。
QEZCLSTRSP	SMALLINT	このオブジェクトは、 xSeries サーバーが仮想ディスク・ドライブとして使用するために、統合 xSeries サーバー用に割り振られた記憶域です。 iSeries サーバーから見て、仮想ドライブは統合ファイル・システム内のバイト・ストリーム・ファイルとして認識されます。 0 - オブジェクトは仮想ディスク装置ではありません。 1 - オブジェクトは仮想ディスク装置です。
QEZCASE	SMALLINT	このオブジェクトを格納するファイル・システムでの大文字小文字の区別を示す。 0 - ファイル・システムは大文字小文字を区別しません。 1 - ファイル・システムは大文字小文字を区別します。
QEZOFLOW	SMALLINT	オブジェクトが格納されている補助記憶域プールがオーバーフローしたかどうかを示す。有効な値は次のとおりです。 0 - 補助記憶域プールはオーバーフローしていません。 1 - 補助記憶域プールがオーバーフローしています。
QEZPCREAD	SMALLINT	オブジェクトへの書き込み、オブジェクトの削除、オブジェクトの拡張属性の変更または削除、オブジェクト・サイズの変更が可能かどうか。有効な値は次のとおりです。 0 - オブジェクトを変更できます。 1 - オブジェクトは変更できません。
QEZPCHID [1]	SMALLINT	通常のディレクトリー・リストを使ってオブジェクトを表示できるかどうか。 0 - オブジェクトは隠されません。 1 - オブジェクトは隠されます。
QEZPCSYS	SMALLINT	オブジェクトがシステム・ファイルであるため、通常のディレクトリー検索から除外されるかどうか。 0 - オブジェクトはシステム・ファイルではありません。 1 - オブジェクトはシステム・ファイルです。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
QEZPCARC	SMALLINT	オブジェクトが最後に検査された後、オブジェクトが変更されたかどうか。 0 - オブジェクトは変更されていません。 1 - オブジェクトが変更されました。
QEZSYSARC	SMALLINT	オブジェクトが変更されたため、保管する必要があるかどうか。オブジェクトの変更時刻が更新されたとき、これがオンに設定されます。オブジェクトが保管されると、これがオフに設定されます。 0 - オブジェクトは変更されていないため、保管する必要はありません。 1 - オブジェクトが変更されたため、保管する必要があります。
QEZJRCVNAM	GRAPHIC(10)	ジャーナルの変更内容を正常に適用するために必要な最も古いジャーナル・レシーバー。情報の適用フィールドが PARTIAL_TRANSACTION に設定されている場合、ジャーナル・レシーバーには部分的なトランザクションの開始が含まれます。それ以外の場合、ジャーナル・レシーバーには保管操作の開始が含まれます。
QEZJRCVLIB	GRAPHIC(10)	ジャーナルの変更内容を正常に適用するために必要なジャーナル・レシーバーを含んでいるライブラリーの名前。
QEZJRCVASP	GRAPHIC(10)	ジャーナルの変更内容を正常に適用するために必要なジャーナル・レシーバーを含んでいる ASP の名前。有効な値は次のとおりです。 *SYSBAS - ジャーナル・レシーバーはシステムまたはユーザーの ASP に格納されています。 ASP デバイス - ジャーナル・レシーバーを格納している ASP デバイスの名前。
QEZJTRNI	GRAPHIC(1)	このフィールドは、コミットメント制御の境界に関連した、オブジェクトの現在の状態についての情報を示します。有効な値は次のとおりです。 x'00' (NONE) - 部分的なトランザクションは存在しません。 x'01' (PARTIAL_TRANSACTION) - 部分的なトランザクションを使用してオブジェクトが復元されました。ジャーナル変更の適用 (APYJRNCHG) コマンドまたはジャーナル変更の削除 (RMVJRNCHG) コマンドを使って部分的なトランザクションを完了またはロールバックするまでは、このオブジェクトを使用できません。 x'02' (ROLLBACK_ENDED) - 「コミットメント定義の処理 (Work with Commitment Definition)」(WRKCMTDFN) 画面の「ロールバック終了 (End Rollback)」オプションを使用して、オブジェクトのロールバック操作が終了されました。オブジェクトを使用できないため、復元することをお勧めします。最後のオプションとして、ジャーナル・オブジェクトの変更 (CHGJRNOBJ) コマンドを使ってオブジェクトを使用可能にすることもできます。ただし、この場合、オブジェクトが不整合状態のままになる可能性があります。

表 8. *QAEZDxxxxO (オブジェクト属性の保管) (続き)*

フィールド名	フィールド・タイプ	フィールド記述
注:		
1. このフィールドは、 PRTDIRINF コマンドで使用されるフィールドのサブセットに含まれます。		
2. このフィールドには、オブジェクト名だけが保管されます。パス名の残りの部分の保管場所は、表 9 の中で、ディレクトリー名の長さが 1 KB 未満の場合は QEZDIRNAM1、ディレクトリー名の長さが 1 KB を超える場合は QEZDIRNAM2 です。		

表 9 は、 RTVDIRINF コマンドによって処理されるディレクトリーをリストした表の例です。

表 9. *QAEZDxxxD (ディレクトリー属性の保管)*

フィールド名	フィールド・タイプ	フィールド記述
QEZDIRIDX	INTEGER	パス名の ID (ディレクトリーにのみ適用される)。
QEZDIRNAM1 [1]	VARGRAPHIC (1024)	親ディレクトリー・パス。パスの長さが 1 KB 未満の場合にのみ使用される。
QEZDIRNAM2 [1]	DBCLOB (16M)	親ディレクトリー・パス。パスの長さが 1 KB を超える場合にのみ使用される。 16 MB までの長さのパスを保管可能。
QEZDRCCSID	INTEGER	ディレクトリー CCSID。
QEZDREGION	GRAPHIC (2)	ディレクトリー・パスの国識別コード。
QEZLANGID	GRAPHIC (3)	ディレクトリー・パスの言語 ID。
QEZDIRLEN [1]	INTEGER	ディレクトリーのパス名の長さ。
QEZDIRFID	GRAPHIC (16)	ディレクトリーのファイル ID。オブジェクトに関連付けられた識別子。 Qp0lGetPathFromFileID() と、ファイル ID を指定すれば、オブジェクトのパス名を検索できます。
QEZDFID	INTEGER	ディレクトリーのファイル ID。
QEZDIRFSID	BIGINT	ディレクトリーのファイル・システム ID。
QEZDIRGID	BIGINT	生成 ID。
注:		
1. このフィールドは、 PRTDIRINF コマンドで使用されるフィールドのサブセットに含まれます。		

