IBM i

ファイルおよびファイル・システム
統合ファイル・システム

7.1

IBM i

ファイルおよびファイル・システム
統合ファイル・システム

7.1

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、 167 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 i 7.1 (製品番号 5770-SS1) 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。

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： IBM i
Files and file systems
Integrated file system
7.1

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2010.4

目次

統合ファイル・システム

統合ファイル・システムは i5/OS® オペレーティング・システムの一部であり、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、システムに格納されているすべての情報の統合構造も提供します。

注: このコーディング例を使用することによって、 165 ページの『コードに関するライセンス情報および特記事項』の条件に合意することになります。

IBM i 7.1 の新機能

統合ファイル・システムのトピック・コレクションの新しい情報または変更された情報についてお読みください。

一時ユーザー定義ファイル・システム

一時ユーザー定義ファイル・システム内の一時オブジェクトの使用。

詳しくは、 39 ページの『一時ユーザー定義ファイル・システム』を参照してください。

新規情報または変更情報の見分け方

技術上の変更が加えられた場所を見分けるのに役立つように、Information Center では以下のイメージを使用しています。

- » イメージにより、新規または変更された情報の開始点を示します。
- « イメージにより、新規または変更された情報の終了点を示します。

PDF ファイルでは、左マージンに新規および変更情報のリビジョン・バー (|) があります。

今回のリリースの新規情報または変更情報に関するその他の情報は、プログラム資料説明書を参照してください。

統合ファイル・システム用の PDF ファイル

この情報の PDF ファイルを表示または印刷できます。

この文書の PDF バージョン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. ご使用のブラウザーで PDF リンクを右クリックする。
2. PDF をローカルに保存するオプションをクリックする。
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 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® Reader がご使用のシステムにインストールされている必要があります。このアプリケーションは、 Adobe Web サイト (www.adobe.com/products/acrobat/readstep.html) から無償でダウンロードできます。

統合ファイル・システムの概要

ここでは、i5/OS オペレーティング・システムの統合ファイル・システム、およびシステムでのその使用について基本的な情報が得られます。

統合ファイル・システムとは

*統合ファイル・システム*とは、i5/OS オペレーティング・システムの一部であり、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 10 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。

統合ファイル・システムの重要な機能は以下のとおりです。

- 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は、ストリーム・ファイルだけでなく、データベース・ファイル、文書、およびシステムに保管されている他のオブジェクトにもアクセスすることができます。
- ご使用のシステム、Integrated xSeries® Server (IXS)、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、別のリモート System i® 製品、またはネットワーク・ファイル・システム (NFS) サーバーにリモートに保管することもできます。

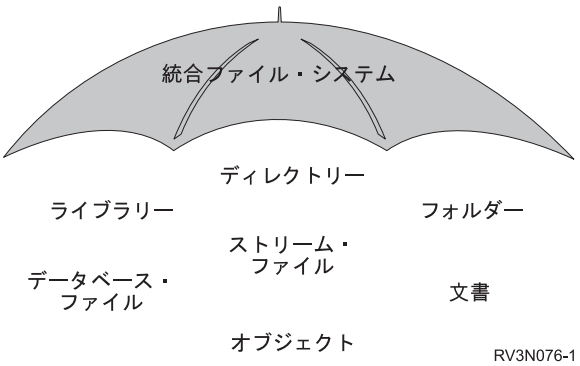

図 1. i5/OS オペレーティング・システムに保管される全情報の構造

関連概念

25 ページの『ファイル・システム』
ファイル・システムは、論理単位として編成された記憶域の特定のセグメントへのアクセスを提供します。システムの論理単位とは、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。

統合ファイル・システムを使用する理由

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、i5/OS 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、以下のことが実行できます。

- i5/OS ファイル・サーバーを使用する System i Access などのアプリケーションでは特に、 i5/OS データに迅速にアクセスすることができます。
- 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 オペレーティング・システムをベースにしたオープン・システム標準仕様 (Portable Operating System Interface for Computer Environments (POSIX) および X/Open Portability Guide (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 オペレーティング・システム・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。

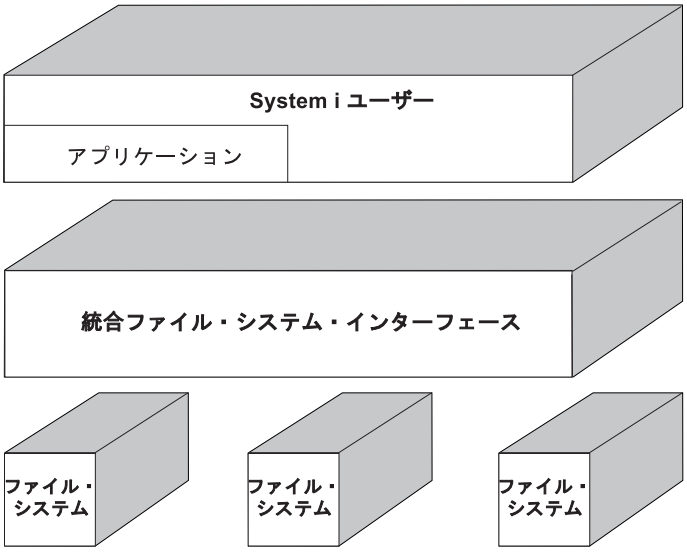

図 2. 別々のファイル・システムへの共通インターフェース

- 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、i5/OS 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。

関連概念

25 ページの『ファイル・システム』
ファイル・システムは、論理単位として編成された記憶域の特定のセグメントへのアクセスを提供します。システムの論理単位とは、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。

統合ファイル・システムの概念

このトピックでは、ディレクトリー、リンク、パス名、ストリーム・ファイル、名前の継続性、拡張属性、およびスキャン・サポートなどの、統合ファイル・システムの基本概念を紹介します。

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。そのリストには、他のディレクトリーを含めることができます。

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、システム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。

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。統合ファイル・システムのディレクトリー・ツリーの例は、以下の図を参照してください。

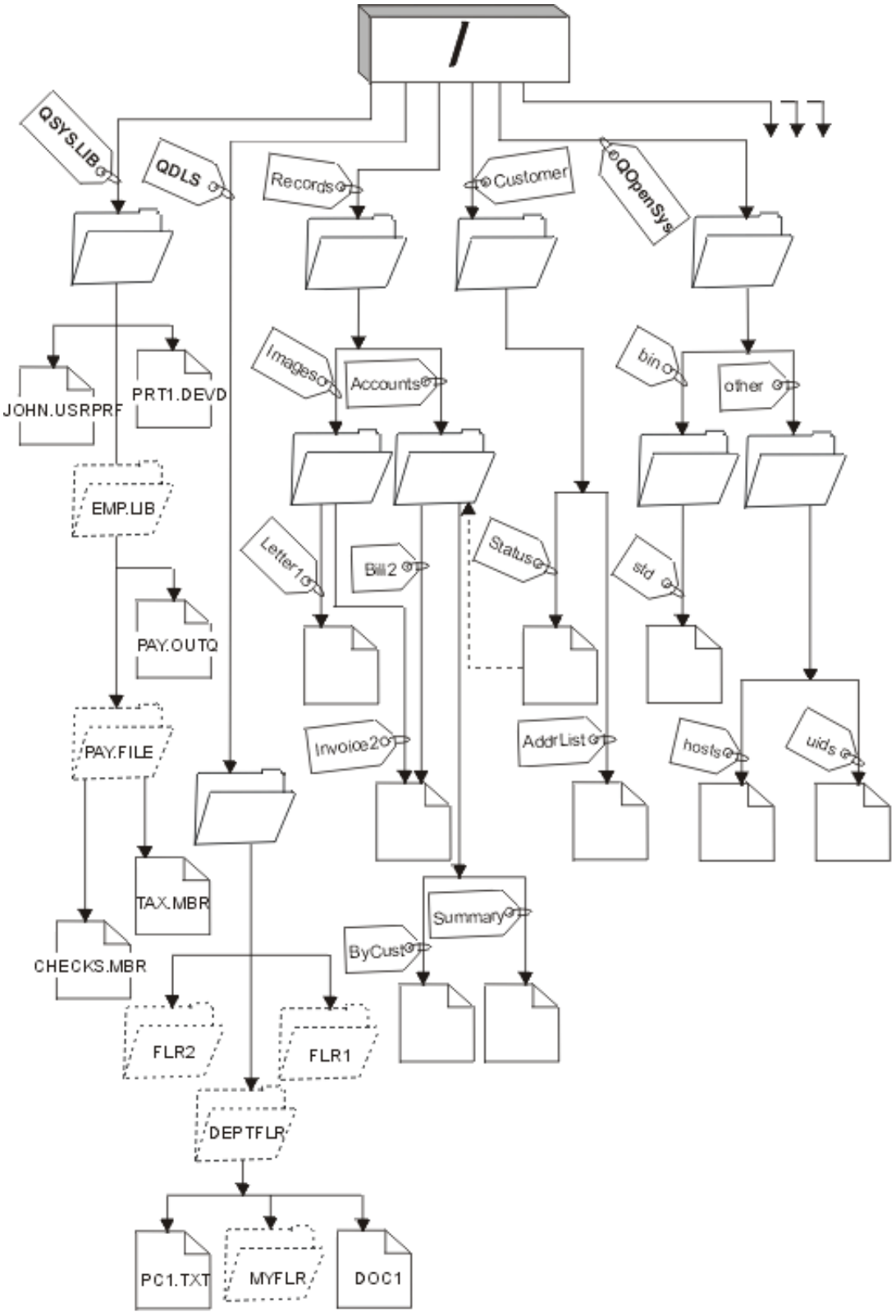

図 3. サンプルの統合ファイル・システムのディレクトリー・ツリー

現行ディレクトリー

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。

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。

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。 各ユーザーの個別のホーム・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更しなく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・ディレクトリーが存在しない場合は「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。

関連情報

現行ディレクトリーの変更 (CHGCURDIR) コマンド

chdir()--Change Current Directory API

fchdir()--Change Current Directory by Descriptor API

アプリケーション・プログラミング・インターフェース (API)

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まだ存在しない場合は、統合ファイル・システムによってそれらが作成されます。システムによる作成後、これらのディレクトリーの移動や名前変更は実行しないでください。

注: 以下のシステム作成ディレクトリーは、他のオブジェクトへのシンボリック・リンクと置き換えないでください。例えば、/home は、独立 ASP 上のディレクトリーへのシンボリック・リンクと置き換えないでください。そうでない場合、独立 ASP での問題や、新規のユーザー・プロファイルの作成時の問題となることがあります。

/tmp /tmp ディレクトリーは、アプリケーションが一時オブジェクト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オブジェクトが /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 からオブジェクトを除去したり、 /tmp 内のオブジェクト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オブジェクトを管理するために、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オブジェクトをコピーしたり、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・プログラミング・インターフェース (API) を使用して、 /tmp ディレクトリーとその中のオブジェクトを管理できます。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 内のオブジェクトを除去することができます。

/tmp ディレクトリーは、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。

オペレーティング・システムの共通性やセキュリティーのために、/tmp ディレクトリーでは、制限された名前変更およびリンク解除属性を「はい」に設定できます。

注: 制限された名前変更およびリンク解除属性はディレクトリーの S_ISVTX モード・ビットと同等のものです。

制限された名前変更およびリンク解除属性が「はい」に設定された場合は、以下の条件の 1 つが真でなければ、 /tmp ディレクトリー内のオブジェクトの名前変更またはリンク解除はできません。

- ユーザーがオブジェクトの所有者である。
- ユーザーがディレクトリーの所有者である。
- ユーザーに全オブジェクト (*ALLOBJ) 特殊権限がある

この属性が「はい」に設定されて、適切な権限がユーザーにない場合は、以下のコマンドまたは API を使用した時の名前変更およびリンク解除の失敗では、エラー番号 3027 (EPERM) またはメッセージ MSGCPFA0B1 (要求された操作は許可されない。アクセス問題) が出されます。

- リンクの除去 (RMVLNK、DEL、および ERASE) コマンド
- ディレクトリーの除去 (RMVDIR、RD、および RMDIR) コマンド
- オブジェクトの名前変更 (RNM および REN) コマンド
- オブジェクトの移動 (MOV および MOVE) コマンド
- ファイルまたはディレクトリーの名前変更 (rename()) API
- ファイルまたはディレクトリーの名前変更、存在する場合は「新規」の保持 (Qp0lRenameKeep()) API
- ファイルまたはディレクトリーの名前変更、存在する場合は「新規」のリンク解除 (Qp0lRenameUnlink()) API
- ディレクトリーの名前変更 (rmdir()) API
- ファイルへのリンクの除去 (unlink()) API

ユーザーがオブジェクトの所有者であるか、あるいはユーザーが全オブジェクト (*ALLOBJ) 特殊権限をもっている場合は、制限された名前変更およびリンク解除属性および S_ISVTX モード・ビットは、Change Attribute (CHGATR) コマンドまたは 属性の設定 (Qp0lSetAttr()) または ファイル権限の変更 (chmod) API を使用して変更できます。ただし、この属性が「いいえ」に変更された場合は、「はい」の設定で提供されたオペレーティング・システムの共通性やセキュリティーの利点は失われます。

システムの再始動中に /tmp ディレクトリーが作成されると、その属性は「はい」に設定されます。システムの再始動中に /tmp ディレクトリーがすでに存在すると、その属性は変更されません。

/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)。

/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・ディレクトリーであり、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。

関連概念

6 ページの『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。

関連資料

34 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。

35 ページの『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 X/Open Portability Guide (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。

関連情報

オブジェクト・リンクの処理 (WRKLNK) コマンド

*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。

注: *TYPE1 と *TYPE2 のストリーム・ファイルの概念は、*TYPE1 と *TYPE2 のディレクトリー形式の概念とは異なっています。相互の関連性はありません。

*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
- 多くの場合、補助記憶域スペースが小さい

*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。

- *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。
- *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。
- System i 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。
- 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 または i5/OS V5R3 以降が事前インストールされる新規 System i プラットフォームには、*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。
- OS/400 V5R2、i5/OS V5R3、またはそれ以降のリリースを、System i プラットフォームに初めてインストールする場合、プラットフォームには *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。
- 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。 変換ユーティリティーについての詳細は、「V5R2 iSeries Information Center」の『*TYPE2 ディレクトリーへの変換』セクションを参照してください。
- 独立 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 または i5/OS V5R3 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。

- 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、i5/OS V5R3 またはそれ以降のリリースのインストールの後で開始します。 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ないはずです。

ご使用のシステム上のファイル・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 ディレクトリーの変換 (CVTDIR) 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 V5R2 または i5/OS V5R3 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。

関連資料

106 ページの『*TYPE1 から *TYPE2 へのディレクトリーの変換』
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。

関連情報

記憶域の再利用 (RCLSTG) コマンド

ディレクトリーの変換 (CVTDIR) コマンド

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、システム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 (6 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、ハード・リンクとシンボリック・リンクの 2 種類があります。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。

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。

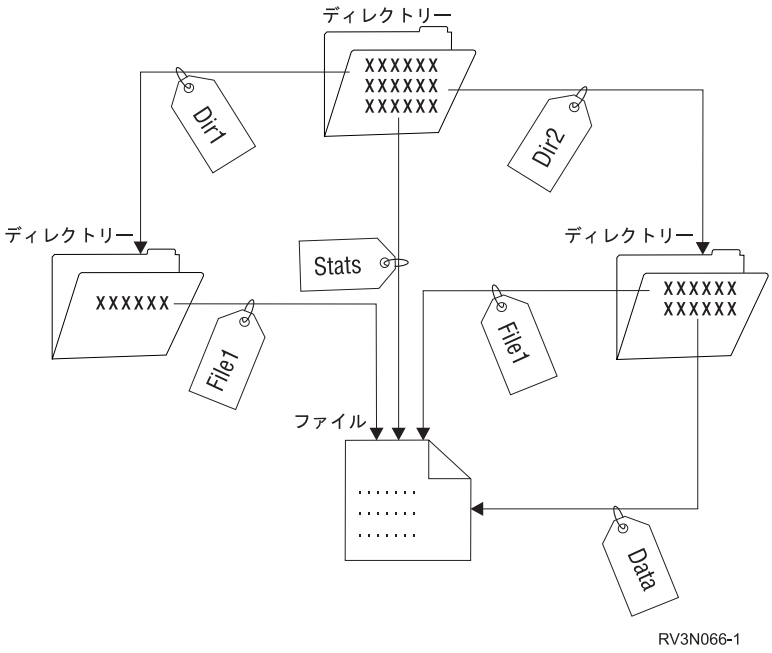

図 4. 各ハード・リンクを定義するディレクトリー項目

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システムから削除されます。 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システムから削除されます。 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。

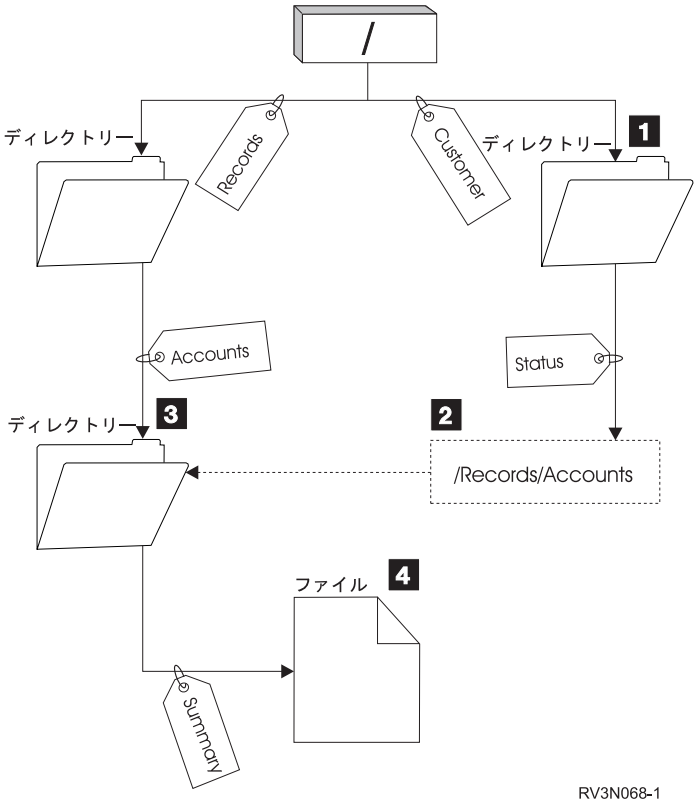

図 5. シンボリック・リンクの使用例

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。このパスでもう 1 つのディレクトリー 3 に移動します。これで、システムはプログラムが提供するパス名のパスを完了します。*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。

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システムに指示します。

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。

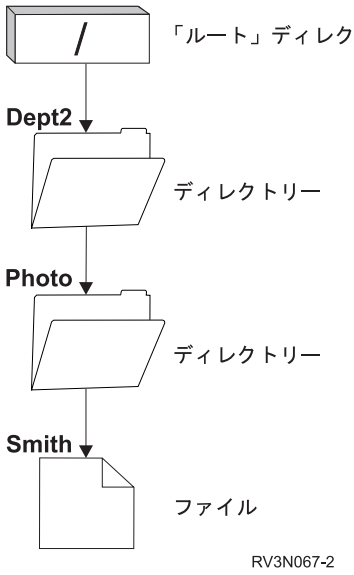

図 6. パス名の構成要素

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。

関連資料

139 ページの『API のパス名規則』
統合ファイル・システムまたは ILE C 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。 これは、API でパス名を指定する際の考慮すべき規則の要約です。

82 ページの『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。

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。システム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 i5/OS 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。

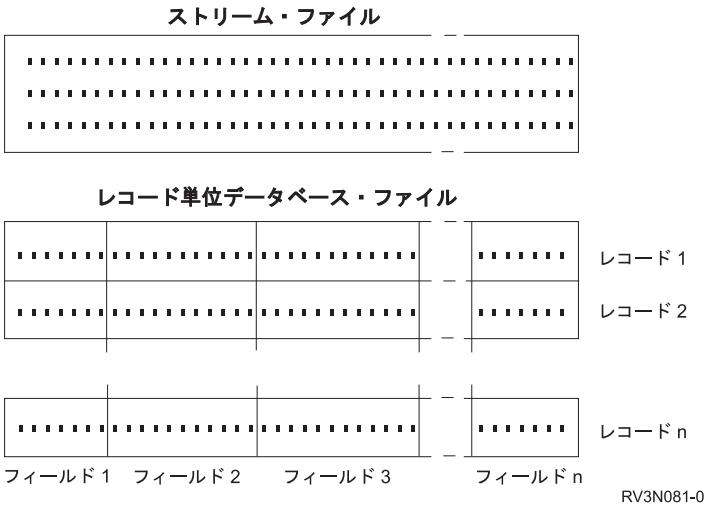

図 7. ストリーム・ファイルとレコード単位ファイルの比較

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、システム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。

注: *TYPE1 と *TYPE2 のストリーム・ファイルの概念は、*TYPE1 と *TYPE2 のディレクトリー形式の概念とは異なっています。相互の関連性はありません。

*TYPE1 ストリーム・ファイル

*TYPE1 ストリーム・ファイルは、 OS/400 V4R4 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。

*TYPE1 ストリーム・ファイルの最大オブジェクト・サイズは約 128 GB (1 GB は約 1 073 741 824 バイト) です。

*TYPE2 ストリーム・ファイル

*TYPE2 ストリーム・ファイルには、高性能ファイル・アクセスがあります。

「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムでは、*TYPE2 ストリーム・ファイルの最大オブジェクト・サイズは約 1 TB (1 TB は約 1 099 511 627 776 バイト) です。それ以外の場合は、最大サイズは約 256 GB 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。 OS/400 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 GB より大きいファイルは、i5/OS 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) を使用する複数のシステム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。 システムでは、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。 UCS2 レベル 1 および UTF-16 は ISO 10646 規格のサブセットです。名前が使用されると、システムは保管される文字形式を、使用されているコード・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。

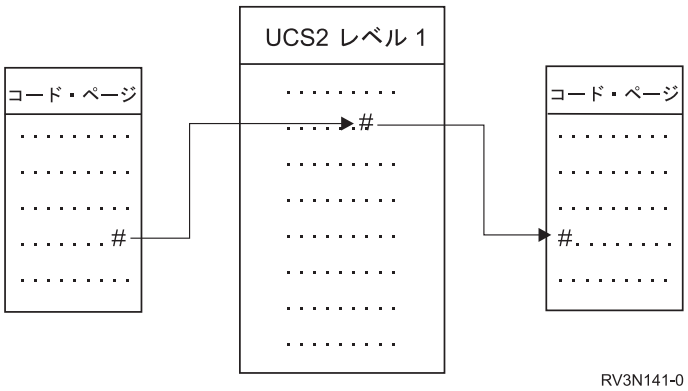

図 8. エンコード・スキーム間での同じ文字の保証

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から、システムと容易に対話することができます。例えば、PC が i5/OS システムとは異なるコード・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システム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、システムによって自動的に処理されます。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。

関連概念

9 ページの『*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。

112 ページの『自動名前変換の概要』
「ルート」(/) や CASE (*MONO) により作成される UDFS などの大文字と小文字を区別しないファイル・システムは、Unicode 規格 4.0 形式で格納される名前をサポートします。システムは、名前に使用されている追加の文字をサポートするために自動名前変換を実行します。

拡張属性

拡張属性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。拡張属性は、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。

オブジェクトの拡張属性は、オブジェクトが存在している間だけ存在します。

拡張属性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 つの拡張属性については、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拡張属性にも使用されます。

名前の最初の文字としてのピリオド (.) は、この拡張属性が標準システム拡張属性 (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。

拡張属性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 つの事前定義拡張属性をサポートします。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、どのような拡張属性でも付けることができます。フォルダーおよび文書には、拡張属性を付けても、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拡張属性を付けることができます。また、拡張属性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) コマンドおよびオブジェクト・リンクの表示 (DSPLNK) コマンドを使用して、オブジェクトの .SUBJECT 拡張属性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拡張属性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拡張属性を変更することができます。

クライアント PC が OS/2 または Windows オペレーティング・システムを介して System i プラットフォーム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・オブジェクトの拡張属性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拡張属性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。

- 拡張属性の名前の長さは、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拡張属性は、標準システム拡張属性と解釈されます。
- 名前の競合の可能性を最小限に抑えるために、拡張属性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

i5/OS オペレーティング・システムを使用して、統合ファイル・システムのオブジェクトをスキャンすることができます。

このサポートによってユーザーに柔軟性が提供され、ユーザーはさまざまな項目をスキャンし、スキャンをいつ実行するかを決定し、スキャン結果に応じてどんなアクションをとるかを決定することができます。

このサポートに関連して、以下の 2 つの出口点が提供されます。

- 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。

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。

関連タスク

157 ページの『オブジェクトをスキャンするかどうかの設定』
「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムで、オブジェクトをスキャンするかどうかを指定できます。以下のステップに従って、スキャン・オプションを設定します。

関連情報

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

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

例: ウィルスとオープンされるファイルのスキャン

これらの例には、スキャンの対象にできる出口プログラムが示されてます。

- 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。 System i プラットフォーム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、システム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を実行することができます。 以下のリストは、スキャンを実行する時点に応じた異なるタイプのスキャンを説明しています。

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

   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。ファイルが PC からオープンされる時に、そのファイルがスキャンされます。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です。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。出口プログラムがファイルを修復した後では、そのファイルにはもう波及しません。このようにして、PC からのアクセスにも波及しないし、その波及が広がることもありません。

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。その後で、感染したファイルに PC からアクセスすると、ウィルスがその PC に波及することがあります。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。

この方式の大きな欠点は、スキャンを実行するためにリソース時間が必要になることです。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、アクセスの度ではなく、必要な場合に限りスキャンを実行するようにします。

2. 一括起動または手動起動のスキャン

   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できます。この場合、週末などシステムがオフライン状態になる時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時、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ます。したがって、大量スキャンの完了後に変更を行わないファイルでは、実行時スキャンのオーバーヘッドを削減できます。このようなファイルに再びアクセスする時には、再スキャンが不要になるためです。

関連概念

『関連するシステム値』
QSCANFS および QSCANFSCTL のシステム値を使用して、ご使用のシステムに適したスキャン環境を設定することができます。

関連情報

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

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

関連するシステム値

QSCANFS および QSCANFSCTL のシステム値を使用して、ご使用のシステム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、スキャン関連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、System i 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、System i ナビゲーター 名の後の括弧内に示されます。例えばシステム値 QSCANFSCTL の場合、System i 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。

これらのシステム値の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。 System i ナビゲーター のシステム値に基づい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、以下に記載されています。

- 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 System i プラットフォーム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。

関連情報

セキュリティー・システム値: 登録済み出口プログラムを使用したルート (/)、QOpenSys、およびユーザー定義のファイル・システムのスキャン

セキュリティー・システム値: スキャン制御

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。以下には、どんな時に、どんな理由でスキャンを実行するかについての情報が示されています。

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、オブジェクト・リンクの表示 (DSPLNK) コマンド、属性の取得 (Qp0lGetAttr()) API、または System i ナビゲーター の「プロパティー」ページを使用することができます。

関連情報

オブジェクト・リンクの処理 (WRKLNK) コマンド

オブジェクト・リンクの表示 (DSPLNK) コマンド

Qp0lGetAttr()--Get Attributes API

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。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。

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。

関連情報

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

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

スキャン・シグニチャーの変更 (QP0LCHSG) API

異なる CCSID:

以前にオブジェクトをスキャンした時とは異なるコード化文字セット ID (CCSID) でそのオブジェクトがアクセスされた場合は、スキャンが起動されます。

このスキャンの例は、CCSID 819 で保管されたデータのあるファイルが CCSID 1200 でオープンされて、正常にスキャンされる時です。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。

保管操作の時:

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。

オブジェクトの保管 (SAV) コマンドには、ファイルの保管時にスキャンするかどうかを指定できる SCAN パラメーターが組み込まれました。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。

関連情報

オブジェクトの保管 (SAV) コマンド

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する場合には、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。

関連情報

オブジェクト保全性の変更 (CHGOBJITG) コマンド

ファイル・システム

ファイル・システムは、論理単位として編成された記憶域の特定のセグメントへのアクセスを提供します。システムの論理単位とは、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 i5/OS 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 i5/OS 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。

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・プログラミング・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。

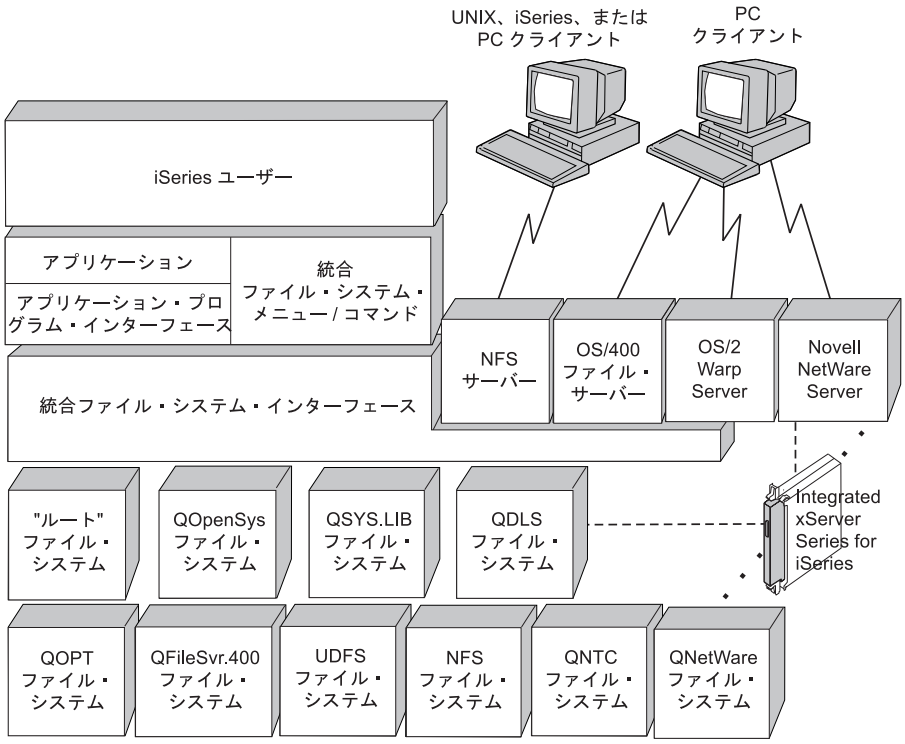

図 9. ファイル・システム、ファイル・サーバー、および統合ファイル・システム・インターフェース

統合ファイル・システム・インターフェースを介したネットワーク・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 (NFS) は、統合ファイル・システムのインターフェースを介してアクセス可能となります。これらの考慮事項および制限事項に注意してください。

関連情報

光ディスク・ストレージ

 i5/OS ネットワーク・ファイル・システム・サポート PDF

統合ファイル・システムのセキュリティーの計画

統合ファイル・システムのセキュリティーの計画

ファイル・システムの比較

これらの表は、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
i5/OS の標準機能	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)	はい (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=128 GB	T2=1 TB、T1=128 GB	データベース・ファイル・サイズ	4 GB	不定 [17]
i5/OS ファイル・サーバーによるアクセス	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)
ファイル・サーバー・プロセッサー による直接アクセス [1]	いいえ (No)	いいえ (No)	いいえ (No)	いいえ (No)	はい (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	いいえ (No)	はい (Yes)	いいえ (No) [4]	いいえ (No) [5]	いいえ (No)
パス名の各構成要素の最大長	255 文字 [3]	255 文字 [3]	10/6 文字 [6]	8/3 文字 [7]	255 文字 [3]
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	はい (Yes)	はい (Yes)	いいえ (No)	いいえ (No)	いいえ (No)
オブジェクト / ファイルの所有者の設定	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)	いいえ (No)
統合ファイル・システム・コマンドのサポート	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)
統合ファイル・システム API のサポート	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)
階層ファイル・システム (HFS) API のサポート	いいえ (No)	いいえ (No)	いいえ (No)	はい (Yes)	いいえ (No)
スレッド・セーフ [13]	はい (Yes)	はい (Yes)	はい (Yes)	いいえ (No)	はい (Yes)
オブジェクト・ジャーナル処理のサポート	はい (Yes)	はい (Yes)	はい (Yes)[14]	いいえ (No)	いいえ (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・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. 速度は、i5/OS ファイル・サーバーを介して、ファイル・システムにアクセスする場合に適用されます。
3. 特定の CCSID 値では、最大長を 255 文字以下にすることができます。
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。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。
5. 詳しくは、 54 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. その値は、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。
7. その値は、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。
8. 値は、絶対パス名が / で始まり、その後にファイル・システム名が続く (/QDLS... など) という前提に基づいています。 .
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. 例外は、別のディレクトリーとのリンクを 1 つだけ設定できるディレクトリーの場合です。
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・システム中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・システム以外のさまざまなオブジェクト・タイプのジャーナル処理をサポートします。
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。
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. この制限は、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。

省略形

- 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. ファイル・システムの要約 (2/2)

機能	QOPT	QFileSvr.400	UDFS	NFS
i5/OS の標準機能	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)
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
ファイル・サイズの制限	4 GB	不定 [3]	T2=1 TB、T1=128 GB	不定 [14]
i5/OS ファイル・サーバーによるアクセス	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)

表 3. ファイル・システムの要約 (2/2) (続き)

機能	QOPT	QFileSvr.400	UDFS	NFS
ファイル・サーバー・プロセッサー による直接アクセス [1]	いいえ (No)	いいえ (No)	いいえ (No)	いいえ (No)
オープン / クローズの相対的な速度	低	低 [2]	中 [2]	中 [2]
英語の大文字小文字を区別した名前の検索	いいえ (No)	いいえ (No) [2]	はい (Yes) [11]	不定 [2]
パス名の各構成要素の最大長	不定 [4]	不定 [2]	255 文字 [15]	不定 [2]
パス名の最大長 [5]	294 文字	制限なし [2]	16 MB	制限なし [2]
オブジェクトの拡張属性の最大長	8 MB	0 [6]	2 GB [10]	0 [6]
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]
オブジェクトごとのリンクの最大数 [8]	1	1	不定 [13]	不定 [2]
シンボリック・リンクのサポート	いいえ (No)	いいえ (No)	はい (Yes)	はい (Yes) [2]
オブジェクト / ファイルの所有者の設定	いいえ (No)	いいえ (No) [9]	はい (Yes)	はい (Yes) [2]
統合ファイル・システム・コマンドのサポート	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)
統合ファイル・システム API のサポート	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)
階層ファイル・システム (HFS) API のサポート	はい (Yes)	いいえ (No)	いいえ (No)	いいえ (No) [2]
スレッド・セーフ [12]	はい (Yes)	はい (Yes)	はい (Yes)	はい (Yes)
オブジェクト・ジャーナル処理のサポート	いいえ (No)	いいえ (No)	はい (Yes)	いいえ (No)

表 3. ファイル・システムの要約 (2/2) (続き)

機能	QOPT	QFileSvr.400	UDFS	NFS

注:

1. ファイル・サーバー・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. この値は、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. V6R1 より前のシステムに接続する場合、ファイル・サイズ制限は 2 GB-1 です。その他の場合、ファイル・サイズ制限はアクセス対象のファイル・システムに応じて異なります。
4. 詳しくは、 57 ページの『光ファイル・システム (QOPT)』を参照してください。
5. 値は、絶対パス名が / で始まり、その後にファイル・システム名が続くという前提に基づいています。
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. 例外は、別のディレクトリーとのリンクを 1 つだけ設定できるディレクトリーの場合です。
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. 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。
12. 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。
13. *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。
14. この制限は、アクセス対象のシステムに応じて異なります。
15. 特定の CCSID 値では、最大長を 255 文字以下にすることができます。