表 10 は、 RTVDIRINF コマンド使用時に作成される表の例です。 RTVDIRINF コマンドのさまざまなインスタンスによって検索される情報がどのデータベース・ファイルに保管されているかを認識するために、この表を PRTDIRINF コマンドで指定されます。

表 10. *QUSRSYS/QAEZDBFILE (作成されるファイルの保管)*

フィールド名	フィールド・タイプ	フィールド記述
QEZDIRSRC	VARGRAPHIC (5000)	DIR パラメーターで指定されたパス (RTVDIRINF)。
QEZPRCCSID	INTEGER	パスの CCSID。
QEZPREGION	GRAPHIC (2)	パスの国識別コード。
QEZPLANGID	GRAPHIC (3)	パスの言語 ID。
QEZOBJFILE [1]	VARGRAPHIC (20)	オブジェクトの属性を保管するために生成されたファイルの名前。
QEZDIRFILE [1]	VARGRAPHIC (20)	ディレクトリーの索引を保管するために生成されたファイルの名前。
QEZLIB [1]	VARGRAPHIC (20)	生成された両方のファイルが入っているライブラリー。
QEZSTRTIME	TIMESTAMP	RTVDIRINF が実行依頼された日時。

表 10. QUSRSYS/QAEZDBFILE (作成されるファイルの保管) (続き)

フィールド名	フィールド・タイプ	フィールド記述
QEZENDTIME	TIMESTAMP	RTVDIRINF が完了した日時。
注:		
1. このフィールドは、 PRTDIRINF コマンドで使用されるフィールドのサブセットに含まれます。		

RTVDIRINF のデータへのアクセス: 表のデータにアクセスする方法はいくつかあります。 RTVDIRINF コマンドを使用して生成されたデータにアクセスする方法を以下にリストします。

- PRTDIRINF コマンドの使用

  このコマンドを使用して、統合ファイル・システム内のオブジェクトとディレクトリーに関するディレクトリー情報を出力します。このコマンドが出力する情報は、ユーザーが RTVDIRINF コマンドで指定したデータベース・ファイルにすでに保管されています。

- iSeries で DB2® 表に対する照会を実行する任意の IBM プログラムまたはコマンドを使用する。

  汎用のツールとして、 SQL 対話式セッションの開始 (STRSQL) コマンド、および iSeries ナビゲーターを使用できます。

  たとえば、(RTVDIRINF コマンドによってすでに収集された) 特定のパス内にある、割り振りサイズが 10 KB より大きいオブジェクトを選択するには、以下のような照会を実行できます。

  SELECT QEZOBJNAM, QEZALCSIZE FROM library_name/QAEZDxxxxO WHERE QEZALCSIZE > 10240

- 独自のプログラムを作成して、任意の適切なデータベース機能を使ってデータベース表にアクセスできます。たとえば、 組み込み SQL や SQL CLI を使用できます。

RTVDIRINF のデータの使用: 以下にリストされる例は、データがなぜ重要か、また 3 つのそれぞれの表から生成されるデータをどのように使用できるかを示しています。

- 72 ページの表 8 では、この表内の任意のフィールドに基づくレポートや統計を作成するための照会を使用できます。 PRTDIRINF には、すべてのフィールドに基づくレポートは含まれません。その代わり、サブセットが使用されます。
- 85 ページの表 9 のデータには、 RTVDIRINF コマンドの DIR パラメーターで指定されたパス内にあるすべてのディレクトリーが含まれます。パス名の特定の属性 (たとえば CCSID、言語 ID、または長さ) を知りたい場合には、このデータが役立ちます。さらに、この表に保管される各ディレクトリーには、それを識別する固有値、あるいは索引が割り当てられます。 72 ページの表 8 には同じフィールド QEZDIRIDX があり、どのオブジェクトがどのディレクトリーに属するかを示します。どのオブジェクトがどのディレクトリーに属するかを調べるには、結合を使って照会を出すことができます。たとえば、以下の照会ステートメントは、ディレクトリー "/MYDIR" 内に存在するすべてのオブジェクトの名前を選択します。

```
SELECT QEZOBJNAM FROM library_name/QAEZxxxxO, library_name/QAEZxxxxD WHERE QEZDIRNAM1 = "/MYDIR" AND
library_name/QAEZxxxxO.QEZDIRIDX=library_name/QAEZxxxxD.QEZDIRIDX
```

- 85 ページの表 10 は多くの場合、 RTVDIRINF 実行に関する特定のデータを得るために PRTDIRINF コマンドによって使用されます。たとえば、作成された表の名前、表が入っているライブラリー、処理の開始時刻と終了時刻などのデータを取得できます。この表を使用して、RTVDIRINF が発行された時刻、あるいは、これらを照会するためにどの表を検索すべきかを知ることができます。

API を使用したアクセス

アプリケーション・プログラム・インターフェース (API) を使用して統合ファイル・システムにアクセスすることができます。 API を使って統合ファイル・システムを処理する方法の詳細については、 『API を使用した操作の実行』を参照してください。

iSeries ナビゲーターを使用したアクセス

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 ナビゲーターを使用すると、システムの運用および管理がより簡単でより生産的になります。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提供されています。

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。以下には、基本的な操作に役立つ一般的なファイル・システム・タスクがリストされています。

ファイルおよびフォルダーの処理

- 123 ページの『フォルダーの作成』
- 123 ページの『フォルダーの除去』
- 122 ページの『ファイルのチェックイン』
- 122 ページの『ファイルのチェックアウト』
- 125 ページの『許可の設定』
- 125 ページの『ファイル・テキスト変換のセットアップ』
- 125 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』
- 126 ページの『パッケージ定義のオプションの変更』
- 126 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』
- 127 ページの『オブジェクトをスキャンするかどうかの設定』

ファイル共用の処理

- 127 ページの『ファイル共用の作成』
- 127 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

- 38 ページの『統合ファイル・システム UDFS の作成』
- 38 ページの『統合ファイル・システム UDFS のマウント』
- 38 ページの『統合ファイル・システム UDFS のアンマウント』

オブジェクトのジャーナル処理

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

iSeries ネットサーバーを使用したアクセス

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するための IBM Operating System/400® (OS/400) の機能です。 iSeries ネットサーバーを使用すると、PC は、Windows ソフトウェア

を実行して 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ことができます。ネットワーク上の PC クライアントは、オペレーティング・システムのファイルおよび印刷共用機能を使用します。つまり、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ありません。