省略形

- 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

関連資料

『「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。このシステムには、DOS および OS/2 のファイル・システムの特性があります。

35 ページの『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 X/Open Portability Guide (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。

37 ページの『ユーザー定義ファイル・システム (UDFS)』
ユーザー定義ファイル・システム (UDFS) 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。

47 ページの『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、i5/OS・ライブラリー構造をサポートします。

50 ページの『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 i5/OS・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 i5/OS・オブジェクト・タイプへのアクセスを提供します。

54 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)』
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。

57 ページの『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。

59 ページの『i5/OS NetClient ファイル・システム (QNTC)』
QNTC ファイル・システムは、 Integrated xSeries Server (IXS)を実行中の Windows NT 4.0 またはそれ以降、あるいは Linux® オペレーティング・システムに保管されているデータおよびオブジェクトへのアクセスを提供します。 また、QNTC ファイル・システムは Windows NT 4.0 またはそれ以降、Linux Samba 3.0 またはそれ以降、あるいは i5/OS NetServer のサポートされるバージョンを実行しているリモート・サーバーに保管されたデータおよびオブジェクトへのアクセスも提供します。

65 ページの『i5/OS ファイル・サーバー・ファイル・システム (QFileSvr.400)』
QFileSvr.400 ファイル・システムは、リモート System i プラットフォーム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。

70 ページの『ネットワーク・ファイル・システム (NFS)』
ネットワーク・ファイル・システム (NFS) 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。

関連情報

ジャーナル管理

「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。このシステムには、DOS および OS/2 のファイル・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 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・システムへのアクセス

「ルート」(/) ファイル・システムにアクセスするには、i5/OS 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。

「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、システム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

「ルート」 (/) ファイル・システムでは、パス名は固有の形式となります。

```
/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、プログラムおよびシステム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。

関連概念

17 ページの『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。

9 ページの『*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。

15 ページの『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システムに指示します。

「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。

関連概念

11 ページの『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、システム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。

「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

「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、「メニューおよび表示画面を使用したアクセス」に記述してある表示画面が「ルート」(/) ファイル・システムで使用できます。ただし、マルチスレッド可能プロセスでは、これらのコマンドを使用することは安全でない可能性があります。

関連タスク

77 ページの『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、システム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。

関連資料

79 ページの『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。

「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

「API を使用した操作の実行」にリストされたすべての API は、「ルート」 (/) ファイル・システムで使用できます。

関連資料

132 ページの『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多くのアプリケーション・プログラミング・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。

関連情報

アプリケーション・プログラミング・インターフェース (API)

「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・システムのオブジェクト・タイプの中には、ジャーナル処理できるものがあります。この機能によって、オブジェクトの最後の保管以降にそのオブジェクトに対して行った変更を回復することができます。

関連概念

115 ページの『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。

「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。 ユーザー・データグラム・プロトコル (UDP)、および伝送制御プロトコル (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・メジャー番号を受け取ります。さらに、どちらもクローン・オープン操作が可能です。つまり、それぞれのオープン操作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がサポートされるのは、i5/OS ポータブル・アプリケーション・ソリューション環境 (PASE) の場合に限られます。 以下の表には、作成されるオブジェクトと、それらのプロパティーが示されています。

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

関連情報

i5/OS PASE

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 X/Open Portability Guide (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 へのアクセス

QOpenSys にアクセスするには、i5/OS 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。

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、オブジェクト名の検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

QOpenSys ファイル・システムでは、パス名は固有の形式となります。

```
/QOpenSys/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 MB 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、プログラムおよびシステム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。

関連概念

17 ページの『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。

9 ページの『*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。

15 ページの『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システムに指示します。

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。

関連概念

11 ページの『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、システム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。

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、「CL コマンドを使用したアクセス」トピックにリストしてあるすべてのコマンドと、「メニューおよび表示画面を使用したアクセス」トピックに記述してある表示画面を操作できます。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。

関連タスク

77 ページの『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、システム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。

関連資料

79 ページの『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。

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、「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。

関連資料

132 ページの『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多くのアプリケーション・プログラミング・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。

関連情報

アプリケーション・プログラミング・インターフェース (API)

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・システムのオブジェクト・タイプの中には、ジャーナル処理できるものがあります。この機能によって、オブジェクトの最後の保管以降にそのオブジェクトに対して行った変更を回復することができます。

関連概念

115 ページの『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。

ユーザー定義ファイル・システム (UDFS)

ユーザー定義ファイル・システム (UDFS) 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

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。
- 制限された名前変更およびリンク解除の属性値
- UDFS 内に作成されるオブジェクトの監査値
- UDFS 内に作成されるストリーム・ファイルの異なるフォーマットである *TYPE1 と *TYPE2

- UDFS 内に作成されるストリーム・ファイルのディスク・ストレージ・オプション
- UDFS 内に作成されるストリーム・ファイルの主ストレージ・オプション

ユーザー定義ファイル・システムの概念

ユーザー定義ファイル・システム (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。

注: ブロック特殊ファイルの属性や権限を変更すると、同等の変更が UDFS のルート・ディレクトリーに反映されます。その逆も同様です。これらの変更により、UDFS 内の他のオブジェクトが影響を受けることはありません。

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。

システムの初期プログラム・ロード (IPL)、またはディレクトリーの記憶域再利用 (RCLSTG) 操作で UDFS がすべてアンマウントされます。したがって、IPL の実行やディレクトリーに対する RCLSTG コマンドの実行後には、UDFS の再マウントが必要となります。

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。

統合ファイル・システム・インターフェースを介したユーザー定義ファイル・システムへのアクセス

ユーザー定義ファイル・システム (UDFS) にアクセスするには、i5/OS 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。

関連概念

11 ページの『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、システム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。

16 ページの『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。

関連情報

ユーザー定義ファイル・システムの作成 (CRTUDFS) コマンド

一時ユーザー定義ファイル・システム

一時ユーザー定義ファイル・システムは、補助記憶域操作を削減することにより、パフォーマンスを向上させることができます。

統合ファイル・システムは、システムの IPL 時や異常終了時にファイルおよびディレクトリーが保持されるように、補助記憶域操作を行います。しかし、多くのアプリケーションは、システムの IPL 間で保持する必要がない一時作業ファイルおよびディレクトリーを使用します。このようなアプリケーションは、オブジェクトを強制的に永続ストレージに入れることによって不必要にスローダウンします。ユーザーは、一時オブジェクトのみを含む特殊なタイプの UDFS を作成しマウントすることができます。一時オブジェクトは、システムの再始動時やファイル・システムのアンマウント時にシステムにより自動的に削除されるため、余分な補助記憶域操作を必要としません。

一時ファイル・システムは、「ユーザー定義ファイル・システムの作成 (CRTUDFS)」コマンドによって、新しい命名規則を使用して作成されます。永続 UDFS の名前は、「.udfs」で終わる必要があります。一時 UDFS の名前は、「.tmpudfs」で終わる必要があります。例えば、次のようになります。

```
CRTUDFS UDFS('/dev/QASP01/new.tmpudfs')
```

一時 UDFS の作成者は、 *ALLOBJ 特殊権限を持っている必要があります。一時 UDFS は、システム補助記憶域プール (ASP) 内にのみ作成できます。つまり、一時 UDFS は、'/dev/QASP01' 内にのみ作成できます。

一時 UDFS は、作成後にマウントして、永続 UDFS のように使用することができます。ただし、以下の制約事項があります。

- 一時オブジェクトは、権限リストによって保護できません。
- 一時オブジェクトのユーザー・ジャーナリングは許可されません。
- 一時ファイル・システムからオブジェクトを保管することはできません。また、一時ファイル・システムにオブジェクトを復元することもできません。
- 拡張属性は、一時オブジェクトではサポートされません。
- 一時オブジェクトのオブジェクト署名は許可されません。
- 一時 UDFS を読み取り専用ファイル・システムとしてマウントすることはできません。

一時ファイル・システムのユーザーは、以下の考慮事項をよく理解しておく必要があります。

- 一時オブジェクト用のストレージは、所有するユーザー・プロファイルにも、いずれかのプロセスにも帰属しません。そのため、ユーザー・プロファイルで、そのユーザー・プロファイルに対して許可されている最大ストレージを超える一時オブジェクトを作成することができます。

- ファイル・システム (システムによって提供される QDLS.LIB や QSYS.LIB などのファイル・システムも含む) がアンマウントされると、マウントされていたファイル・システムにアクセスしようとしている他のプロセスが遅延する可能性があります。このような遅延は、通常、短時間で気付かない程度のものです。一時 UDFS がアンマウントされると、その UDFS のすべての内容が削除されます。一時 UDFS に大量のオブジェクトがある場合、他のプロセスで、他のファイル・システムへのアクセスにかなりの遅延が発生する可能性があります。この考慮事項による影響が考えられる場合には、ファイル・システムをアンマウントする前に、必要に応じてディレクトリー除去 (RMVDIR) コマンドおよびリンク除去 (RMVLNK) コマンドを使用して、一時 UDFS の内容を削除することをお勧めします。この場合には、RMVDIR コマンドの SUBTREE(*ALL) 値が特に役立ちます。
- 一時 UDFS をアンマウントすると UDFS の内容がすべて削除されるため、アンマウント操作は、ユーザー定義ファイル・システムの削除 (DLTUDFS) コマンドの制約事項のうちの多くの制約事項と同じ制約を受けます。アンマウントは、以下のいずれかが真である場合に失敗する可能性があります。
  - ユーザーが UDFS 内の一部のオブジェクトに対する *OBJEXIST 権限しか持っていない。
  - ユーザーが UDFS 内の空でない一部のディレクトリーに対する *WX データ権限しか持っていない。
  - UDFS 内のいずれかのオブジェクトがチェックアウトされている。
  - UDFS 内のいずれかのオブジェクトが読み取り専用である。
  - ディレクトリーの「制限された名前変更およびリンク解除」属性が「はい」に設定されいる。なおかつ、ユーザーはそのディレクトリーの所有者でなく、ディレクトリー内のすべてのオブジェクトを所有しておらず、 *ALLOBJ 特殊権限を持っていない。

  これらのいずれかの理由によりアンマウントが失敗すると、失敗の理由を示す診断メッセージが、削除されたオブジェクト数を示すメッセージと共にジョブ・ログに書き込まれます。
- 一時 UDFS に関連したブロック特殊ファイル (BLKSF) オブジェクト自体は、永続オブジェクトです。一時的なのは、UDFS の内容のみです。ただし、この BLKSF は保管できません。また、保管できるように属性を変更することもできません。また、この BLKSF は、権限リストによって保護できません。

統合ファイル・システムのユーザー定義ファイル・システムの大/小文字の区別

ユーザー定義ファイル・システム (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認識されます。大/小文字の区別 UDFS を作成するには、ユーザー定義ファイル・システムの作成 (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、システム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、システムは /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、システム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。

関連情報

ユーザー定義ファイル・システムの作成 (CRTUDFS) コマンド

統合ファイル・システムのユーザー定義ファイル・システムのパス名

ブロック特殊ファイル (*BLKSF) は、ユーザー定義ファイル・システム (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 上に存在し、一時 UDFS でない場合、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 が一時 UDFS である場合、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/QASP01/udfs_name.tmpudfs`

| ここで、`udfs_name` は UDFS の固有の名前です。一時 UDFS はシステム ASP 上にのみ存在することができるため、ブロック特殊ファイルは `/dev/QASP01` にのみ作成できます。一時 UDFS 名が必ず `.tmpudfs` 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 MB 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。

関連概念

17 ページの『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。

9 ページの『*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。

15 ページの『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システムに指示します。

統合ファイル・システムのユーザー定義ファイル・システムのリンク

ユーザー定義ファイル・システム (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。

関連概念

11 ページの『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、システム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。

ユーザー定義ファイル・システムでの統合ファイル・システム・コマンドの使用

ユーザー定義ファイル・システムでは、「CL コマンドを使用したアクセス」トピックにリストしてあるすべてのコマンドと、「メニューおよび表示画面を使用したアクセス」トピックに記述してある表示画面を操作できます。

UDFS 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 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。

表 5. ユーザー定義ファイル・システム CL コマンド (続き)

コマンド	説明
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。

関連タスク

77 ページの『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、システム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。

関連資料

79 ページの『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。

ユーザー定義ファイル・システムでの統合ファイル・システム API の使用

ユーザー定義ファイル・システムでは、「API を使用した操作の実行」トピック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。

関連資料

132 ページの『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多くのアプリケーション・プログラミング・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。

関連情報

アプリケーション・プログラミング・インターフェース (API)

ユーザー定義ファイル・システムのグラフィカル・ユーザー・インターフェース

| Systems Director Navigator for i (PC 上のグラフィカル・インターフェース) により、ユーザー定義ファイル・システム (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。

| Systems Director Navigator for i 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。

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

- 156 ページの『ユーザー定義ファイル・システムのアンマウント』

統合ファイル・システムのユーザー定義ファイル・システムの作成

ユーザー定義ファイル・システムの作成 (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、ユーザー定義ファイル・システム (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。

- 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。
- 制限された名前変更およびリンク解除の属性値
- UDFS 内に作成されるオブジェクトの監査値
- UDFS 内に作成されるストリーム・ファイルの異なるフォーマットである *TYPE1 と *TYPE2
- UDFS 内に作成されるストリーム・ファイルのディスク・ストレージ・オプション
- UDFS 内に作成されるストリーム・ファイルの主ストレージ・オプション

関連情報

ユーザー定義ファイル・システムの作成 (CRTUDFS) コマンド

マウント・ファイル・システムの追加 (ADDMFS) コマンド

統合ファイル・システムのユーザー定義ファイル・システムの削除

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ユーザー定義ファイル・システム (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。

関連情報

ユーザー定義ファイル・システムの削除 (DLTUDFS) コマンド

統合ファイル・システムのユーザー定義ファイル・システムの表示

ユーザー定義ファイル・システムの表示 (DSPUDFS) コマンドは、既存のユーザー定義ファイル・システム (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。

関連情報

ユーザー定義ファイル・システムの表示 (DSPUDFS) コマンド

マウント・ファイル・システムの情報の表示 (DSPMFSINF) コマンド

統合ファイル・システムのユーザー定義ファイル・システムのマウント

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。

ユーザー定義ファイル・システム (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。

システムの初期プログラム・ロード (IPL)、またはディレクトリーの記憶域再利用 (RCLSTG) 操作で UDFS がすべてアンマウントされます。したがって、IPL の実行やディレクトリーに対する RCLSTG コマンドの実行後には、UDFS の再マウントが必要となります。

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。

関連情報

マウント・ファイル・システムの追加 (ADDMFS) コマンド

統合ファイル・システムのユーザー定義ファイル・システムのアンマウント

アンマウント・コマンドは、ユーザー定義ファイル・システム (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。

注: システムの初期プログラム・ロード (IPL)、またはディレクトリーの記憶域再利用 (RCLSTG) 操作で UDFS もすべてアンマウントされます。

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 (例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。

関連情報

マウント・ファイル・システムの除去 (RMVMFS) コマンド

統合ファイル・システムのユーザー定義ファイル・システムの保管および復元

UDFS が一時 UDFS でない限り、ユーザー定義ファイル・システム (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。

オブジェクトの保管 (SAV) コマンドによって UDFS 内のオブジェクトを保管できます。また、オブジェクトの復元 (RST) コマンドによって UDFS オブジェクトを復元できます。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。

関連情報

オブジェクトの保管 (SAV) コマンド

オブジェクトの復元 (RST) コマンド

ユーザー定義ファイル・システムでのオブジェクト変更のジャーナル処理

ユーザー定義ファイル・システム (UDFS) のオブジェクト・タイプの中には、ジャーナル処理できるものがあります。この機能によって、オブジェクトの最後の保管以降にそのオブジェクトに対して行った変更を回復することができます。

関連概念

115 ページの『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。

ユーザー定義ファイル・システムと独立補助記憶域プール

独立補助記憶域プール (ASP) をオンに変更すると、「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。

これらの変更は次のとおりです。

- /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。

  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。

関連資料

50 ページの『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 i5/OS・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 i5/OS・オブジェクト・タイプへのアクセスを提供します。

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、i5/OS・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補助記憶域プール (ASP) 内で管理する、他のすべての i5/OS・オブジェクト・タイプへのアクセスを提供します。

さらに、

- i5/OS・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。
- i5/OS・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 における統合ファイル・システムの導入前は、QSYS.LIB ファイル・システムが唯一のファイル・システムでした。 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実行する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基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。

統合ファイル・システム・インターフェースを介した QSYS.LIB へのアクセス

QSYS.LIB ファイル・システムにアクセスするには、i5/OS 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。

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

QSYS.LIB ファイル・システムでファイルを処理する時に注意する制約事項は次の通りです。

- 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、『fcntl()--ファイル制御コマンドの実行 (Perform File Control Command)』 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。

例えば、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。

例えば、既存の保存ファイルには、異なる既存の空の保存ファイル・オブジェクトにデータを移動させることが必要になるまで、読み取りや別のファイルへのコピーが可能なデータが入っています。保存ファイルが書き込みのためにオープンされている場合、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。保存ファイルでは、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。保存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場合、保存ファイル・データへのストリーム入出力操作を実行することはできません。

保存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・システムを介してエクスポートされる場合には、保存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。

- 例えば、

  ```
  /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 の詳細については、 i5/OS グローバリゼーション のトピックを参照してください。

関連概念

15 ページの『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システムに指示します。

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ライブラリー中のオブジェクトとの関係は、ライブラリーとそのライブラリー中の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が可能です。

関連概念

11 ページの『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、システム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。

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、多くの統合ファイル・システム・コマンドおよび表示画面が有効となります。

QSYS.LIB ファイル・システムでは、 79 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、以下の制約があります。

- リンクの追加 (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。
- ジャーナル開始 (STRJRN) およびジャーナル終了 (ENDJRN) コマンドは、データベース物理ファイルやライブラリーに対して使用できません。
- 次のコマンドは、サポートされていません。
  - オブジェクトのチェックイン (CHKIN)
  - オブジェクトのチェックアウト (CHKOUT)
  - オブジェクト・リンクの再利用 (RCLLNK)

77 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、多くの統合ファイル・システム API が有効となります。

QSYS.LIB ファイル・システムでは、 132 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 が使用できますが、以下の制限があります。

- 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、またはライブラリーに対しては、QjoStartJournal() および QjoEndJournal() API を使用できません。

関連情報

アプリケーション・プログラミング・インターフェース (API)

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 i5/OS・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 i5/OS・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 i5/OS・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。
- i5/OS・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。

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 へのアクセス

独立 ASP QSYS.LIB ファイル・システムにアクセスするには、i5/OS 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。

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、幾つか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

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限事項

独立 ASP QSYS.LIB ファイル・システムでファイルを処理する時に注意する制約事項は次の通りです。

- 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、『fcntl()--ファイル制御コマンドの実行 (Perform File Control Command)』 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。

例えば、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。

例えば、既存の保存ファイルには、異なる既存の空の保存ファイル・オブジェクトにデータを移動させることが必要になるまで、読み取りや別のファイルへのコピーが可能なデータが入っています。保存ファイルが書き込みのためにオープンされている場合、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。保存ファイルでは、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。保存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場合、保存ファイル・データへのストリーム入出力操作を実行することはできません。

保存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・システムを介してエクスポートされる場合には、保存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。

- 例えば、

  ```
  /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、名前が保管される際に、コード化文字セット ID (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 の詳細については、i5/OS Information Center の 「i5/OS グローバリゼーション」のトピックを参照してください。

関連概念

15 ページの『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システムに指示します。

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ライブラリー中のオブジェクトとの関係は、ライブラリーとそのライブラリー中の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。

関連概念

11 ページの『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、システム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。

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

統合ファイル・システムの多くのコマンドおよび表示が独立 ASP QSYS.LIB ファイル・システムで有効となります。

独立 ASP QSYS.LIB ファイル・システムでは、 79 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が使用できます。ただし、次の例外があります。

- リンクの追加 (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。
- ジャーナル開始 (STRJRN) およびジャーナル終了 (ENDJRN) コマンドは、データベース物理ファイルやライブラリーに対して使用できません。
- オブジェクトの移動 (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。
- オブジェクトの保存 (SAV) またはオブジェクトの復元 (RST) を使用して独立 ASP 上にライブラリー・オブジェクトを保存または復元する場合、その独立 ASP が SAV または RST 操作を実行する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。
- 次のコマンドは、サポートされていません。
  - オブジェクトのチェックイン (CHKIN)
  - オブジェクトのチェックアウト (CHKOUT)
  - オブジェクト・リンクの再利用 (RCLLNK)

77 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、多くの統合ファイル・システム API が有効となります。

独立 ASP QSYS.LIB ファイル・システムでは、 132 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 が使用できますが、以下の状態は除きます。

- 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、またはライブラリーに対しては、QjoStartJournal() および QjoEndJournal() API を使用できません。
- QsrSave() または QsrRestore() API 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は、この独立 ASP が保管または復元操作を行うジョブと関連しているか、またはその独立 ASP が ASPDEV キーに指定されていなければなりません。パス名の命名規則 (/asp_name/QSYS.LIB/object.type) は、QsrSave() および QsrRestore() API ではサポートされていません。

関連情報

アプリケーション・プログラミング・インターフェース (API)

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、

- i5/OS 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。

統合ファイル・システム・インターフェースを介した QDLS へのアクセス

QDLS ファイル・システムにアクセスするには、i5/OS 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。

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 Integrated Language Environment® (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 System i 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。

関連情報

階層ファイル・システム API

QDLS ファイル・システムでのユーザー登録

QDLS ファイル・システム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 ファイル・システムでは、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。

関連情報

フォルダーおよび文書名

QDLS ファイル・システムでのパス名

パス名の各構成要素は、1 つの名前のみで構成できます。

- 例えば、

  ```
  /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)。
- 文書ライブラリー・サービス (QDLS) ファイル・システム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・ページの詳細については、i5/OS Information Center の 「i5/OS グローバリゼーション」のトピックを参照してください。

関連概念

15 ページの『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システムに指示します。

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。

関連概念

11 ページの『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、システム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。

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、多くの統合ファイル・システム・コマンドおよび表示画面が有効になります。

QDLS ファイル・システムでは、 79 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、以下のコマンドを除きます。

- 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、文書に対してはサポートされていますが、フォルダーに対してはサポートされていません。
- 次のコマンドは、サポートされていません。
  - APYJRNCHG
  - CHGJRNOBJ
  - DSPJRN
  - ENDJRN
  - RCLLNK
  - RCVJRNE
  - RTVJRNE
  - SNDJRNE
  - STRJRN

77 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、多くの統合ファイル・システム API が有効になります。

QDLS ファイル・システムでは、 132 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。

- 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()
- QjoRetrieveJournalEntries()
- QjoRetrieveJournalInformation()
- QJORJIDI()
- QJOSJRNE()
- QjoStartJournal()
- readlink()
- takedescriptor()

関連情報

アプリケーション・プログラミング・インターフェース (API)

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。

統合ファイル・システムを介した QOPT へのアクセス

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。

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。

関連情報

光ディスク・ストレージ

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 System i 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 の詳細については、『光ディスク装置プログラミング』トピック・コレクションを参照してください。

関連情報

階層ファイル・システム API

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。

- 例えば、

  ```
  /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、『パス名 (Path names)』 の 『パス名規則 (Path Name Rules)』 を参照してください。

関連概念

15 ページの『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システムに指示します。

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」(/)、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。

関連概念

11 ページの『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、システム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。

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、多くの統合ファイル・システム・コマンドおよび表示画面が有効になります。

QOPT ファイル・システムでは、 79 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。 77 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
- DSPJRN
- ENDJRN
- RCLLNK
- RCVJRNE
- RTVJRNE
- 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

QOPT ファイル・システムでは、多数の統合ファイル・システム API が有効になります。

QOPT ファイル・システムでは、 132 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、以下の API を除きます。

- QjoEndJournal()
- QjoRetrieveJournalEntries()
- QjoRetrieveJournalInformation()
- QJORJIDI()
- QJOSJRNE()
- QjoStartJournal()
- Qp0lGetPathFromFileID()

関連情報

アプリケーション・プログラミング・インターフェース (API)

i5/OS NetClient ファイル・システム (QNTC)

QNTC ファイル・システムは、 Integrated xSeries Server (IXS)を実行中の Windows NT 4.0 またはそれ以降、あるいは Linux オペレーティング・システムに保管されているデータおよびオブジェクトへのアクセスを提供します。 また、QNTC ファイル・システムは Windows NT 4.0 またはそれ以降、Linux Samba

3.0 またはそれ以降、あるいは i5/OS NetServer のサポートされるバージョンを実行しているリモート・サーバーに保管されたデータおよびオブジェクトへのアクセスも提供します。

QNTC ファイル・システムは基本 i5/OS 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、統合サーバー・サポート (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。

統合ファイル・システム・インターフェースを介した QNTC へのアクセス

i5/OS NetServer、System i 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小文字の別を、入力されたのと同じ状態のまま保持しますが、QNTC ファイル・システム自体で名前の大文字と小文字の区別はしません。そのため、サーバーのファイル・システムが大/小文字の区別をする場合には、QNTC ファイル・システムでパス名の大/小文字を適切に指定する必要があります。

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

パスは、ファイル・システム名、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。

パス名の要件は以下のとおりです。

- 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。パス名の形式は、次のとおりです。

  ```
      /QNTC/Servername/Sharename/Directory/ . . . /Object
  (QNTC はパス名の必須部分です。). . /Object(QNTC はパス名の必須部分です。)
  ```

- サーバー名は QNTC パス名の必須部分です。サーバー名には、TCP/IP ホスト名、NetBIOS 名、または TCP/IP アドレスを設定できます。 V6R1 より、IPv4 アドレスに加えて IPv6 アドレスもサポートされています。
- 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。
- デフォルトで、ローカル・サブネットで機能している、サポートされる各サーバーは、/QNTC の下のディレクトリーとして自動的に表示されます。ディレクトリーの作成 (CRTDIR) コマンドまたは mkdir() API を使用して、ローカル・サブネットの外にあるアクセス可能システムを追加します。

関連概念

15 ページの『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システムに指示します。

関連情報

ディレクトリーの作成 (MKDIR) コマンド

mkdir()--Make Directory API

i5/OS 用語集

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。

関連概念

11 ページの『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、システム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。

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、多くの統合ファイル・システム・コマンドおよび表示画面が有効になります。

QNTC ファイル・システムでは、 79 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、以下のコマンドを除きます。

- ADDLNK
- APYJRNCHG
- CHGJRNOBJ
- CHGOWN
- CHGAUT
- CHGPGP
- CHKIN
- CHKOUT
- DSPAUT
- DSPJRN
- ENDJRN
- RCLLNK

- RCVJRNE
- RTVJRNE
- 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

77 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、多くの統合ファイル・システム API が有効になります。

QNTC ファイル・システムでは、132 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。

- 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()
  - fclear()
  - fclear64()
  - givedescriptor()
  - link()
  - QjoEndJournal()
  - QjoRetrieveJournalEntries()
  - QjoRetrieveJournalInformation()
  - QJORJIDI()
  - QJOSJRNE()
  - QjoStartJournal()
  - Qp0lGetPathFromFileID()
  - readlink()
  - symlink()
  - takedescriptor()
- QNTC ファイル・システムにおいて、setrlimit() API を使用して設定されたリソース制限は、以下の関数の実行時に無視されます。
  - write()
  - writev()
  - pwrite()

– pwrite64()

関連情報

アプリケーション・プログラミング・インターフェース (API)

QNTC 環境変数

QNTC のネットワーク・ブラウズ動作は 2 つの環境変数によって制御することができます。 これらの環境変数のサポートは i5/OS V5R4 から開始されました。ADDENVVAR CL コマンドを使用して、これらの環境変数を作成します。

QZLC_SERVERLIST

この環境変数が「2」に設定された場合は、統合ファイル・システムの /QNTC ディレクトリー中に現れるすべてのサーバーが QNTC によってアクセス可能になります。V5R4 以前では、これはデフォルトの動作でした。 この環境変数が「2」に設定されなかった場合は、/QNTC ディレクトリー中に現れる一部のサーバーがアクセス可能にならないことがあります。

QIBM_ZLC_NO_BROWSE

この環境変数が「1」に設定された場合は、/QNTC ディレクトリーには、 CRTDIR CL コマンドまたは mkdir() API によって作成したサーバーのみが含まれます。 この環境変数が設定されると、QNTC ファイル・システムに対する多くの操作のパフォーマンスが向上します。ただし、CL コマンドを使用してすべての /QNTC ディレクトリーを作成する必要があります。

QNTC ファイル・システムでのディレクトリーの作成

ディレクトリーの作成 (CRTDIR) コマンド、または mkdir() API を使用して、サーバー・ディレクトリーを /QNTC ディレクトリーに追加できます。

デフォルトで、i5/OS NetServer ドメインおよびローカル・サブネット中のすべての機能サーバーに対して QNTC ディレクトリーが自動的に作成されます。ローカル・サブネットまたは i5/OS NetServer ドメインの外にあるサーバーは、CRTDIR コマンドまたは mkdir() API を使用して追加する必要があります。例えば、

```
CRTDIR '/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。サーバー名には、IPv4 アドレスまたは IPv6 アドレスを指定できます。例えば、

```
CRTDIR '/QNTC/9.130.67.24'
```

または、

```
CRTDIR '/QNTC/2001:0db8:3c4d:0015:0000:0000:abcd:ef12'
```

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:

- i5/OS 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。
- ディレクトリー構造へのディレクトリーの追加に CRTDIR CL コマンドまたは mkdir() API を使用した場合、これらのディレクトリーは、システム IPL の実行後または記憶域の再利用 (RCLSTG) コ

マンド実行後に表示されなくなります。システム IPL を実行した場合、またはディレクトリーに対して RCLSTG コマンドを実行した場合は、その後に、CRTDIR コマンドまたは mkdir() API を再発行する必要があります。

API または CL コマンドを使用してディレクトリーを追加したい場合は、次の例のように、環境変数 QIBM_ZLC_NO_BROWSE を追加してこれらのコマンドのパフォーマンスを向上させることができます。

```
ADDENVVAR ENVVAR(QIBM_ZLC_NO_BROWSE) VALUE(1) LEVEL(*SYS)
```

この環境変数により、ファイル操作の実行時に、ファイル・システムはすべてのネットワーク・ブラウズを迂回することになります。

関連情報

ディレクトリーの作成 (MKDIR) コマンド

mkdir()--Make Directory API

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 (NFS) では、多くの統合ファイル・システム API が有効になります。

ネットワーク・ファイル・システムでは、 132 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、以下の API を除きます。

- mkfifo()
- QjoEndJournal()
- QjoRetrieveJournalEntries()
- 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、 i5/OS 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。

関連情報

アプリケーション・プログラミング・インターフェース (API)

ネットワーク認証サービスで QNTC ファイル・システムを使用可能にする

QNTC ファイル・システムでは、Kerberos V5 認証プロトコルをサポートする共通統合ファイル・システム (CIFS) サーバーに、System i プラットフォーム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、 System i 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。

QNTC で使用するためにネットワーク認証サービス (NAS) を有効にするには、以下の項目を構成する必要があります。

- ネットワーク認証サービス (NAS)
- 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 NAS 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 NAS 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 i5/OS ユーザー・プロファイルで、ローカル・パスワード管理 (LCLPWDMGT) パラメーターを *NO に設定する必要があります。 *NO と指定すると、ユーザーはサーバーへのパスワードを持たず、5250 セッションにサインオンできません。システムにアクセスする唯一の手段は、System i ナビゲーター または System i Access 5250 Display Emulator などの、NAS を使用可能なアプリケーションを介して実行する方法です。

  *YES を指定した場合、パスワードはサーバーによって管理され、ユーザーは NAS を使用せずに認証されます。
- Kerberos チケット、および System i ナビゲーター 接続が必要です。
- ご使用の System i プラットフォーム用の 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。
  1. NAS レルムの KDC 上の「**Active Directory Users and Computers**」ツールにアクセスします。
  2. ユーザーを選択します。
  3. サービス・プリンシパル名に対応する名前を選択します。
  4. 「**プロパティー**」を選択します。
  5. 「**アカウント (Account)**」タブを選択します。
  6. 「**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。

関連情報

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)

i5/OS ファイル・サーバー・ファイル・システム (QFileSvr.400)

QFileSvr.400 ファイル・システムは、リモート System i プラットフォーム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 i5/OS 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。

統合ファイル・システム・インターフェースを介した QFileSvr.400 へのアクセス

QFileSvr.400 ファイル・システムにアクセスするには、i5/OS 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。

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 i5/OS 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。

QFileSvr.400 ファイル・システムでのパス名

QFileSvr.400 ファイル・システムでは、パス名は固有の形式をもちます。

- 形式は次の通りです。

  ```
  /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、以下の両方の属性を表します。
  - 通信の接続を確立するために使用されるターゲット・システムの名前。ターゲット・システムの名前は、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)

      注: ターゲット・システムも V6R1 であれば、ホスト名は IPv4 または IPv6 アドレスへ解決できます。 V6R1 より前のリリースの場合、IPv4 アドレスのみサポートされます。
    - SNA LU 6.2 の名前 (例えば、appn.newyork)
  - ターゲット・システムの「ルート」(/) ディレクトリー

  こうした表記のため、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。

  このファイル・システムを使用するには、第 1 レベル・ディレクトリーを作成する必要があります。そのためには、ディレクトリーを作成する統合ファイル・システム・インターフェースを使用します。

  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。
- 名前に使用されている文字は、名前格納時に Unicode 形式に変換されます。

関連概念

17 ページの『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。

15 ページの『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システムに指示します。

QFileSvr.400 ファイル・システムでの通信

QFileSvr.400 ファイル・システムは以下の方法で通信を行います。

- 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。 LU 6.2 通信の詳細については、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、 i5/OS 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。

     注: ターゲット・システムも V6R1 であれば、リモート・ロケーション名は IPv4 または IPv6 アドレスへ解決できます。 V6R1 より前のリリースの場合、IPv4 アドレスのみサポートされます。
  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 i5/OS 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 i5/OS 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 の名前と想定されます。それから、i5/OS 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。

QFileSvr.400 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

両方のシステムにネットワーク認証サービスおよびエンタープライズ識別マッピング (EIM) が構成されており、ユーザーが Kerberos に認証された場合は、Kerberos を使用して、ターゲット System i プラットフォーム上に存在するファイル・システムにアクセス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・システムがアクセスを検査した後で、発券許可証または System i チケットの有効期限が切れた場合、ターゲット・システム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 を使って認証しない場合、ターゲットの System iプラットフォーム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・システムのユーザー ID およびパスワードに一致する、ターゲット・システムのユーザー 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・システムまたはターゲット・システムのパスワードが変更された場合、ターゲット・システム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・システムのユーザー・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・システム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・システム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・システムのファイル・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・システムのユーザー・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。

関連情報

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)

QFileSvr.400 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。

関連概念

11 ページの『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、システム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。

QFileSvr.400 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、多くの統合ファイル・システム・コマンドおよび表示画面が有効となります。

QFileSvr.400 ファイル・システムでは、 79 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、以下のコマンドを除きます。

- ADDLNK
- APYJRNCHG
- CHGAUT
- CHGJRNOBJ
- CHGOWN
- DSPAUT
- DSPJRN
- ENDJRN
- RCLLNK
- RCVJRNE
- RST
- RTVJRNE
- SAV
- SNDJRNE
- STRJRN
- WRKOBJOWN
- WRKOBJPGP