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。 iSeries からエクスポートされた NFS ファイル・システムをマウントするのと同様の方法で、共用 iSeries ネットサーバー・ディレクトリーを Samba ファイル・システム (smbfs) として Linux クライアント上にマウントできます。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・ディレクトリーから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、必要に応じて、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**」を選択して、 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・ドライブの割り当て**」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 Family 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 Family 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 を使用して共用すると、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があるかもしれません。詳しくは、 123 ページの『別のファイル・システムへのファイルまたはフォルダーの移動』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 クライアントによる iSeries ネットサーバー・ファイル共用へのアクセス

ファイル転送プログラムを使用したアクセス

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て、 iSeries サーバー上の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、不在バッチ・モードで対話式に実行することもできます。この場合、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ます。また、FTP クライアントには、サーバー上でファイルを操作する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)
- iSeries NetClient 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

PC を使用したアクセス

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 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 の間で自動変換することができます。 EBCDIC は拡張 2 進化 10 進コード、 ASCII は情報交換用米国標準コードです。この変換を自動的に実行するよう、iSeries Access Family 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) 以降、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 Family 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 Family に含まれる 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 Family では、統合ファイル・システムにアクセスするためにネットサーバーに完全に依存します。ネットサーバー・サポートは、OS/400 V4R2 以上を実行している iSeries サーバーへの TCP/IP 接続でのみ使用できます。

*TYPE1 から *TYPE2 へのディレクトリーの変換

| OS/400 V5R1 以降、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、改善されたパフォーマンスおよび信頼性を提供します。

| V5R3 がインストールされるとすぐに、 *TYPE2 ディレクトリーをサポートするようまだ変換されていないすべてのファイル・システムに関して、 *TYPE2 ディレクトリーへの自動変換が開始されます。この変換は、システム・アクティビティーには大きな影響を与えません。

- 91 ページの『*TYPE1 から *TYPE2 への変換の概要』
- 91 ページの『ヒント: 変換』

*TYPE1 から *TYPE2 への変換の概要

OS/400 V5R1 以降、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、改善されたパフォーマンスおよび信頼性を提供します。パフォーマンスと信頼性の改善に加えて、一部の新機能 (たとえば統合ファイル・システムのスキャン・サポート) を、 *TYPE2 ディレクトリー内のオブジェクトに対してのみ実行することができます。詳しくは、 20 ページの『スキャンのサポート』を参照してください。

V5R3 がインストールされるとすぐに、 *TYPE2 ディレクトリーをサポートするようまだ変換されていないすべてのファイル・システムに関して、 *TYPE2 ディレクトリーへの自動変換が開始されます。この変換は優先度の低いバックグラウンド・ジョブとして実行されるため、システム・アクティビティーに大きな影響を与えません。

変換機能が未完了で、システムが正常または異常 IPL されると、 IPL が完了した後に変換機能が再開します。可能なすべてのファイル・システムが完全に変換されるまで、IPL のたびに変換が再開します。

変換機能が完了したかどうかについては、 92 ページの『変換状況の判別』を参照してください。

この自動変換が可能なファイル・システムは、「ルート」(/)、 QOpenSys、およびユーザー定義ファイル・システム (ASP 1 から 32 用) です。独立 ASP 上のユーザー定義ファイル・システムについては、 94 ページの『ヒント: 独立 ASP』を参照してください。

注: V5R3 のインストール前にファイル・システムを変換すれば、 *TYPE2 ディレクトリーへの自動変換は実行されません。 11 ページの『OS/400 V5R1 または V5R2 での *TYPE2 ディレクトリーの使用』を参照してください。

ヒント: 変換

V5R3 がインストールされるとすぐに、 *TYPE2 ディレクトリーをサポートするようまだ変換されていないすべてのファイル・システムに関して、 *TYPE2 ディレクトリーへの自動変換が開始されます。この変換処理は、 QFILESYS1 システム・ジョブの 2 次スレッドで実行されます。

変換処理の際に考慮すべきいくつかの点があります。

- 92 ページの『変換状況の判別』
- 92 ページの『ユーザー・プロファイルの作成』
- 92 ページの『名前変更されるオブジェクト』
- 93 ページの『ユーザー・プロファイルに関する考慮事項』
- 93 ページの『補助記憶域要件』
- 94 ページの『ヒント: シンボリック・リンク』
- 94 ページの『ヒント: 独立 ASP』
- 94 ページの『ヒント: 保管と復元』
- 94 ページの『記憶域の再利用 (RCLSTG)』
- 94 ページの『統合ファイル・システムのスキャン』

変換状況の判別

V5R3 がインストールされるとすぐに、 *TYPE2 ディレクトリーをサポートするようまだ変換されていないすべてのファイル・システムに関して、 *TYPE2 ディレクトリーへの自動変換が開始されます。この変換処理は、 QFILESYS1 システム・ジョブの 2 次スレッドで実行されます。

変換処理の状況を判別するには、 ディレクトリーの変換 (CVTDIR) を以下のように使用できます。

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

この CVTDIR コマンドの呼び出しによって、「ルート」(/)、QOpenSys、および UDFS ファイル・システムの現在のディレクトリー・フォーマットがリストされ、ファイル・システムが現在変換中かどうか示されます。さらに、変換機能の現在の優先度、システムによって現在変換されているファイル・システム、そのファイル・システムに関連する処理済みリンクの数、ファイル・システムに関連する処理済みディレクトリーのパーセントがリストされます。変換機能がシステム活動に大きな影響を与えないようにするために、システムは非常に低い優先度 (99) で変換機能を開始します。ただし、CVTDIR コマンドの OPTION パラメーターに *CHGPTY 値を使用すれば、変換機能の優先度を変更できます。このパラメーターの指定についての追加情報は、 CVTDIR を参照してください。

QFILESYS1 ジョブが変換を処理しているため、 QFILESYS1 ジョブ・ログを表示して、変換に関連した問題があるかどうか調べることができます。さらに、ファイル・システムの変換に関するさまざまな進行状況メッセージが送られます。これらのメッセージには、現在変換されているファイル・システム、そのファイル・システム内の処理済みリンクの数、ファイル・システム内の処理済みディレクトリーのパーセントなどの情報が含まれます。すべてのエラー・メッセージと、ほとんどの進行状況メッセージは、 QSYSOPR メッセージ・キューにも送られます。このため、後で参照するために、これらのメッセージが入っている QHST ログまたは QFILESYS1 ジョブ・ログを保持しておくとよいかもしれません。ファイル・システムが完全に変換され、統合ファイル・システムが正常に機能していることが確認されれば、この履歴情報を削除することができます。