77 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。

QFileSvr.400 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、多くの統合ファイル・システム API が有効になります。

QFileSvr.400 ファイル・システムでは、 132 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。

- chown()
- fchown()
- givedescriptor()
- link()
- mkfifo()

- QjoEndJournal()
- QjoRetrieveJournalEntries()
- QjoRetrieveJournalInformation()
- QJORJIDI()
- QJOSJRNE
- QjoStartJournal
- Qp0lGetPathFromFileID()
- symlink()
- takedescriptor()

関連情報

アプリケーション・プログラミング・インターフェース (API)

ネットワーク・ファイル・システム (NFS)

ネットワーク・ファイル・システム (NFS) 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。

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・システムへのアクセス

NFS は、統合ファイル・システムのインターフェースを介してアクセス可能となります。これらの考慮事項および制限事項に注意してください。

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 (NFS) の機能および特性に影響を及ぼします。

以下の方法が可能です。

- ユーザーが System i プラットフォームから、ファイル・システムをクライアントにマウントする。
- ユーザーが UNIX プラットフォームから、ファイル・システムをクライアントにマウントする。
- ユーザーが System i プラットフォームまたは UNIX プラットフォームのいずれでもないシステムから、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、 System iプラットフォーム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。例えば、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。例えば、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、i5/OS 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。

3 番目のシナリオでは、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。

関連資料

35 ページの『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 X/Open Portability Guide (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。

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。

関連概念

11 ページの『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、システム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。

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 (NFS) では、多くの統合ファイル・システムのコマンドが有効になります。

ネットワーク・ファイル・システムでは、 79 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、 77 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、以下のコマンドを除きます。

- APYJRNCHG
- CHGJRNOBJ
- CHGAUD
- CHGATR
- CHGAUT
- CHGOWN
- CHGPGP
- CHKIN
- CHKOUT
- DSPJRN
- ENDJRN
- RCLLNK
- RCVJRNE
- RTVJRNE
- 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。

表 6. ネットワーク・ファイル・システム 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。

表 6. ネットワーク・ファイル・システム CL コマンド (続き)

コマンド	説明
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。

関連情報

i5/OS ネットワーク・ファイル・システム・サポート PDF

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 (NFS) では、多くの統合ファイル・システム API が有効になります。

ネットワーク・ファイル・システムでは、 132 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、以下の API を除きます。

- mkfifo()
- QjoEndJournal()
- QjoRetrieveJournalEntries()
- 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、 i5/OS 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。

関連情報

アプリケーション・プログラミング・インターフェース (API)

ネットワーク・ファイル・システム・バージョン 4 とそれ以前のバージョンとの比較

NFSv4 と NFS バージョン 2 および 3 との相違点の要約。

IBM® i 6.1 から、ネットワーク・ファイル・システム・バージョン 4 (NFSv4) が基本オペレーティング・システムに組み込まれています。以下に、NFSv4 と NFS バージョン 2 および 3 との相違点の簡単な要約を示します。

- NFS バージョン 2 および 3 のプロトコルはステートレスですが、 NFSv4 プロトコルではステートが導入されています。 NFSv4 クライアントによるオブジェクトの使用情報は、サーバーによって保守されます。NFSv4 プロトコルを介した操作 (オープン、ロック、読み取り、書き込みなど) は、クライアントにオブジェクトを使用する意図があることをサーバーに通知するステート情報を携行します。サーバーは、同じオブジェクトを使用する意図がある他のクライアントに関する情報をクライアントに返します。NFSv4 プロトコルによってサーバーで永続的なオープンを使用することにより、 NFS バージョン 2 または 3 のクライアントがファイルへの書き込み中にロックアウトされる可能性があるというような状況が回避されます。
- NFSv4 プロトコルには、基本プロトコルの一部として、バイト範囲ロックのサポートおよび共有モードのサポートが組み込まれています。NFSv4 でのロックはリース・ベースです。つまり、NFSv4 クライアントは、クライアントが所有するオープン状態およびロック状態を保持するために、サーバーとの接触を維持する必要があります。
- NFSv4 プロトコルでは、複合要求フォーマットが導入されています。NFSv4 クライアントは、いくつかの簡単な操作 (例えば、LOOKUP、OPEN、READ など) をサーバーに対する単一の要求に結合することができます。単一の要求にすることによって、 NFSv4 は、1 つのネットワーク交換で複合操作を実行することができます。
- NFSv4 プロトコルは、従来のプロトコル・バージョンで要求されていたセキュリティー・メカニズムよりも改善されたセキュリティー・メカニズムを指定します。IBM i は、従来の AUTH_SYS セキュリティーに加えて、 Kerberos 5 による認証およびデータ保護に対するサポートを提供します。NFSv4 が使用するセキュリティー API によって、将来、新しいセキュリティー・メカニズムの追加が容易になります。
- NFSv4 プロトコルは、ストリング・データの表記を標準化します。プロトコルで使用されるすべてのストリング・データは、ネットワーク内を移動する際に UTF-8 で表されます。ユーザー情報およびグループ情報は、従来のバージョンのように数値としてではなく、ストリング形式で、クライアントとサーバー間でやり取りされます。
- NFSv4 プロトコルは、従来の NFS バージョンの個々のコンポーネントのプロトコルを単一のプロトコル仕様に結合します。NFSv4 プロトコルによってコンタクト・ポイントが 1 つに結合されることにより、ネットワーク・ファイアウォールとの互換性が改善されます。
- NFS バージョン 4 では、TCP などのストリーミング・ネットワーク・トランスポート・プロトコルでの RPC のサポートを必要とします。IBM i により提供される NFSv4 サポートは、 TCP のみを使用します。

NFSv4 プロトコルに関する詳しい情報は、 RFC 3530 http://www.ietf.org/rfc/rfc3530.txt を参照してください。

RPCSEC-GSS 用のネットワークのセットアップ

RPCSEC-GSS 用のネットワークのセットアップ。

このシナリオでセットアップするネットワークは、RPCSEC-GSS 用に構成され、 5 つのサーバーが含まれています。

この作業について

ネットワーク上の 5 つのサーバーは、以下のとおりです。

- kdc.rochester.ibm.com
- alpha.rochester.ibm.com
- beta.rochester.ibm.com

- gamma.rochester.ibm.com
- zeta.rochester.ibm.com

システム kdc.rochester.ibm.com は鍵配布センター (KDC) サーバーとして構成され、 Kerberos レルム ROCHESTER.IBM.COM が作成されます。ここでは、 kdc.rochester.ibm.com と zeta.rochester.ibm.com を除くすべてのシステムは、 RPCSEC-GSS を使用してエクスポートされるファイル・システムを提供する NFS サーバーです。

また、このネットワークでは、以下のユーザーが一部のシステムで構成されています。

- adam
- brian
- charlie
- dave
- eric

注: 以下のセットアップは、1 つの例として示しており、すべての環境に適している訳ではありません。新しい Kerberos レルムをセットアップする前に、『ネットワーク認証サービス』を参照してください。

注: Kerberos では、ネットワーク全体でシステム時刻がほぼ完全に同じである必要があります。この手順を開始する前に、ネットワーク全体で時刻を自動的に同期化するためのメカニズム (NTP など) をセットアップできます。

1. 『ネットワーク認証サービスの構成』トピックに記載されているように KDC サーバーをセットアップします。

   注: KDC サーバーは、他の目的に使用するには理想的ではありません。 KDC の暗号情報が漏えいすると、すべての Kerberos プリンシパルの暗号情報が漏えいします。
2. ユーザーおよびホストごとにプリンシパルを作成します。この例では、関連したユーザーのユーザー・プロファイル名に一致する Kerberos プリンシパルを作成します。プリンシパル名は、プリンシパルに関連したローカル資格情報を判別するために、NFS によってユーザー名にマップされます。プリンシパルとユーザー名とのより一般的なマッピングの使用方法については、『ID マッピング』を参照してください。このネットワークでは、以下のプリンシパルを作成しています。
   - adam
   - brian
   - charlie
   - dave
   - eric
   - nfs/alpha.rochester.ibm.com
   - nfs/beta.rochester.ibm.com
   - nfs/gamma.rochester.ibm.com

   注: この簡単なシナリオでは、選択されたユーザー・プリンシパル名が、IBM i 上の対応するユーザー名に一致する必要があります。 NFS は、ローカル・システム上のユーザー ID およびグループ ID を入手するために、プリンシパル名をユーザー名として使用します。名前が一致しない場合、そのアクセスは匿名アクセスとして扱われます。

   これで、 KDC が構成されます。

3. IBM i ネットワーク認証サービス構成ウィザードを使用して、各 NFS クライアントおよびサーバーを Kerberos クライアントとして構成します。マシンをサーバーとして機能させる場合は必ず、構成ウィザードで NFS サービス・プリンシパルを作成するオプションを選択してください。詳しくは、『ネットワーク認証サービスの構成』を参照してください。

   これで、構成済みの任意のシステムで任意のユーザー・プリンシパルとして kinit を実行できるようになりました。例えば、ユーザー adam として kinit を実行するには、 Qshell 環境で以下のコマンドを実行します。

   kinit adam

   adam の IBM i パスワードではなく Kerberos パスワードを指定する必要があります。
4. システムごとにこのセットアップを繰り返します。
5. クライアント・マシンとサーバー・マシンの両方で、NFS GSS デーモン (QNFSGSSD) が開始されていることを確認します。
6. これで NFS サーバーは作動可能になりましたが、すべてのユーザーが NFS 匿名プロファイル (QNFSANON) にマップされます。すべてのユーザーが、すべてのサーバーに同じ UID および GID で存在しているようにすることをお勧めします。サーバーに存在していないユーザーは、エクスポートされたディレクトリーに対して QNFSANON としてしかアクセスできなくなります。ユーザー名を正しくマップするには、 NFS レジストリー・デーモンを構成する必要があります。NFS レジストリー・デーモンの構成については、 『ID マッピング』を参照してください。

   注: ユーザーは、Kerberos クライアント操作を行う前に、 kinit を使用して有効な資格情報を取得する必要があります。

ID マッピング

ID マッピングは、ローカルの NFS サーバーおよびクライアントに、外部のユーザーおよびグループをローカルのユーザーおよびグループに変換する方法を提供します。

IBM i は、 EIM (エンタープライズ識別マッピング) テクノロジーを使用します。このテクノロジーは、LDAP に基づいて、 ID マッピングを行います。すべての NFS ID マッピング・データは、 LDAP サーバー上に保管されます。

すべてのクライアントおよびサーバーが、 CFGTCP オプション 12 でマシンに構成されている DNS サフィックスに一致する、単一の NFS ドメイン・ネーム・スペースに存在する単純な環境では、EIM 構成は不要です。そのような場合、 IBM i は、ローカルのネーム・レゾリューションを使用して、ユーザーおよびグループのストリング表記をネイティブのユーザー ID に変換します。

クライアントとサーバーが同じ NFS ドメイン・ネーム・スペースに属していない環境や、 Kerberos 5 が使用される環境では、EIM サービスを構成する必要があります。

IBM i マシンが現時点で EIM ドメインの一部でない場合は、システムを既存の EIM ドメインに加えるか、または新しい EIM ドメインを作成する必要があります。『EIM (エンタープライズ識別マッピング) の構成』を参照してください。

初回の EIM 構成をすでに行っている場合、または IBM i マシンがすでに EIM ドメインの一部である場合には、ドメインに正しい NFS レジストリーを追加する必要があります。

| IBM i 上の NFSv4 の場合、ユーザー名マッピングは、「NFS_」という接頭部が付いたレジストリー内に
| なければなりません。例えば、「rochester.ibm.com」ネーム・スペースのユーザー・マッピングを検索する
| 場合、 IBM i は、レジストリー名が「NFS_rochester.ibm.com」であると想定します。

| グループ名マッピングは、「NFSGR_」という接頭部が付いたレジストリーになければなりません。例え
| ば、「rochester.ibm.com」ネーム・スペースのグループ・マッピングを検索する場合、 IBM i は、レジス
| トリー名が「NFSGR_rochester.ibm.com」であると想定します。

| EIM ドメインに適切なレジストリーを追加する方法については、『EIM (エンタープライズ識別マッピン
| グ) レジストリー定義の管理』を参照してください。

| 構成手順が完了したら、 EIM 管理者は、LDAP サーバーに NFS ID マッピング・データを追加できま
| す。EIM ID の処理方法については、『EIM (エンタープライズ識別マッピング) レジストリー定義の管
| 理』を参照してください。

| EIM のマッピング・データを使用するように IBM i を構成した後、 NFS レジストリー・デーモン
| (QNFSRGYD) を再始動する必要があります。NFS レジストリー・デーモンは、始動時に EIM サーバーの
| 使用可能性を検査します。EIM サーバーが検出されると、マッピング機能は EIM を使用します。

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、システムから統合ファイル・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・プログラミング・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 デスクトップからシステムを管理および制御するために使用されるグラフィカル・インターフェースである System i 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。

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、システム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. システムにサインオンします。
2. 続行するには、Enter キーを押してください。
3. 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、システム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**」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またはコマンド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。

関連概念

25 ページの『ファイル・システム』
ファイル・システムは、論理単位として編成された記憶域の特定のセグメントへのアクセスを提供します。システムの論理単位とは、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。

関連資料

82 ページの『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。

『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。

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、 37 ページの『ユーザー定義ファイル・システム (UDFS)』および 70 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[3]	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[3]	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] [3]	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[3]	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[3]	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[3]	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[3]	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

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
CHGJRNOBJ[2] [3]	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[3]	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[3]	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[3]	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。	
CHKOBJITG[3]	オブジェクト保全性の検査。オブジェクトの保全性の違反があるかどうかを検査する。	
CHKOUT[3]	チェックアウト。他のユーザーがオブジェクトを変更しないように、オブジェクトをチェックアウトする。	
CPY[3]	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[3]	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[3]	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[3]	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[3]	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[3]	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。	
DSPJRN[2] [3]	ジャーナルの表示。ジャーナル項目 (1 つまたは複数のレシーバーに入っている) を外部表記に適切な形式に変換する。	
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。	

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
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] [3]	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[3]	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。	
RCLLNK[3]	オブジェクト・リンクの再利用 使用中のマウントされたファイル・システムの問題を識別して、可能な場合は、それを訂正する。	
RCVJRNE[2] [3]	ジャーナル項目の受信。 指定されたユーザー出口プログラムがジャーナル項目を継続して受信できるようにする。	
RLSIFSLCK[3]	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[3]	ディレクトリーの除去。システムからディレクトリーを除去する。	RD、RMDIR
RMVLNK[3]	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[3]	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[3]	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[3]	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。	
RTVJRNE[2] [3]	ジャーナル項目の検索。 特定のジャーナル項目を取得して、その結果を CL 変数中に入れる。	
SAV[3]	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] [3]	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] [3]	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。	

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
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、i5/OS Information Center の『ジャーナル管理』を参照してください。
3. これらのコマンドは Unicode に対応しています。詳細については、i5/OS Information Center の『制御言語でのユニコード対応 (Unicode support in control language)』を参照してください。

関連概念

25 ページの『ファイル・システム』
ファイル・システムは、論理単位として編成された記憶域の特定のセグメントへのアクセスを提供します。システムの論理単位とは、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。

関連タスク

77 ページの『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、システム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。

関連情報

制御言語 (CL)

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。

次のリストは、パス名を指定する際の考慮すべき規則の要約を示しています。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 (coded character set identifier) で表記する必要があります。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。

注: Unicode 対応のコマンドを、Unicode サポートが有効になるような方法で呼び出す場合、それらのコマンドが上記の制限を受けることはありません。例えば、Unicode CCSID でのコマンドやパス名情報を使用して、QCAPCMD API を呼び出すことができます。詳細については、i5/OS Information Center の『制御言語でのユニコード対応 (Unicode support in control language)』を参照してください。

- コマンド行にパス名を入力するときには、単一引用符 (') でそれを囲む必要があります。表示画面にパス名を入力する場合、これらのマークはオプションです。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。パスの各構成要素の名前は、スラッシュ (/) で区切ります。

  注: また、一部の CL コマンドでは、円記号 (¥) を自動的にスラッシュ (/) に変換することによって、円記号 (¥) も区切り記号として使用できます。ただし、円記号 (¥) を他の文字の処理と変わらない方法で処理する CL コマンドもあります。したがって、円記号 (¥) を区切り記号として使用する場合には注意が必要です。

  例えば、

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

  または

  ```
  'Dir1¥Dir2¥Dir3¥UsrFile'
  ```

- スラッシュ (/) と円記号 (¥) 文字およびヌルは、スラッシュ (/) および円記号 (¥) を区切り記号として使用する時には、パス名の個別のコンポーネントには使用できません。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。
- パス名の先頭の区切り記号文字 (例えば、/) 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。例えば、次の通りです。

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

- パス名が区切り記号文字 (例: /) で開始しない場合は、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。
- パス名の先頭で波形記号 (~) 文字の後に区切り記号文字 (例えば、/) が続くと、コマンドを入力するユーザーのホーム・ディレクトリーでそのパスが始まることを示します。例えば、次の通りです。

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

- パス名の先頭に波形記号 (~) 文字、その後にユーザー名、次に区切り記号文字 (例: /) 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。例えば、次の通りです。

```
'~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'
  ```

- i5/OS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'
  ```

  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。たとえば、以下のようにします。

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

- 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'
  ```

- コンポーネント名に以下のいずれかの文字が使用されている場合には、パス名を追加セットの単一引用符 (') または引用符 (") で囲まなければなりません。
  - アスタリスク (*)

    注: i5/OS の特殊値と混同しないようにするため、パス名の先頭文字を単一アスタリスク (*) にはしないでください。
  - 疑問符 (?)
  - 単一引用符 (')
  - 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)。

  例えば、

  ```
  '"/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、以下の例のように 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。さらに、16 進 0 値をパス名に含めることはできません。

関連概念

25 ページの『ファイル・システム』
ファイル・システムは、論理単位として編成された記憶域の特定のセグメントへのアクセスを提供します。システムの論理単位とは、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。

関連情報

制御言語 (CL)

RTVDIRINF および PRTDIRINF コマンドの出力の処理

ディレクトリー情報の検索 (RTVDIRINF) コマンドは、統合ファイル・システム中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。

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。

V6R1 より、これらの 3 つの表は System i ナビゲーター でも作成できます。詳しくは、 100 ページの『Systems Director Navigator for i によるフォルダー属性の収集および分析』を参照してください。

以下の表は、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。情報ファイル接頭部 (INFFILEPFX) パラメーターに *GEN を指定した場合は、このコマンドで生成され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。 例えば、*GEN を指定してこのコマンドを初めて実行した場合は、ファイル QAEZD0001D および QAEZD0001O が情報ライブラリー (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	ディレクトリーとオブジェクト・テーブル間のリレーショナル ID表を結合して完全なパス名情報を得るために使用します。オブジェクト・テーブル内の QEZDIRIDX フィールドの値は、オブジェクトの親ディレクトリーのディレクトリー・テーブルにおける QEZDIRIDX の値と一致します。 **注:** 「ルート」ディレクトリー (/) が RTVDIRINF への入力データとして指定される場合、「ルート」ディレクトリー (/) に親ディレクトリーがなくても QEZDIRIDX の値は 1 になります。
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 QEZOBJNAM) に含まれるバイト数。
QEZNMCCSID	INTEGER	オブジェクト名 (フィールド QEZOBJNAM) を表現する CCSID。

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
QEZREGION	GRAPHIC (2)	オブジェクト名 (フィールド QEZOBJNAM) の国を表す 2 文字の ID。 この ID は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及ぼします。
QEZLANGID	GRAPHIC (3)	オブジェクト名 (フィールド QEZOBJNAM) に使用されている言語を表す 3 文字の ID。
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 NULL がその値として入れられます。
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。

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
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。
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。

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
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。
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。オブジェクトがジャーナル処理されなかった場合は、このフィールドはブランクになります。
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一度もジャーナル処理されたことがない場合、このフィールドのすべてのバイトは 2 進ゼロに設定されます。オブジェクトがジャーナル処理されなかった場合は、このフィールドはブランクになります。
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一度もジャーナル処理されたことがない場合、このフィールドのすべてのバイトは 2 進ゼロに設定されます。オブジェクトがジャーナル処理されなかった場合は、このフィールドはブランクになります。
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一度もジャーナル処理されたことがない場合、このフィールドは 2 進ゼロに設定されます。オブジェクトがジャーナル処理されなかった場合は、このフィールドにはその値として NULL が入れられます。

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
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。 *NOTAVL - 操作を実行するユーザーは、現行オブジェクトの監査値を取得できません。
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。
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。

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
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。 *NOTAVL - 操作を実行するユーザーは、現行作成オブジェクトの監査値を取得できません。 *SYSVAL - ディレクトリーに作成されるオブジェクトのオブジェクト監査値は、システム監査値 (QCRTOBJAUD) によって決まります。
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。 有効な値は次のとおりです。 x'00' (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。 x'01' (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。 x'02' (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。 有効な値は次のとおりです。 x'00' - 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'01' - 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'02' - 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。 i5/OS 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 i5/OS 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 i5/OS ディジタル署名がありません。 1 - オブジェクトには i5/OS 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 i5/OS 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。
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。
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。

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。オブジェクトがジャーナル処理されなかった場合は、このフィールドはブランクになります。
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。オブジェクトがジャーナル処理されなかった場合は、このフィールドはブランクになります。
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、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。		

以下の表は、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	ディレクトリーとオブジェクト・テーブル間のリレーショナル ID表を結合して完全なパス名情報を得るために使用します。オブジェクト・テーブル内の QEZDIRIDX フィールドの値は、オブジェクトの親ディレクトリーのディレクトリー・テーブルにおける QEZDIRIDX の値と一致します。
QEZDIRNAM1 [1]	VARGRAPHIC (1024)	親ディレクトリー・パス。 パス長が 1 KB (1024 バイト) 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。 パス長が 1 KB (1024 バイト) 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。
QEZPARDIR	INTEGER	親ディレクトリー索引
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。		

以下の表には、RTVDIRINF コマンドの実行時に作成されたファイルに関してこのコマンドが保管する情報が示されます。この情報が入っているファイルが存在しないと、RTVDIRINF コマンドによってファイルが作成されます。それ以降にこのコマンドが実行されると、その情報は既存のファイルに追加されます。 PRTDIRINF コマンドはこの情報を使用して、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た情報の保管にどのデータベース・ファイルが使用されたかを判別します。

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。

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
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。
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。		

関連タスク

100 ページの『Systems Director Navigator for i によるフォルダー属性の収集および分析』
Systems Director Navigator for i により、統合ファイル・システム内のオブジェクトの属性を収集し、分析することができます。この使いやすいインターフェースは、ディレクトリー情報検索 (RTVDIRINF) コマンドと同様の機能を提供します。このグラフィカル・インターフェースによって収集したデータを、RTVDIRINF コマンドで収集するデータの場合と同様に、検査および照会することができます。

関連情報

ディレクトリー情報の検索 (RTVDIRINF) コマンド

Qp0lGetPathFromFileID()--Get Path Name of Object from Its File ID API

Qp0lSetAttr()--Set Attributes API

ジャーナル処理済み変更の適用 (APYJRNCHG) コマンド

ジャーナル処理済み変更の除去 (RMVJRNCHG) コマンド

ジャーナル処理済みオブジェクトの変更 (CHGJRNOBJ) コマンド

ディレクトリー情報の印刷 (PRTDIRINF) コマンド

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。

以下のリストは、ディレクトリー情報の検索 (RTVDIRINF) コマンドによって作成されたデータにアクセスする方法を示しています。

- ディレクトリー情報の印刷 (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。

- i5/OS オペレーティング・システムで、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

  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 System i 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。
- コマンド発行により各種の照会を実行する代わりに、System i ナビゲーター を使用して、容易にディレクトリー情報データ (System i ナビゲーター ではフォルダー属性データ) の取得、表示、および分析が可能です。詳細は、『System i ナビゲーター によるフォルダー属性の収集および分析』を参照してください。

関連タスク

『Systems Director Navigator for i によるフォルダー属性の収集および分析』

Systems Director Navigator for i により、統合ファイル・システム内のオブジェクトの属性を収集し、分析することができます。この使いやすいインターフェースは、ディレクトリー情報検索 (RTVDIRINF) コマンドと同様の機能を提供します。このグラフィカル・インターフェースによって収集したデータを、RTVDIRINF コマンドで収集するデータの場合と同様に、検査および照会することができます。

関連情報

ディレクトリー情報の印刷 (PRTDIRINF) コマンド

SQL 対話式セッションの開始 (STRSQL) コマンド

組み込み SQL プログラミング

SQL コール・レベル・インターフェース

RTVDIRINF のデータの使用:

以下の例は、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。

- 85 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。
- 98 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。 85 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 QAEZDxxxxO.QEZOBJNAM FROM library_name/QAEZDxxxxD,
  library_name/QAEZDxxxxO WHERE QAEZDxxxxD.QEZDIRNAM1 = '/MYDIR' AND
  QAEZDxxxxD.QEZDIRIDX = QAEZDxxxxO.QEZDIRIDX
  ```

- 98 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。

Systems Director Navigator for i によるフォルダー属性の収集および分析:

| Systems Director Navigator for i により、統合ファイル・システム内のオブジェクトの属性を収集し、分析
| することができます。この使いやすいインターフェースは、ディレクトリー情報検索 (RTVDIRINF) コマン

| ドと同様の機能を提供します。このグラフィカル・インターフェースによって収集したデータを、
| RTVDIRINF コマンドで収集するデータの場合と同様に、検査および照会することができます。

| **注:** この方式は、Systems Director Navigator for i V6R1、またはそれ以降のシステムで使用できます。

| 統合ファイル・システムで特定のオブジェクトの属性についてのレポートを生成するには、次のステップに
| 従ってください。

| 1. Systems Director Navigator for i で、**「ユーザー接続」 → *ご使用のシステム* → 「ファイル・システム」 → 「統合ファイル・システム」**の順に展開します。
| 2. 関心のあるオブジェクトを含むフォルダーの横にある矢印をクリックし、**「フォルダー属性情報」 → 「属性の収集」**の順に選択します。
3. 「属性の収集 (Collect Attributes)」ウィンドウで、プリファレンスを指定します。サブフォルダーの属性も収集する場合は、**「このフォルダーに含まれるサブフォルダーの内容を組み込む」**を選択します。必要に応じて、**「ファイル接頭部」**および**「ライブラリー」**を指定できます。**「OK」**をクリックして、オブジェクト属性の収集を開始します。

   このデータ収集処理に、時間がかかる場合があります。「収集された属性の表示」ウィンドウが表示されるまで、数秒間待つ必要があります。
| 4. 「収集された属性の表示」ウィンドウで、分析する項目の横にある矢印をクリックし、**「情報の分析」**を選択します。

   | **注:** System i ナビゲーター から属性の収集操作を実行した場合、またはディレクトリー情報検索 (RTVDIRINF) コマンドを実行した場合は、フォルダーの横にある矢印をクリックして、**「フォルダー属性情報」 → 「収集された属性の表示」** の順に選択することにより、「収集された属性の表示」を直接表示できます。
5. 「フォルダー情報の分析」ウィンドウで、**「列」**、**「フィルター」**、および**「順序」**タブにより、表示する属性をカスタマイズします。それから、**「OK」**をクリックしてフォルダー属性情報レポートを生成します。

   以下に例を示します。 10 MB より大きいサイズのファイルおよびそれらの所有者を、まず「サイズ」、次いで「所有者」の順にソートして表示するとします。
   - **「列」**タブで、*「所有者」*を選択し、**「追加」**をクリックします。*「親フォルダー・パス」*を選択し、**「追加」**をクリックします。*「オブジェクト名」*を選択し、**「追加」**をクリックします。*「割り振りサイズ」*を選択し、**「追加」**をクリックします。
   - **「フィルター」**タブで、**「フィールド」**の*「割り振りサイズ」*を選択します。**「条件」**で*「より大のサイズ」*を選択し、**「サイズ」**に 10 を入力、**「単位」**で*「メガバイト」*を選択します。**「追加」**をクリックして、フィルターを作成します。
   - **「順序」**タブで、**「最初のソート」**として*「割り振りサイズ」*および*「降順」*を選択し、**「2 番目のソート」**として*「所有者」*および*「降順」*を選択します。
   - **「OK」**をクリックします。「フォルダー属性情報レポート」に調整された情報が表示されます。
| 6. 「フォルダー属性情報レポート」ウィンドウで、レポート・ウィンドウを終了せずに、「削除」アクションを選択して、レポートにリストされている統合ファイル・システム・オブジェクトを除去できます。

関連資料

85 ページの『RTVDIRINF および PRTDIRINF コマンドの出力の処理』
ディレクトリー情報の検索 (RTVDIRINF) コマンドは、統合ファイル・システム中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。

99 ページの『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。

API を使用したアクセス

アプリケーション・プログラミング・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。

関連資料

132 ページの『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多くのアプリケーション・プログラミング・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。

PC を使用したアクセス

PC が System i 製品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システムから PC にコピーする必要がある場合には、システム・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 System i 製品と PC との間でコピーされるオブジェクトはすべて、EBCDIC (拡張 2 進化 10 進コード) と ASCII (情報交換用米国標準コード) の間で自動変換することができます。 IBM i Access Family を構成して、この変換を自動的に実行できます。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。

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。例えば、テキストを含むストリーム・ファイルを、PC のエディターを使用して編集できます。

PC を使用して System i 製品に接続すると、統合ファイル・システムにより、システム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 System i ナビゲーター (IBM i 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、システム上で稼働する i5/OS NetServer にアクセスします。

関連概念

『System i ナビゲーター を使用したアクセス』
System i ナビゲーター は、ユーザーの Windows デスクトップからユーザー・システムを管理するためのグラフィカル・インターフェースです。System i ナビゲーター によって、システムの運用および管理がより簡単で生産的になります。

関連タスク

104 ページの『i5/OS NetServer を使用したアクセス』
i5/OS Support for Windows Network Neighborhood (i5/OS NetServer) は、Windows クライアントが i5/OS 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できるようにするための機能です。 i5/OS NetServer によって、Windows ソフトウェアを実行する PC は、ご使用の System i プラットフォームで管理され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。

関連資料

105 ページの『ファイル転送プロトコル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、System i プラットフォーム上のファイルを転送することができます。

System i ナビゲーター を使用したアクセス

System i ナビゲーター は、ユーザーの Windows デスクトップからユーザー・システムを管理するためのグラフィカル・インターフェースです。System i ナビゲーター によって、システムの運用および管理がより簡単で生産的になります。

例えば、あるシステムから別のシステムにユーザー・プロファイルを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。

System i ナビゲーター を使用して、さまざまな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統合ファイル・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

- 150 ページの『フォルダーの作成』
- 150 ページの『ファイルまたはフォルダーの除去』
- 152 ページの『許可の設定』
- 152 ページの『ファイル・テキスト変換のセットアップ』
- 153 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』
- 153 ページの『ファイルまたはフォルダー送信のためのオプション変更』
- 157 ページの『オブジェクトをスキャンするかどうかの設定』
- 158 ページの『オブジェクトのチェックイン』
- 158 ページの『オブジェクトのチェックアウト』
- 100 ページの『Systems Director Navigator for i によるフォルダー属性の収集および分析』

ファイル共用の処理

- 154 ページの『ファイル共用の作成』
- 154 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

- 155 ページの『新規のユーザー定義ファイル・システムの作成』
- 155 ページの『ユーザー定義ファイル・システムのマウント』
- 156 ページの『ユーザー定義ファイル・システムのアンマウント』
- 156 ページの『動的にマウントされたファイル・システムの処理』

オブジェクトのジャーナル処理

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

i5/OS NetServer を使用したアクセス

i5/OS Support for Windows Network Neighborhood (i5/OS NetServer) は、Windows クライアントが i5/OS 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できるようにするための機能です。 i5/OS NetServer によって、Windows ソフトウェアを実行する PC は、ご使用の System i プラットフォームで管理され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、i5/OS NetServer 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、i5/OS NetServer 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。 i5/OS 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 i5/OS NetServer 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。

i5/OS NetServer ファイル共用は、i5/OS NetServer が System i 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、システム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。 i5/OS NetServer を使用してファイル共用を処理する前に、 i5/OS NetServer ファイル共用を作成し、必要な場合は、System i ナビゲーター を使用して i5/OS NetServer ファイル共用を変更する必要があります。

i5/OS NetServer 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. i5/OS NetServer ファイル共用の名前を入力します。 例えば、以下の構文を入力できます。
   `¥¥QSYSTEM1¥`*`Sharename`*

   注: `QSYSTEM1` は i5/OS NetServer 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: i5/OS NetServer を使用して接続するとき、システム名は IBM i Access Family により使用される名前と異なる場合があります。例えば、i5/OS NetServer 名は `QAS400X` となり、ファイルを処理するための

パスは、¥¥QAS400X¥QDLS¥MYFOLDER.FLR¥MYFILE.DOC となります。しかし、IBM i 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。

i5/OS NetServer を使用して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、システム名の下の第 1 レベルとして表されます。例えば、/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 i5/OS ファイル・システムよりパフォーマンスがはるかに良くなります。「ルート」(/) ファイル・システムにファイルを移動することが必要な場合があるかもしれません。詳しくは、150 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。

関連情報

i5/OS NetServer

i5/OS NetServer ファイル共用

ファイル転送プロトコル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、System i プラットフォーム上のファイルを転送することができます。

また、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、システム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)
- i5/OS NetClient ファイル・システム (QNTC)
- QFileSvr.400 ファイル・システム

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。

関連情報

ファイル転送プロトコル

ファイル転送プロトコルを使用したファイル転送

統合ファイル・システムの変換

i5/OS オペレーティング・システムは、統合ファイル・システムのファイル・システム用に自動変換を実行することにより、新しいディレクトリー・フォーマットや Unicode 規格をサポートします。

*TYPE1 から *TYPE2 へのディレクトリーの変換

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。

i5/OS V5R3 またはそれ以降のリリース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に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変換が自動的に開始されます。 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。

*TYPE1 から *TYPE2 への変換の概要

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、19 ページの『スキャンのサポート』を参照してください。

i5/OS V5R3M0 オペレーティング・システムまたはそれ以降のリリース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に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変換が自動的に開始されます。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 (Initial Program Load) のたびに変換が再開します。

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。

注: V5R3 オペレーティング・システムまたはそれ以降のリリース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。

関連概念

9 ページの『*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。

関連資料

『変換状況の判別』
i5/OS V5R3M0 オペレーティング・システムまたはそれ以降のリリース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に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変換が自動的に開始されます。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。

110 ページの『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は、オペレーティング・システムの 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。

ディレクトリー変換に関する考慮事項

ディレクトリー変換処理の際に考慮すべきいくつかの点があります。

変換状況の判別:

i5/OS V5R3M0 オペレーティング・システムまたはそれ以降のリリース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に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変換が自動的に開始されます。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た後、この履歴情報を削除することができます。

関連情報

ディレクトリーの変換 (CVTDIR) コマンド

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。

関連資料

109 ページの『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、代替ユーザー・プロファイルに設定されます。

名前変更されるオブジェクト:

*TYPE2 ディレクトリーでは、リンク名は有効な UTF-16 の名前でなければなりません。

*TYPE2 ディレクトリーの命名規則は、UCS2 レベル 1 の名前の *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。

UTF-16 の詳細については、Unicode ホーム・ページ (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 文字で構成できるものもあります。

例えば、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、アクセント (例えば、é または à) あるいはウムラウト (例えば、ä または ö) をもつ文字は、そ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または*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一時的に存在するディレクトリーの数が同じユーザー・プロファイルによって所有されます。