ユーザー・プロファイルの作成

変換機能は 1 つのユーザー・プロファイルを作成し、変換機能の実行中にこれが使用されます。このユーザー・プロファイルの名前は QP0FCWA です。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、ファイル・システム内の変換済みディレクトリーを所有するために、このユーザー・プロファイルが変換機能によって使用されます。

このユーザー・プロファイルは、可能であれば、変換が完了したときに削除されます。ディレクトリーの所有権がこのユーザー・プロファイルに与えられた場合、メッセージ CPIA08B が QFILESYS1 ジョブ・ログおよび QSYSOPR メッセージ・キューに送られます。

名前変更されるオブジェクト

*TYPE2 ディレクトリーでは、リンク名は有効な UTF-16 の名前でなければなりません。これは、UCS2 レベル 1 の名前を持つ *TYPE1 ディレクトリーとは異なります。このため、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った場合、名前は固有の有効な UTF-16 名に変更され、元の名前と新しい名前をリストするメッセージ CPIA08A が QFILESYS1 ジョブ・ログおよび QSYSOPR メッセージ・キューに送信されます。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。

文字について、詳しくは、 93 ページの『結合文字』または 93 ページの『サロゲート文字』を参照してください。

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含む文字です。すべてのオブジェクトが固有名を持つようにするために、これらの文字をディレクトリーに保管する前に、共通フォーマットに変更もしくは正規化する必要があります。結合文字の正規化とは、既知で予測可能なフォーマットに文字を変えるプロセスです。 *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あっても、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の名前が変更されます。

サロゲート文字: 文字の中には、Unicode 内に有効な表記が存在し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 文字は最初の範囲 (たとえば、0xD800 から 0xD8FF) 内にあり、 2 番目の Unicode 文字は 2 番目の範囲 (たとえば、0xDC00 から 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った場合、オブジェクトを *TYPE2 ディレクトリーにリンクする前に名前が変更されます。

ユーザー・プロファイルに関する考慮事項

変換の実行中、*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できるように、すべての試行が行われます。 *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・プロファイルの詳細について、以下のトピックを参照してください。

- 『ユーザー・プロファイル用の最大ストレージの変更』
- 『ディレクトリーの所有者の変更』

| **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は、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ディレクトリーが一時的に多数存在します。変換処理中にユーザー・プロファイル用の最大ストレージの制限に達した場合、ユーザー・プロファイルの最大ストレージ制限は増やされます。メッセージ CPIA08C が QFILESYS1 ジョブ・ログおよび QSYSOPR メッセージ・キューに送られます。

| **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・プロファイルに設定されます。メッセージ CPIA08B が QFILESYS1 ジョブ・ログおよび QSYSOPR メッセージ・キューに送られて、変換が続行します。

補助記憶域要件

| システムがファイル・システム内のディレクトリーを *TYPE2 フォーマットに変換している間、補助記憶域の要件を考慮する必要があります。補助記憶域の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 (平均して) 10 % 大きくなります。

| 変換機能が実行している間は、追加記憶域が必要です。変換機能では、ディレクトリー内に *TYPE1 バー
| ジョンと *TYPE2 バージョンの両方を同時に存在させる必要があります。

| **注:** V5R3 をインストールする前に、V5R2 の (CVTDIR) コマンドで *ESTIMATE を実行するとよいかも
| しれません。これによって、変換中に必要な補助記憶域の量を見積もることができます。

ヒント: シンボリック・リンク

シンボリック・リンクとは、別のオブジェクトへのパスを含んでいる統合ファイル・システム・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場合がいくつかあります。シンボリック・リンク内のパスのいずれかの要素が変換中に名前変更されると、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。

ヒント: 独立 ASP

| 独立 ASP 内のユーザー定義ファイル・システムがまだ *TYPE2 ディレクトリー・フォーマットに変換さ
| れていない場合、 OS/400 V5R2 以降がインストールされたシステムで初めて独立 ASP をオンに変更したとき、ユーザー定義ファイル・システムが変換されます。計画に利用するために、変換の実行にかかる時間を見積もる機能が OS/400 V5R1 で提供されています。 V5R2 以降のサーバーで独立 ASP をオンに変更する前に、独立 ASP (ASP_NAME という名前とする) がオンに変更され、アクティブである間に、 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)
```

ヒント: 保管と復元

*TYPE1 で存在するディレクトリーを、 *TYPE2 に変換済みのファイル・システム内に保管および復元することができます。同様に、*TYPE2 のディレクトリーが存在するときに *TYPE1 制限を超えない限り、*TYPE2 のディレクトリーを *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。

記憶域の再利用 (RCLSTG)

| *TYPE2 ディレクトリー・フォーマットをサポートするためにシステムが「ルート」(/)、QOpenSys、およびユーザー ASP UDFS ファイル・システムを変換している間、独立 ASP 内のディレクトリーを含むすべての統合ファイル・システム・ディレクトリーに対して、RCLSTG コマンドを実行することはできません。ただし、OMIT(*DIR) パラメーター値を使用することにより、統合ファイル・システム・ディレクトリーを除外して、ディレクトリーに関連しないオブジェクトを再利用することができます。統合ファイル・システム・ディレクトリーを含めて RCLSTG を実行するために、ファイル・システムが完全に変換された時点を判別するには、 92 ページの『変換状況の判別』を参照してください。

統合ファイル・システムのスキャン

| 「ルート」(/)、QOpenSys、およびユーザー ASP UDFS ファイル・システム内のオブジェクトは、ファイル・システムが *TYPE2 ディレクトリー・フォーマットに完全に変換されるまでは、統合ファイル・システムのスキャン関連出口点を使ってスキャンされません。ファイル・システムがまだ完全に変換されていなくても、オブジェクトをスキャンするかどうかを指定するために、 *TYPE1 および *TYPE2 ディレクトリー内のオブジェクトに関するスキャン関連属性を設定できます。統合ファイル・システムのスキャン・サポートについての詳細は、 20 ページの『スキャンのサポート』を参照してください。

| システムがオブジェクトを *TYPE1 ディレクトリー・フォーマットから *TYPE2 ディレクトリー・フォーマットに変換している間、変換済みオブジェクトは復元されたかのように見なされ、スキャン制御システム値「オブジェクト復元後の次のアクセスでスキャンを実行 (Scan on next access after object has been restored)」が考慮されます。たとえば、変換の進行中に「オブジェクト復元後の次のアクセスでスキャンを実行 (Scan on next access after object has been restored)」値が指定された場合、 *TYPE1 ディレクトリー内の「オブジェクトをスキャンしない (the object will not be scanned)」属性を指定されたオブジェクトは、ファイル・システムが完全に変換された後、少なくとも一度スキャンされます。スキャン関連のシステム値についての詳細は、 21 ページの『関連するシステム値』を参照してください。

| 統合ファイル・システムのスキャンを開始するために、特定のファイル・システムが完全に変換された時点を判別するには、 92 ページの『変換状況の判別』を参照してください。

オブジェクトのジャーナル処理

| 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。ジャーナル処理の他の主要な用途として、高可用性またはワークロード・バランシングのために、オブジェクトの変更内容を他のシステムに複製するうえでも役立ちます。

この情報は、ジャーナル管理の概要、統合ファイル・システム・オブジェクトをジャーナル処理する際の考慮事項、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説明しています。

オブジェクトのジャーナル処理に関する詳細は、以下のトピックを参照してください。

- 『ジャーナル処理の概要』
- 100 ページの『ジャーナル処理の開始』
- 100 ページの『ジャーナル処理の変更』
- 100 ページの『ジャーナル処理の終了』

ジャーナル処理の概要

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』
- 96 ページの『ジャーナル処理するオブジェクト』
- 96 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』
- 98 ページの『ジャーナル処理される操作』
- 98 ページの『ジャーナル項目についての特別な考慮事項』
- 99 ページの『複数のハード・リンクとジャーナル処理に関する考慮事項』

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関連して発生する活動の監査証跡
- ジャーナル処理できないオブジェクトに関連して発生する活動の記録

- 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
- 高可用性またはワークロード・バランシングのために、オブジェクトの変更内容を他のシステムに複製するうえでの利用
- 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できます。オブジェクトのジャーナル処理についての詳しい考慮事項は、 『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成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、再構成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、変更を再構成する際の遅延は、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。受注情報を再構成する場合、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するファイル・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・オブジェクトは、従来のシステム・インターフェース (CL コマンドや 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を実行したり、ジャーナル処理状況の情報を表示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・スペース用に iSeries の統合 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。また、ブロック特殊ファイル・オブジェクトを含む可能性のあるディレクトリーをジャーナル処理することもできません。このような例には、/dev/QASP01、/dev/QASP22、および /dev/IASPNAME があります。

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、ジャーナル処理の変更、およびジャーナル処理済み変更の適用など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できます。オプションで、組み込みリストおよび除外リストを使用でき、その際にはオブジェクト名のワイルド・カード・パターンを使用できます。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" に一致しないすべてのオブジェクトを開始するように指定できます。
- 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、属性の設定がサポートされます。ジャーナル処理対象のディレクトリーに関し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またはリンクされ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・リンクは、システムによって自動的にジャーナル処理を開始されます。

  注: ジャーナル処理属性の継承に関する考慮事項:
  - 現在存在するのと同じディレクトリー内でオブジェクトを名前変更すると、ディレクトリーの「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。
  - あるディレクトリーが、「ジャーナル処理の継承」属性がオンであるディレクトリーに移動された場合、その移動されたディレクトリーのジャーナル処理だけが開始されます (適切な場合)。その移動されたディレクトリー内のオブジェクトは影響を受けません。
  - 「ジャーナル処理の継承」属性がオンであるディレクトリーにオブジェクトが復元された場合、そのオブジェクトが以前にジャーナル処理された場合でも、そのオブジェクトのジャーナル処理は開始されません。
  - ジャーナル処理済み変更の適用 (APYJRNCHG) コマンドを使用するとき、ディレクトリーに関する「ジャーナル処理の継承」属性の現在の値は、いずれも使用されません。その代わり、適用の結果として作成されるすべてのオブジェクトのジャーナル処理が開始され、適用対象の実行時活動には依存しません。
- オブジェクト名および完全パス名が、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) 回復が提供されています。
- さまざまな書き込みプログラム・インターフェースによってサポートされる最大書き込み制限は、 2 GB -1 です。 RCVSIZOPT (*MAXOPT2 または *MAXOPT3) 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には、 *MAXOPT2 または *MAXOPT3 サポートを使っ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、『ジャーナル管理』のトピック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 内の、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、および上記のレイアウトの定義については、『ジャーナル管理』のトピックにあるジャーナル項目情報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

操作で使用されるオブジェクトまたはリンクがジャーナル処理可能なタイプである場合に限って、以下の操作がジャーナル処理されます。

- 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。

- データの書き込みまたは消去
- 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

以下の操作のジャーナル処理は、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・オブジェクトの変更 (CHGJRNOBJ) コマンド
- 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、 『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、『ジャーナル管理』のトピックにあるジャーナル項目情報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的に使用して、操作中に実行される複数の関数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) がない限り、これらのジャーナル処理操作が完了したとみなすことはできません。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) が含まれるジャーナル処理操作は、失敗した操作であり、それらの中のジャーナル項目を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プログラム、または操作を再実行するプログラムは、通常、要約項目だけを使用します。

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。

複数のハード・リンクとジャーナル処理に関する考慮事項

ハード・リンクとは何か、ハード・リンクはどのように機能するかなどの情報は、 13 ページの『ハード・リンク』を参照してください。

ジャーナル処理対象の統合ファイル・システムへのハード・リンクが複数存在する場合、リンク関係を保持するために、関連するジャーナル情報とともにすべてのリンクを保管および復元する必要があります。

いずれかのジャーナル関連コマンドで名前を指定する場合、実際には名前が複数のハード・リンクであれば、オブジェクトは「一度だけ」操作されます。その他のハード・リンクは実質的に無視されます。

| 複数のハード・リンクは同じオブジェクトを指し、ジャーナル項目には同じオブジェクトに対するファイル
| 識別子 (ファイル ID) が 1 つだけ含まれるため、パス名を表示するすべてのジャーナル・インターフェー
| ス (たとえばジャーナルの表示 (DSPJRN)) は、オブジェクトのリンク名を 1 つだけ表示します。ただし、
| オブジェクトに対してどんな名前で操作を実行しても結果は同じであるため、これは問題にはなりません。

| 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。

| iSeries ナビゲーターを介してオブジェクトの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。

| 文字ベースのインターフェースを介してオブジェクトのジャーナル処理を開始するには、ジャーナル処理の
| 開始 (STRJRN) コマンド、または QjoStartJournal API を使用できます。

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。

| ジャーナル処理の変更

| 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。オブジェクトに対するジャーナル処理をいったん開始した後、さまざま
| な理由で、ジャーナル処理を終了および再始動せずに、このオブジェクトのジャーナル属性を変更したい場