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、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。

*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、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。

関連資料

108 ページの『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。

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は次の通りです。

- *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: i5/OS V5R3M0 オペレーティング・システムまたはそれ以降のリリースをインストールする前に、OS/400 V5R2 (CVTDIR) コマンドで、*ESTIMATE オプションの実行を考慮したい場合があります。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。

関連情報

ディレクトリーの変換 (CVTDIR) コマンド

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。

関連概念

11 ページの『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、システム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。

関連資料

108 ページの『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。

関連情報

symlink()--Make Symbolic Link

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は、オペレーティング・システムの 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。

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・システム内に保存および復元することができます。

同じように、*TYPE2 として存在するディレクトリーは、そのディレクトリーが *TYPE2 のディレクトリーとして存在していた時点で *TYPE1 制限を超えたのでない限り、*TYPE1 フォーマットのファイル・システム内に保存および復元できます。

ヒント: 統合ファイル・システム・オブジェクトの再利用:

*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、記憶域の再利用 (RCLSTG) と オブジェクト・リンクの再利用 (RCLLNK) のコマンドは実行できません。

OMIT(*DIR) パラメーター値を RCLSTG コマンドで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、統合ファイル・システム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。

関連概念

122 ページの『「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムの再利用操作』
「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムの再利用は、オブジェクト・リンクの再利用 (RCLLNK) コマンドおよび記憶域の再利用 (RCLSTG) コマンドを使用して遂行できます。

関連情報

記憶域の再利用 (RCLSTG) コマンド

オブジェクト・リンクの再利用 (RCLLNK) コマンド

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。

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。

関連概念

19 ページの『スキャンのサポート』
i5/OS オペレーティング・システムを使用して、統合ファイル・システムのオブジェクトをスキャンすることができます。

21 ページの『関連するシステム値』
QSCANFS および QSCANFSCTL のシステム値を使用して、ご使用のシステムに適したスキャン環境を設定することができます。

追加の文字をサポートする名前変換

ファイル・システムでは Unicode で名前が格納されます。大文字と小文字を区別しないファイル・システムの場合、特定の Unicode 規格による文字の変更や大文字小文字の規則の変更に影響されます。「ルート」(/) や CASE (*MONO) により作成されるユーザー定義ファイル・システム (UDFS) など、大文字と小文字を区別しないファイル・システムは、i5/OS V6R1 より Unicode 規格 4.0 をサポートしています。

i5/OS がインストールされるとすぐに、Unicode 規格 4.0 をサポートするように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、その Unicode 規格へのディレクトリーの変換が自動的に開始されます。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。

関連情報

http://www.unicode.org

自動名前変換の概要

「ルート」(/) や CASE (*MONO) により作成される UDFS などの大文字と小文字を区別しないファイル・システムは、Unicode 規格 4.0 形式で格納される名前をサポートします。システムは、名前に使用されている追加の文字をサポートするために自動名前変換を実行します。

統合ファイル・システムにおいて、操作されるオブジェクトを特定するために統合ファイル・システムの CL コマンドまたは API に名前が与えられると、名前の検索が実行されます。新しく文字をサポートするために、追加の文字が定義されたり大文字小文字の規則が更新されたりすると、そうした変更の影響を受ける文字を含む名前は検出できない恐れがあります。

統合ファイル・システムは、Unicode 形式ですべての名前を格納します。 i5/OS V6R1 以前、統合ファイル・システムは Unicode 規格 2.0 をサポートしていました。 V6R1 より、統合ファイル・システムは Unicode 規格 4.0 をサポートしています。名前変換ユーティリティーが自動的に実行されて、大文字と小文字の区別をしないファイル・システムのディレクトリーが更新され、Unicode 規格 4.0 がサポートされます。大文字と小文字の区別をしないファイル・システムのうち、この変換に含められるのは「ルート」(/)、および CASE (*MONO) により作成される (任意の補助記憶域プールの) ユーザー定義ファイル・システムです。

「ルート」(/) および基本的なユーザー補助記憶域プール (1 から 32 まで) 中の UDFS における名前の変換は、i5/OS オペレーティング・システムのインストール後すぐに自動的に開始され、QFILESYS1 システム・ジョブの優先度の低いスレッドで実行されます。変換機能の未完了時にシステムが初期プログラム・ロード (IPL) を実行した場合、IPL の完了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。

独立 ASP においてオンに変更する操作中、独立ディスク・プール・グループ ##### のジョブ QFSYS##### が自動的に開始され、独立 ASP ##### の UDFS の名前変換が、このシステムのジョブにおける優先度の低いスレッドで実行されます。独立 ASP の UDFS の名前変換では、「ルート」(/) および基

本ユーザー ASP の UDFS の名前変換が完了するまで、ディレクトリーの変換は開始されません。変換機能がまだ完了していない時点で、独立 ASP がオフに変更されたりシステムが IPL を実行したりした場合、IPL が完了して独立 ASP がオンに変更されると変換機能が再開されます。

注: 複数の独立 ASP が同時にオンに変更される場合、常に 1 つの独立 ASP の名前変換のみアクティブになります。 1 つの変換が完了すると、別の変換がアクティブになります。

以前のリリースでオブジェクト変換分析 (ANZOBJCVN) コマンドを使用すれば、名前変換に向けてプランを立てることができます。 ANZOBJCVN コマンドは、システム上のオブジェクトに関する変換についての情報を収集または報告します。それには、新リリースで使用される新しい Unicode 文字や大文字小文字の規則に影響される可能性のある文字を少なくとも 1 つ含む統合ファイル・システムのオブジェクト名についての情報が含まれます。影響を受けるオブジェクトは、新リリースへのアップグレード後に実行される自動変換時に、新しい Unicode 文字や大文字小文字の規則に合わせて名前変更される可能性があります。変更されたオブジェクト名はアプリケーションに影響を与える場合があるため、新しいリリースにアップグレードする前に影響を受けるオブジェクトの名前を変更しておくこともできます。

関連概念

17 ページの『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。

関連情報

http://www.unicode.org

提案: PTF のインストールおよびオブジェクト変換のためのシステム分析 (Suggested: Installing PTFs and analyzing your system for object conversions)

統合ファイル・システムのセキュリティーの計画

名前変換に関する考慮事項

自動名前変換の際に考慮すべきいくつかの点があります。

変換状況の判別:

変換済みファイル・システムを示す一般的なメッセージ、および各種の進行メッセージが、変換を実行するジョブのジョブ・ログに送信されます。

「ルート」(/) や、ASP (1 から 32 まで) の UDFS の場合、メッセージは QFILESYS1 ジョブのジョブ・ログに送信されます。独立 ASP の UDFS の場合、メッセージは独立ディスク・プール・グループ ##### におけるジョブ QFSYS##### のジョブ・ログに送信されます。これらのメッセージには、変換されているファイル・システムや、ファイル・システム内の処理済みリンクの数などの情報が含まれます。多くの一般的なメッセージは、QSYSOPR メッセージ・キューにも送られます。エラー・メッセージは、QHST ヒストリー・ログに送信されます。

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た時点で、この履歴情報を削除することができます。

名前変更されるオブジェクト:

大文字と小文字を区別しないファイル・システムでは、検索時に名前の大文字と小文字を区別しません。大文字小文字の規則の変更は、大文字小文字の区別なしの際に同一視される文字に影響を及ぼします。

自動名前変換中に、Unicode 規格 2.0 では同一に扱われなかった名前が、Unicode 規格 4.0 で同一の名前として扱われる場合があります。その際には、片方のオブジェクトの名前を変更する必要があります。固有な名前への変更後、メッセージ CPDA0BC が、元の名前と新しい名前をリストする QHST に送信されます。

異なるバージョンの Unicode 規格についての詳細は、Unicode ホーム・ページ (www.unicode.org) を参照してください。

ユーザー・プロファイルに関する考慮事項:

変換実行中、システムは追加のストレージを必要とするため、ユーザー・プロファイルが所有するストレージの量に影響が及びます。

名前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限度に達した場合、システムはユーザー・プロファイル用の限度を引き上げ、メッセージ CPIA08C を QSYSOPR メッセージ・キューに送ります。

ヒント: シンボリック・リンク:

シンボリック・リンクとは、他のオブジェクトを指すパス名を含む統合ファイル・システム・オブジェクトです。シンボリック・リンク内のパス名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。

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、現行リリース用に Unicode 規格バージョンを使用できるようにまだ変換されていない場合は、この変換処理を実行するジョブが独立 ASP をオンに変更した際に開始されます。

システムは独立ディスク・プール・グループ ##### 用の QFSYS##### ジョブを開始します。自動名前変換は、このジョブの優先度の低いスレッドで実行され、グループ内の各独立補助記憶域プールごとに 1 つのスレッドが開始されます。自動名前変換が完了していない場合、システムは独立ディスク・プール・グループがオンに変更されるたびに、このジョブを開始します。

ヒント: 保管と復元:

Unicode 規格 2.0 を使用するディレクトリーを保存し、Unicode 規格 4.0 を使用するファイル・システムに復元できます。また、その逆も可能です。

オブジェクトは、復元時にディレクトリーにリンクされます。オブジェクトがリンクされたディレクトリーの Unicode 規格が、復元されるオブジェクトの名前に適用されます。そのため、ディレクトリーが使用する Unicode 規格において固有の名前でない場合は、オブジェクトを復元できません。そのような場合、システムは理由コード 1 でメッセージ CPD37B9 をジョブ・ログに送信します。

ヒント: 統合ファイル・システム・オブジェクトの再利用:

自動名前変換中にエラーが発生した場合、記憶域の再利用 (RCLSTG) およびオブジェクト・リンクの再利用 (RCLLNK) コマンドによって、ディレクトリーとそのサブディレクトリーの変換を完了できます。これは、ファイル・システム用の自動名前変換処理が完了した後にのみ実行されます。

また RCLSTG コマンドは、破損したディレクトリーを変換して Unicode 規格 4.0 を使用できるようにします。

関連概念

122 ページの『「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムの再利用操作』
「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムの再利用は、オブジェクト・リンクの再利用 (RCLLNK) コマンドおよび記憶域の再利用 (RCLSTG) コマンドを使用して遂行できます。

関連情報

記憶域の再利用 (RCLSTG) コマンド

オブジェクト・リンクの再利用 (RCLLNK) コマンド

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。

関連情報

ジャーナル管理

ジャーナル処理の概要

このトピックでは、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ます。

関連情報

ジャーナル管理

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。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。

関連概念

『ジャーナル処理するオブジェクト』
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かどうかの判別で考慮する問題がいくつかあります。

ジャーナル処理するオブジェクト

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かどうかの判別で考慮する問題がいくつかあります。

ジャーナル処理が必要となるオブジェクトを判別するには、次の問題が考慮する必要があります。

- 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。例えば、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存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、システム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、システム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、最後の保存操作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、i5/OS 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 System i 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。 System i ナビゲーター を介して、ジャーナル処理の開始およびジャーナル処理の終了をすることができます。ジャーナル処理状況情報も表示できます。

注: メモリー・マップ・ストリーム・ファイル、仮想ボリューム・ファイル、および Integrated xSeries Server (IXS) ネットワーク・ストレージ・スペースとして使用されるストリーム・ファイルはジャーナル処理できません。 また、ブロック特殊ファイル (*BLKSF) 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。

以下のリストには、統合ファイル・システムでのジャーナル処理サポートが要約されています。

- 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、ジャーナル処理済みオブジェクトの変更、およびジャーナル処理済みオブジェクト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。例え

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に対してジャーナル処理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インターフェースでサポートされる最大書き込み限界は 2 GB - 1 です。RCVSIZOPT (*MAXOPT2 または *MAXOPT3) が指定された場合の最大ジャーナル項目サイズは 4 000 000 000 バイトです。そうでない場合、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、エラーの発生を防ぐため、*MAXOPT2 または *MAXOPT3 サポートを使用することが必要です。

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。

関連概念

16 ページの『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。

4 ページの『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。そのリストには、他のディレクトリーを含めることができます。

13 ページの『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。

関連タスク

120 ページの『ジャーナル処理の開始』
ジャーナル処理を開始するには、System i ナビゲーター を介して次のステップをオブジェクトに対して実行してください。

121 ページの『ジャーナル処理の終了』
オブジェクトに対するジャーナル処理を開始した後、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は、このトピックで説明したステップを使用することができます。

121 ページの『ジャーナル処理の変更』
オブジェクトに対するジャーナル処理を開始した後、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する場合は、ジャーナル処理済みオブジェクトの変更 (CHGJRNOBJ) コマンドを使用して、ジャーナル処理済みオブジェクトを変更することができます。

関連情報

ジャーナル管理

ジャーナル項目情報ファインダー

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のタイプが、ジャーナル処理も可能なタイプである場合に限って、これら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
- ジャーナル処理済み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

関連情報

ジャーナル管理

ジャーナル項目情報ファインダー

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)
- B7 - 作成されたオブジェクト権限情報
- 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。

関連情報

ジャーナル項目情報ファインダー

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

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。

関連概念

12 ページの『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。

ジャーナル処理の開始

ジャーナル処理を開始するには、System i ナビゲーター を介して次のステップをオブジェクトに対して実行してください。

1. 「**System i 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. ジャーナル処理をする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。

関連情報

ジャーナル処理の開始 (STRJRN) コマンド

ジャーナル処理の開始 (QjoStartJournal) API

ジャーナル管理

ジャーナル処理の変更

オブジェクトに対するジャーナル処理を開始した後、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する場合は、ジャーナル処理済みオブジェクトの変更 (CHGJRNOBJ) コマンドを使用して、ジャーナル処理済みオブジェクトを変更することができます。

関連タスク

120 ページの『ジャーナル処理の開始』

ジャーナル処理を開始するには、System i ナビゲーター を介して次のステップをオブジェクトに対して実行してください。

『ジャーナル処理の終了』

オブジェクトに対するジャーナル処理を開始した後、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は、このトピックで説明したステップを使用することができます。

関連情報

ジャーナル処理済みオブジェクトの変更 (CHGJRNOBJ) コマンド

ジャーナル処理の終了

オブジェクトに対するジャーナル処理を開始した後、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は、このトピックで説明したステップを使用することができます。

System i 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、以下の手順を実行してください。

1. 「**System i 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する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。

関連タスク

120 ページの『ジャーナル処理の開始』
ジャーナル処理を開始するには、System i ナビゲーター を介して次のステップをオブジェクトに対して実行してください。

関連情報

ジャーナルの終了 (ENDJRN) コマンド

ジャーナル処理の終了 (QjoEndJournal) API

ジャーナル管理

「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムの再利用操作

「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムの再利用は、オブジェクト・リンクの再利用 (RCLLNK) コマンドおよび記憶域の再利用 (RCLSTG) コマンドを使用して遂行できます。

RCLLNK と RCLSTG のコマンドを使用することによって、次のタスクを実行できます。

- オブジェクト・ユーザー・プロファイルの問題の訂正
- ユーザー定義ファイル・システムの問題の訂正
- 内部オブジェクトの問題の訂正
- 無効オブジェクト・リンクの除去
- 損傷したオブジェクトの処理
- 欠落したシステム・オブジェクトの作成
- 内部ファイル・システムの問題の訂正 (RCLSTG のみ)
- 脱落したオブジェクトの検索 (RCLSTG のみ)
- 自動名前変換中にエラーが発生した場合、ディレクトリーの変換を完了して Unicode 規格 4.0 を使用できるようにしてください。

オブジェクト・リンクの再利用 (RCLLNK) コマンドと記憶域の再利用 (RCLSTG) コマンドの比較

オブジェクト・リンクの再利用 (RCLLNK) コマンドと記憶域の再利用 (RCLSTG) コマンドの両方を使用して、「ルート」 (/)、QOpenSys、およびユーザー定義ファイル・システムの問題を訂正することができます。

RCLLNK コマンドは、使用中のマウントされたファイル・システムの問題を識別して、可能な場合は、それを訂正します。RCLSTG コマンドには、この機能はありません。ただし、RCLLNK コマンドで識別できない問題または訂正できない問題は、RCLSTG コマンドで訂正できます。以下の表は、この 2 つのコマンドをさらに詳細に比較したものです。

表 11. RCLLNK コマンドと RCLSTG コマンドの比較

	RCLLNK OBJ('/MyDir/MyObj')	RCLSTG ASPDEV(*SYSBAS)	RCLSTG ASPDEV(<IASPNAME>)
システムが制限状態で必要となりますか ?	いいえ (No)	はい (Yes)	いいえ (No)

表 11. RCLLNK コマンドと RCLSTG コマンドの比較 (続き)

	RCLLNK OBJ('/MyDir/MyObj')	RCLSTG ASPDEV(*SYSBAS)	RCLSTG ASPDEV(<IASPNAME>)
再利用操作中にすべてのファイル・システムが使用可能ですか ?	はい (Yes)	いいえ (No)	再利用している独立 ASP 中のファイル・システムは使用不可です。
どの ASP のオブジェクトを再利用できますか ?	システム、ユーザー、および独立 ASP 中のオブジェクトを再利用します。	システムおよびユーザー ASP 中のオブジェクトを再利用します。	独立 ASP 中のオブジェクトを再利用します。
オブジェクトをどのように再利用しますか ?	オブジェクトはコマンドに指定された通り個別またはサブツリー・ベースで再利用されます。	オブジェクトはシステム全体で再利用されます。	オブジェクトは独立 ASP ベースで再利用されます。
ファイル・システムの既知の問題で、該当するどのような問題を識別して訂正しますか (可能な場合) ?	ほとんど (Most) (詳細については、 122 ページの『「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムの再利用操作』 を参照)	すべて (All)	すべて (All)
脱落したオブジェクトを検索しますか ?	いいえ (No)	はい (Yes)	はい (Yes)
アンマウントしたファイル・システムのオブジェクトを再利用しますか ?	いいえ (No)	はい (Yes)	はい (Yes)
コマンドはスレッド・セーフですか ?	はい (Yes)	いいえ (No)	いいえ (No)
コマンドのいくつのインスタンスを同時に実行できますか ?	複数のインスタンス	単一インスタンス	単一インスタンス
適用できる統合ファイル・システム提供のどのようなオブジェクトを必要な場合に再作成しますか ?	すべて (All)	ほとんど (Most) (詳細については、 124 ページの『統合ファイル・システム提供オブジェクトの再作成』 を参照)	なし
再利用していない損傷したオブジェクトを識別できますか ?	はい (Yes)	いいえ (No)	いいえ (No)