| 合があるかもしれません。ジャーナル・オブジェクトの変更 (CHGJRNOBJ) コマンドを使用して、ジャー
| ナル対象のオブジェクトを変更することができます。

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。オブジェクトに対するジャーナル処理をいったん開始した後、さまざまな理由で、このオブジェクトのジャーナル処理を終了したい場合があるかもしれません。

| iSeries ナビゲーターを介してオブジェクト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。

| 文字ベースのインターフェースを介してオブジェクトのジャーナル処理を終了するには、ジャーナル処理の
| 終了 (ENDJRN) コマンド、または QjoEndJournal API を使用できます。

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。

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作動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ているアプリケーション・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 Family 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 を使用した操作の実行
- ソケット・サポート
- 命名および国際サポート
- データ変換
- 例: 統合ファイル・システムの C 関数

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、いくつかの基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 (つまり、ストリーム・ファイルには構造がない) ことが原因です。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、通常、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されるので、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、オペレーティング・システムには、ファイル形式についての情報がほとんど、またはまったくありません。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 と CPYFRMSTMF
- CPYTOIMPF と 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、それが作成されます。

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。コマンドでは、コピーするデータを変換および再フォーマットするための、さまざまなオプションを使用することができます。

さらに、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る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 つのステップに従ってデータ変換を行うことができます。

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 (これらは、宛先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ます) 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ことを指定します。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て、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『制御言語 (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

アプリケーションでストリーム・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・システムの open()、 read()、および write() 関数を使用して、メンバーをオープンし、データを読み取り、 (このファイルまたは別のファイルに) データを書き込むことができます。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 Family 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 Family 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 Family 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、外部記述データベース・ファイル、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 Family 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 ファイルをパラメーターに指定します。
- RFROMPCB。.DTT ファイルか .TFR ファイルをパラメーターに指定します。

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) を使用すれば、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 ファイルを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。 IBM e(logo)server iSeries Access Family 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 ファイルをパラメーターに指定します。
- RFROMPCB。.DTT ファイルか .TFR ファイルをパラメーターに指定します。

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) を使用すれば、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. その後、「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

保管コマンドおよび復元コマンドで使用される保管ファイルは、テープまたはディスケットに別途書き込むためのデータを保持します。このファイルをデータベース・ファイルと同じように利用して、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。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保管ファイルを使用することもできます。

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 (その逆のコピーも可能です)。ただし、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 (最初の保管ファイルからストリーム・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り、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・オブジェクトに対する操作を実行するほとんどのアプリケーション・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 Integrated Language Environment® (ILE) 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・システムがサポートする出口プログラムについての詳細は、 111 ページの表 12 を参照してください。

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してのみ作動します。以下の API がサポートされています。

表 11. 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・ファイル記述子を複写する
dup2()	オープン・ファイル記述子を別の記述子に複写する
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
fclear()	ファイルをクリアする
fclear64()	ファイルをクリアする (ラージ・ファイル使用可能)
fcntl()	ファイル制御処置を実行する
fpathconf()	構成可能なパス名変数を記述子によって入手する

表 11. 統合ファイル・システム API (続き)

関数	説明
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
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

表 11. 統合ファイル・システム API (続き)

関数	説明
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)
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)

表 11. 統合ファイル・システム API (続き)

関数	説明
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()	ファイルのアクセス回数および変更回数を設定する (NLS 化パス名を使用)
QP0FPTOS()	各種ファイル・システム機能を実行する
QP0LCHSG()	スキャン・シグニチャーを変更する
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

表 11. 統合ファイル・システム API (続き)

関数	説明
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
QP0LRRO()	参照されるオブジェクトを検索する
QP0LRTSG()	スキャン・シグニチャーを検索する
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
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

表 11. 統合ファイル・システム API (続き)

関数	説明
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、 『ファイル・システムの処理』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、 117 ページの『例: 統合ファイル・システムの C 関数』を参照してください。

表 12. 統合ファイル・システムの出口プログラム

関数	説明
クローズ API 使用時の統合ファイル・システム・スキャン	close() API などを使用したクローズ処理の間に呼び出されます。この出口プログラムは、ユーザーが提供する必要があります。
オープン API 使用時の統合ファイル・システム・スキャン	open() API などを使用したオープン処理の間に呼び出されます。この出口プログラムは、ユーザーが提供する必要があります。
パス名の処理	API の検索の中で、呼び出し側の選択基準に一致する各オブジェクトに対する Qp0lProcessSubtree() API によって呼び出されます。この出口プログラムは、ユーザーが提供する必要があります。
ストレージを解放して保管	*STMF iSeries オブジェクト・タイプを保管するために、Qp0lSaveStgFree() API によって呼び出されます。

統合ファイル・システムの 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
- API によるデータのコピー
- iSeries Information Center にある『Application programming interfaces (API)』のトピック

ILE C/400 の関数

ILE C/400 は、米国規格協会 (ANSI) によって定義された標準 C 関数を提供します。これらの関数は、C プログラムの作成時にどちらを指定するかに応じて、データ管理入出力サポートまたは統合ファイル・システム・ストリーム入出力サポートを介して作動します。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) コマンド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) パラメーターで、 *IFSIO を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。

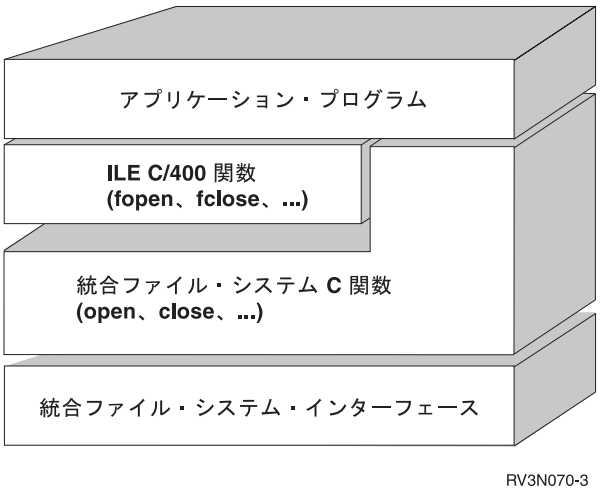


図 10. ILE C/400 関数の統合ファイル・システム・ストリーム入出力関数の使用

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。

ラージ・ファイル・サポート

| 統合ファイル・システム API が拡張されて、非常に大きなファイルの保管や操作をアプリケーションで実
| 行できるようになりました。統合ファイル・システムを使用すると、「ルート」(/)、QOpenSys、およびユ
| ーザー定義ファイル・システムで、最大約 1 テラ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 のセットが提供されているので、既存の 32 ビット API を 64 ビット API に簡単にマッピングすることにより、 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ってマクロ・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場合、動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ときには、ディレクトリー・パスを指定して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上位レベルから、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、 26 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、パス名の先頭文字が / 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ません。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、次のようにします。

  ```
  '**.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ことに注意してください。
- 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* の形式でなければなりません。

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

  詳しくは、 40 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* の形式でなければなりません。たとえば、以下のようにします。

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

  詳しくは、 43 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持っています。
- 統合ファイル・システム・コマンドのパス名 (70 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。
- (NLS を使用可能なパス名を使って) Qlg API インターフェースを使用するとき、パス名の文字の中にヌル文字を含めることはできません (パス名区切り文字としてヌル文字が指定されていない限り)。

ファイル記述子

ファイルの操作を実行するために、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する場合、ポインターを使用してファイルを識別します。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ってオープン・ファイルを識別します。ファイル記述子は、統合ファイル・システムを操作する C 関数では変数 *fildes* で表され、ソケットを操作する 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。

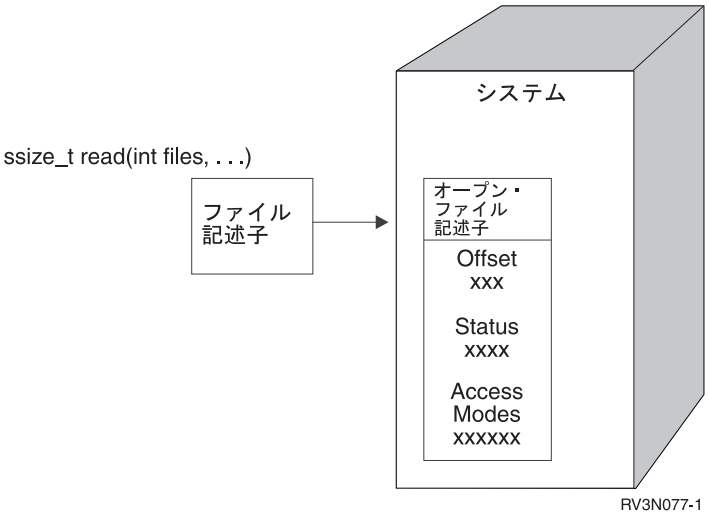


図 11. ファイル記述子とオープン・ファイル記述

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。これを行うには、 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します。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する場合と同様に、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・ジョブだけ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・プロファイルの下で実行されます。

サーバーにおける権限は、 UNIX システムにおける**許可**と同等です。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。ファイルまたはディレクトリーの 『アクセス・モード』 を構成する許可ビットのセットによって、許可が識別されます。「モード変更」関数 chmod() または fchmod() を使用することにより、許可ビットを変更できます。また、ジョブでファイルが作成されるごとに、どのファイル許可ビットを設定するかを制御するために、 umask() 関数を使用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、さまざまな各国語およびデバイス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 (18 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になりますが、番号記号がポンド記号に変わることを前提としてはなりません。

オブジェクトの拡張属性の名前もまた、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・システムを介してファイルにアクセスするとき、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
ファイルに対する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、ファイルの終わりなど) が、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、レコードの終わりまで、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できます。

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。

例: 統合ファイル・システムの C 関数

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。このプログラムは、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。完全な互換性を実現するために、文字固定情報 (オブジェクトやパス名など) は、 API を渡す前にジョブの CCSID に変換される必要があります。

```
#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;
}
```

| **注:** 法律上の重要な情報について、 2 ページの『コードに関する特記事項』をご覧ください。

iSeries ナビゲーターを使用したファイルおよびフォルダーの処理

ファイルとフォルダーに対して、以下のタスクを実行することができます。

- 『ファイルのチェックイン』
- 『ファイルのチェックアウト』
- 123 ページの『フォルダーの作成』
- 123 ページの『フォルダーの除去』
- 123 ページの『別のファイル・システムへのファイルまたはフォルダーの移動』
- 125 ページの『許可の設定』
- 125 ページの『ファイル・テキスト変換のセットアップ』
- 125 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』
- 126 ページの『パッケージ定義のオプションの変更』
- 126 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』
- 127 ページの『ファイル共用の作成』
- 127 ページの『ファイル共用の変更』
- 127 ページの『オブジェクトをスキャンするかどうかの設定』

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。

ドラッグによって実行したい場合、同じサーバー上の統合ファイル・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、別のシステム、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。

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。

さらに、このフォルダー内に作成されるオブジェクトをスキャンの対象とするかどうかを考慮する必要があります。このトピックの詳細については、 127 ページの『オブジェクトをスキャンするかどうかの設定』を参照してください。

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. 除去する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。

別のファイル・システムへのファイルまたはフォルダーの移動

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。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があるかもしれません。オブジェクトを別のファイル・システムに移動する前に、統合ファイル・システムのさまざまなファイル・システムおよびその特性を理解しておく必要があります。詳しくは、 『ファイル・システムの処理』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。

- 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アプリケーションがあるかどうか。

  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、階層ファイル・システム (HFS) をサポートしています。API と

コマンドは文書オブジェクトおよびフォルダー・オブジェクトを処理します。これらのインターフェースを、別のファイル・システムのオブジェクトに対して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ません。

  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。ほとんどの 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 ファイルに対して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。それぞれのファイル・システムの特性の比較については、 26 ページの『ファイル・システムの比較』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・システムから保管したオブジェクトを、別のファイル・システムに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、それらの属性を複写するかどうか決めます。たとえば、ディレクトリーの作成者が所有者であり、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) コマンドで OWNER(*KEEP) パラメーターを指定することによっても、オブジェクトの所有者を保持できます。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するのは不適切かもしれません。属性や権限が廃棄されている場合があるためです。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。

許可の設定

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できます。さまざまな許可を使用して、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。これで、ユーザーまたはグループが、リストの始めに追加されます。
8. 詳細な許可をインプリメントするには、「**詳細**」ボタンをクリックします。
9. 該当するチェック・ボックスをチェックして、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 『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。

作成されたパッケージ定義は保管され、複数のエンドポイント・システムやシステム・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れば、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、後で配布するためにファイルのスナップショットを取ってファイルを保持することができます。こうして、この同じファイル・グループを論理セットまたは物理セットと見なすことができます。

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. 125 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. 「**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されるファイルで置き換えます。
3. 必要に応じて、これらのオプションを変更してください。
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、ユーザーにとって都合のよい日時にファイルまたはフォルダーの送信を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. 125 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・ディレクトリーから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・ディレクトリーから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。

オブジェクトをスキャンするかどうかの設定

オブジェクトをスキャンするかどうか設定するには、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. 「**統合ファイル・システム**」を展開します。
4. 該当するフォルダーまたはファイルを展開します。
5. フォルダーまたはファイルを右クリックし、「**プロパティー**」を選択します。
6. 「**セキュリティー**」タブをクリックします。
7. 「**オブジェクトのスキャン (Scan objects)**」を選択して、必要なオプションを指定します。

オプションについて、詳しくは以下を参照してください。これらのオプションの説明は、ファイルに関するものです。スキャン対象に指定できるのは、ファイルのみです。フォルダーおよびユーザー定義ファイル・システムに関しては、そのフォルダーまたはユーザー定義ファイル・システムの中に作成されるファイルにどんなスキャン属性を設定するかを指定できます。

- はい (Yes)

| オブジェクトが最後にスキャンされた以降に、オブジェクトが変更された場合、またはにスキャン・ソフトウェアがアップデートされた場合には、スキャン関連出口プログラムに記述された規則に従ってオブジェクトがスキャンされます。

- いいえ (No)

  オブジェクトはスキャン関連出口プログラムによってスキャンされません。

  注: この属性を持つオブジェクトが復元されるとき、「オブジェクト復元後の次のアクセスでスキャンを実行 (Scan on next access after object has been restored)」オプションがシステム値に指定されている場合には、オブジェクトは復元後に少なくとも一度スキャンされます。

- オブジェクト変更時のみ (Only when the object has changed)

  オブジェクトが最後にスキャンされた以降にオブジェクトが変更された場合に限り、スキャン関連出口プログラムに記述された規則に従ってオブジェクトがスキャンされます。スキャン・ソフトウェアがアップデートされた場合には、スキャンは実行されません。

  「オブジェクト変更時のみ属性を使用してスキャンを制御する (Use 'only when objects have changed' attribute to control scan)」システム値が指定されて**いない**場合、この「オブジェクト変更時のみ (Only when the object has changed)」属性は使用されません。オブジェクトは、変更された後、およびスキャン・ソフトウェアが更新を示したときにスキャンされます。

注: このファイル用のタブでは、オブジェクトのスキャン状況を判別することもできます。

注: この属性を持つオブジェクトが復元されるとき、「オブジェクト復元後の次のアクセスでスキャンを実行 (Scan on next access after object has been restored)」オプションがシステム値に指定されている場合には、オブジェクトは復元後に少なくとも一度スキャンされます。

統合ファイル・システムについての関連情報

以下のトピックは、統合ファイル・システムについての関連情報です。

- 『トランスポート独立リモート・プロシージャー・コール』
- 133 ページの『参考資料』
- 134 ページの『経験事例』

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。

ネットワーク全体にわたる分散アプリケーションの開発は、 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より重視したアプリケーションです。

トランスポート独立リモート・プロシージャー・コールの詳細については、以下のトピックを参照してください。

- 129 ページの『ネットワークの選択』
- 129 ページの『名前からアドレスへの変換』

- 『外部データ表示 (XDR)』
- 131 ページの『認証』
- 131 ページの『トランスポート独立 RPC (TI-RPC)』

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手段を提供します。

これらの API を使用するには、 *STMF /etc/netconfig ファイルがシステム上に存在し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取得できます。

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。
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って指示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。
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、シリアライゼーション中にヌル・ポインターに特別な意味を付けることはない。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()	符号なし 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 言語の符号なし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 言語の符号なし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。

API	説明
xdr_u_short()	C 言語の符号なし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 は符号なし整数の最大値。 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 パラメーターによって示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 認証ハンドルを作成し、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。

トランスポート独立 RPC (TI-RPC) の詳細については、以下のトピックを参照してください。

- 『TI-RPC 単純化 API』
- 『TI-RPC 最上位 API』
- 132 ページの『TI-RPC 中間レベル API』
- 132 ページの『TI-RPC エキスパート・レベル API』
- 132 ページの『他の TI-RPC API』

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 ハンドルの詳細に対するより高いレベルの制御を提供します。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 と連動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。

API	説明
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。

参考資料

この参考文献リストは、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 サーバー情報を示しています。

- 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、および統合ファイル・システムの出口点または出口プログラム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)、ローカル・ジャーナル、リモート・ジャーナルのセットアップ、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、および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。統合ファイル・システムのスキャン関連処理に影響を与えるセキュリティー関連システム値についても説明しています。

- 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。

経験事例

IBM のデベロッパーたちによって書かれた経験事例には、実世界のシナリオやソリューションをインプリメントするうえでの実践的な情報が掲載されています。 IBM デベロッパーたちの実体験を活用することにより、ステップバイステップの詳細説明やヒントを参考にしながら、特定の iSeries ソリューションをインプリメントすることができます。

以下は、ファイルおよびファイル・システムに関連した経験事例です。

統合ファイル・システムのバックアップ

付録. 特記事項

本書は製造元が提供する製品およびサービスについて作成したものであり、米国以外の国においては本書で述べる製品、サービス、またはプログラムを提供しない場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

| IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信
| ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

| IBM Corporation
| Software Interoperability Coordinator, Department 49XA
| 3605 Highway 52 N
| Rochester, MN 55901
| U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。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。従って 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。お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、 IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

本書には、プログラムを作成するユーザーが統合ファイル・システムのサービスを使用するためのプログラミング・インターフェースが記述されています。

商標

以下は、IBM Corporation の商標です。

Application System/400
AS/400
e (ロゴ)
Freelance
IBM
iSeries
Operating System/400
OS/400
400
C/400
DB2
Integrated Language Environment
Lotus
OfficeVision
OS/2
WebSphere
WordPro
xSeries

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

資料に関するご使用条件

お客様がダウンロードされる資料につきましては、以下の条件にお客様が同意されることを条件にその使用が認められます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、 IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。ただし、 IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、 IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。 IBM は、これらの資料の内容 についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

これらの資料の著作権はすべて、 IBM Corporation に帰属しています。

お客様が、このサイトから資料をダウンロードまたは印刷することにより、これらの条件に同意されたものとさせていただきます。

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