関連概念

125 ページの『例: オブジェクト・リンクの再利用 (RCLLNK) コマンド』
これらの例では、「ルート」(/)、QOpenSys、およびマウントされたユーザー定義ファイル・システムでオブジェクトを再利用するために、オブジェクト・リンクの再利用 (RCLLNK) コマンドが使用できる状態について説明します。

関連資料

『統合ファイル・システム提供オブジェクトの再作成』
この表には、オブジェクトが存在しない場合にオブジェクト・リンクの再利用 (RCLLNK) コマンドで再作成される、統合ファイル・システム提供のオブジェクトが示されます。 これらのオブジェクトは通常、初期プログラム・ロード (IPL) 中に作成されます。また、必要があれば、記憶域の再利用 (RCLSTG) コマンドを使用してこれらのオブジェクトの一部を再作成することもできます。

関連情報

記憶域の再利用 (RCLSTG) コマンド

オブジェクト・リンクの再利用 (RCLLNK) コマンド

オブジェクト・リンクの再利用 (RCLLNK) コマンド

オブジェクト・リンクの再利用 (RCLLNK) コマンドは、システムを制限状態にすることなく、「ルート」(/)、QOpenSys、およびマウントされたユーザー定義ファイル・システム中の損傷したオブジェクトを識別して修復します。生産性を犠牲にすることなく、ファイル・システム中の問題を修正できます。

RCLLNK コマンドは、記憶域の再利用 (RCLSTG) コマンドの代替としてさまざまな状態で使用できます。例えば、以下の状態で問題を識別して修正するには、RCLLNK は理想的です。

- 問題を単一オブジェクトに分離する。
- 問題をオブジェクトのグループに分離する。
- 損傷したオブジェクトを識別して削除する必要がある。
- 再利用操作中にシステムを制限状態にできない。
- 再利用操作中に独立 ASP を使用可能にする必要がある。

統合ファイル・システム提供オブジェクトの再作成

この表には、オブジェクトが存在しない場合にオブジェクト・リンクの再利用 (RCLLNK) コマンドで再作成される、統合ファイル・システム提供のオブジェクトが示されます。 これらのオブジェクトは通常、初期プログラム・ロード (IPL) 中に作成されます。また、必要があれば、記憶域の再利用 (RCLSTG) コマンドを使用してこれらのオブジェクトの一部を再作成することもできます。

表 12. 統合ファイル・システムによって提供され、RCLLNK コマンドと RCLSTG コマンドによって再作成されるオブジェクト

パス名	タイプ	RCLLNK による再作成	RCLSTG ASPDEV(*SYSBASE) による再作成
/dev/zero	*CHRSF	はい (Yes)	はい (Yes)
/dev/null	*CHRSF	はい (Yes)	はい (Yes)
/dev/xti/tcp	*CHRSF	はい (Yes)	いいえ (No)
/dev/xti/udp	*CHRSF	はい (Yes)	いいえ (No)
/etc/vfs	*STMF	はい (Yes)	いいえ (No)

RCLLNK コマンドで、存在しない統合ファイル・システム提供のオブジェクトを再作成するには、親ディレクトリーの指定で SUBTREE パラメーターを *DIR または *ALL に設定して実行する必要があります。このコマンドはシステム・オブジェクトの親ディレクトリーを正常に再利用する必要があります。例えば、

```
RCLLNK OBJ('/dev') SUBTREE(*DIR)
```

これは、/dev/zero と /dev/null *CHRSF のオブジェクトが存在しない場合、これらを再作成します。

RCLSTG コマンドで、存在しない統合ファイル・システム提供オブジェクトを再作成するには、**ASPDEV** パラメーターを *SYSBASE に設定し、再利用のディレクトリー・リカバリー部分を省略しないで、RCLSTG コマンドを実行する必要があります。

関連概念

7 ページの『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まだ存在しない場合は、統合ファイル・システムによってそれらが作成されます。システムによる作成後、これらのディレクトリーの移動や名前変更は実行しないでください。

関連情報

オブジェクト・リンクの再利用 (RCLLNK) コマンド

例: オブジェクト・リンクの再利用 (RCLLNK) コマンド

これらの例では、「ルート」(/)、QOpenSys、およびマウントされたユーザー定義ファイル・システムでオブジェクトを再利用するために、オブジェクト・リンクの再利用 (RCLLNK) コマンドが使用できる状態について説明します。

例: オブジェクトの問題を訂正する

この状態では、既知の問題が 1 つのオブジェクトに分離されます。 このオブジェクトは損傷して使用できません。また、そのオブジェクトのバックアップ・バージョンをメディアから復元できません。この問題は、通常のファイル・システム操作を中断しないで即時訂正する必要があります。

オブジェクトを再利用するには、次のコマンドを使用してください。

```
RCLLNK OBJ('/MyDir/MyBadObject') SUBTREE(*NONE)
```

この場合、/MyDir/MyBadObject は損傷して使用できないオブジェクトの名前です。

例: ディレクトリー・サブツリーに存在する問題を訂正する

この状態では、既知の問題がディレクトリー・サブツリー内のオブジェクトのグループに分離されます。アプリケーションは、ディレクトリー・サブツリー内の問題のために失敗します。この問題は、通常のファイル・システム操作を中断しないで即時訂正する必要があります。

ディレクトリー・サブツリー内のオブジェクトを再利用するには、次のコマンドを使用してください。

```
RCLLNK OBJ('/MyApplicationInstallDirectory') SUBTREE(*ALL)
```

この場合、MyApplicationInstallDirectory は問題のオブジェクトを含むディレクトリーの名前です。

例: 「ルート」(/)、QOpenSys、およびマウントされたユーザー定義ファイル・システムのすべての損傷したオブジェクトを検索する

この状態では、ディスク障害が多くのオブジェクトの損傷の原因となっています。 損傷したオブジェクトの適切なリカバリー方法を決定する前に、それらのオブジェクトを識別する必要があります。

損傷したオブジェクトを識別する解決方法は必要ですが、アクションはまだ取らないでください。通常のファイル・システム操作を中断してはいけません。

損傷したオブジェクトを識別するには、次のコマンドを使用してください。

```
RCLLNK OBJ('/') SUBTREE(*ALL) DMGOBJOPT(*KEEP *KEEP)
```

さらに、このコマンドは、損傷したオブジェクトの識別時に、その損傷したオブジェクト以外の問題も修正します。

例: 「ルート」(/)、QOpenSys、およびマウントされたユーザー定義ファイル・システムのすべての損傷したオブジェクトを削除する

この状態では、ディスク障害によって多くのオブジェクトが損傷する原因となっています。 オブジェクトのバックアップ・コピーをメディアから復元できるように、損傷したオブジェクトを削除する必要があります。

損傷したオブジェクトを削除するには、次のコマンドを使用してください。

```
RCLLNK OBJ('/') SUBTREE(*ALL) DMGOBJOPT(*DELETE *DELETE)
```

損傷したオブジェクトは通常のファイル・システム操作を中断することなく削除されます。さらに、損傷したオブジェクトの削除中に、損傷以外の問題も訂正されます。

例: 複数の RCLLNK コマンドを実行して、「ルート」(/)、QOpenSys、およびマウントされたユーザー定義ファイル・システムのすべてのオブジェクトを即時再利用する

この状態では、ルーチン・システム・メンテナンスの一部として、「ルート」 (/)、QOpenSys、およびマウントされたユーザー定義ファイル・システムのすべてのオブジェクトが再利用されます。 追加のシステム・メンテナンスを完了できるように、再利用操作はできるだけ早く完了したいと考えます。

再利用操作を別のグループに分けることによって、複数の RCLLNK コマンドが並行して実行できるので、再利用操作を早急に終了できます。

キー・システム・ディレクトリーやその他の最上位ディレクトリーに対して複数の再利用操作を実行するには、以下のコマンドを (それぞれ別個のジョブまたはスレッド中で) 実行してください。

```
RCLLNK OBJ('/') SUBTREE(*DIR)
RCLLNK OBJ('/tmp') SUBTREE(*ALL)
RCLLNK OBJ('/home') SUBTREE(*ALL)
RCLLNK OBJ('/etc') SUBTREE(*ALL)
RCLLNK OBJ('/usr') SUBTREE(*ALL)
RCLLNK OBJ('/QIBM') SUBTREE(*ALL)
RCLLNK OBJ('/QOpenSys') SUBTREE(*ALL)
RCLLNK OBJ('/IaspName') SUBTREE(*ALL)
RCLLNK OBJ('/dev') SUBTREE(*ALL)
RCLLNK OBJ('/OtherTopLevelDirectories') SUBTREE(*ALL)
```

この場合、OtherTopLevelDirectories は再利用したい他のディレクトリーです。

プログラミング・サポート

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・プログラミング・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、IBM i 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。

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。

関連概念

16 ページの『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。

CL コマンドによるデータのコピー

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

CPYTOSTMF および CPYFRMSTMF 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(*DBLQUOTE) 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、システム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。

関連情報

ストリーム・ファイルへのコピー (CPYTOSTMF) コマンド

ストリーム・ファイルからのコピー (CPYFRMSTMF) コマンド

インポート・ファイルへのコピー (CPYTOIMPF) コマンド

インポート・ファイルからのコピー (CPYFRMIMPF) コマンド

制御言語 (CL)

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・メンバーをコピーする場合は、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、そこからデータを読み取り、(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。

関連情報

open()-- ファイルのオープン API

read()-- ディスクリプターからの読み取り API

write()-- ディスクリプターへの書き込み API

統合ファイル・システム API

データ転送機能を使用したデータのコピー

IBM i Access Family ライセンス・プログラム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、IBM i Access Family をインストールする必要があり、 PC リソースと i5/OS・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。

IBM i Access Family を PC とシステム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。

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:

システム上でデータベース・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、次のステップに従ってください。

1. システムへの接続を確立します。
2. i5/OS 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. 「IBM i Access for Windows」ウィンドウから、「**System iからのデータ転送**」をクリックします。
4. 転送元のシステムを選択します。
5. i5/OS 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 ファイルの詳細**」をクリックして、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. 「**System i からデータを転送 (Transfer data from System i)**」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。「System iへのデータ転送 (Data Transfer to System i)」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。「System iからのデータ転送 (Data Transfer From System i)」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 i 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:

システム上でストリーム・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、次の手順を実行してください。

1. システムへの接続を確立します。
2. i5/OS 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. 「IBM i Access for Windows」ウィンドウから、「**System iへのデータ転送**」をクリックします。
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システムを選択します。
6. 「**System iにデータを転送 (Transfer data to System i)**」をクリックして、ファイル転送を実行します。

   注: システム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、「System iへのデータ転送 (Data Transfer to System i)」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、「System iからのデータ転送 (Data Transfer from System i)」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、「**System i へのデータ転送 (Transfer data to System i)**」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。「System iへのデータ転送 (Data Transfer to System i)」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。「System iからのデータ転送 (Data Transfer From System i)」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 i 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。

関連資料

131 ページの『形式記述ファイルの作成』
システム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、「System iへのデータ転送 (Data Transfer to System i)」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。

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:

新たに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、以下の指示に従ってください。

1. システムへの接続を確立します。

2. i5/OS 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. 「IBM i Access for Windows」ウィンドウから、**「System iへのデータ転送」**をクリックします。
4. 「System iへのデータ転送 (Data Transfer to System i)」アプリケーションの「ツール」を開きます。
5. **「System i データベース・ファイルの作成 (Create System i database file)」**をクリックします。

   新規の System i 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。System i ファイルの基になる PC ファイルの名前、作成する System i 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必要があります。 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。続いてシステム上にデータベース・ファイル定義を作成します。

形式記述ファイルの作成:

システム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、「System iへのデータ転送 (Data Transfer to System i)」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、「System iからのデータ転送 (Data Transfer from System i)」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. 「System iからのデータ転送 (Data Transfer from System i)」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。

   このデータ転送は、System i機能に対して使用できるようになりました。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。

関連資料

129 ページの『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』
システム上でデータベース・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、次のステップに従ってください。

130 ページの『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』
システム上でストリーム・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、次の手順を実行してください。

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。

オブジェクトのコピー (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。

関連情報

オブジェクトのコピー (CPY) コマンド

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多くのアプリケーション・プログラミング・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 種類あり、どちらの関数も Integrated Language Environment (ILE) C を使用して作成されたプログラム内で使用できます。

- i5/OS オペレーティング・システムに組み込まれている統合ファイル・システム C 関数。
- ILE C 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、137 ページの表 14を参照してください。

統合ファイル・システム機能が作動するのは、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た場合のみです。以下の API がサポートされています。

表 13. 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の変更
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()	新しいファイルを作成する、または既存のファイルに上書きする
creat64()	新しいファイルを作成する、または既存のファイルに上書きする (大容量ファイル対応)
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 (大容量ファイル対応)
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

表 13. 統合ファイル・システム API (続き)

関数	説明
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
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
getpwnam()	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)
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
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

表 13. 統合ファイル・システム API (続き)

関数	説明
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
pread()	オフセットを指定して記述子から読み取る
pread64()	オフセットを指定して記述子から読み取る (ラージ・ファイル使用可能)
pwrite()	オフセットを指定して記述子に書き込む
pwrite64()	オフセットを指定して記述子に書き込む (ラージ・ファイル使用可能)
QjoEndJournal()	ジャーナル処理の終了
QjoRetrieveJournal Information()	ジャーナル情報を検索する
QjoRetrieveJournalEntries()	ジャーナル項目の検索
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()	ジャーナル処理の開始
QlgAccess()	ファイルのアクセス可能性を判別する (NLS 化パス名を使用)
QlgAccessx()	ユーザーのクラスのファイルのアクセス可能性を判別する (NLS 化パス名を使用)
QlgChdir()	現行ディレクトリーを変更する (NLS 化パス名を使用)
QlgChmod()	ファイル許可を変更する (NLS 化パス名を使用)
QlgChown()	ファイルの所有者およびグループを変更する (NLS 化パス名を使用)
QlgCreat()	新規ファイルを作成するか、既存のファイルに上書きする (NLS 化パス名を使用)
QlgCreat64()	新規ファイルを作成するか、既存のファイルに上書きする (ラージ・ファイル使用可能、NLS 化パス名を使用)
QlgCvtPathToQSYSObjName()	統合ファイル・システムのパス名を QSYS オブジェクト名に解決する (NLS 化パス名を使用)
QlgGetAttr()	オブジェクトのシステム属性を入手する (NLS 化パス名を使用)
QlgGetcwd()	現行ディレクトリーのパス名を入手する (NLS 化パス名を使用)
QlgGetPathFromFileID()	オブジェクトのパス名をファイル ID から入手する (NLS 化パス名を使用)
QlgGetpwnam()	ユーザー名についてのユーザー情報を入手する (NLS 化パス名を使用)
QlgGetpwnam_r()	ユーザー名についてのユーザー情報を入手する (NLS 化パス名を使用)
QlgGetpwuid()	ユーザー ID についてのユーザー情報を入手する (NLS 化パス名を使用)

表 13. 統合ファイル・システム API (続き)

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

表 13. 統合ファイル・システム API (続き)

関数	説明
QP0FPTOS()	各種ファイル・システム機能を実行する
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
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
Qp0zPipe()	ソケットを使ったプロセス間チャネルを作成する
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)

表 13. 統合ファイル・システム API (続き)

関数	説明
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
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、i5/OS ソケットにも使用されます。

表 14. 統合ファイル・システム出口プログラム

関数	説明
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	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。
オープン時の統合ファイル・システム・スキャン出口プログラム	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。
記憶域の空きの保管	*STMF 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。 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。

関連概念

25 ページの『ファイル・システム』
ファイル・システムは、論理単位として編成された記憶域の特定のセグメントへのアクセスを提供します。システムの論理単位とは、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。

関連資料

144 ページの『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。

128 ページの『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・メンバーをコピーする場合は、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、そこからデータを読み取り、(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。

関連情報

アプリケーション・プログラミング・インターフェース (API)

ILE C 関数

ILE C 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 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。つまり、ILE C 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。

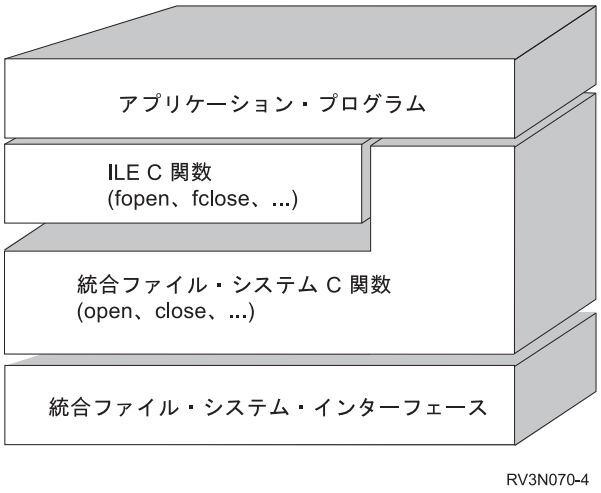

図 10. ILE C 関数の統合ファイル・システム・ストリーム入出力関数の使用

統合ファイル・システム・ストリーム入出力での ILE C 関数の使用についての詳細は、WebSphere® Development Studio: ILE C/C++ Programmer's Guide の資料を参照してください。 ILE C の個々の 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・システムによって、「ルート」 (/)、QOpenSys、およびユーザー定義ファイル・システムで約 1 TB (1 TB は約 1 099 511 627 776 バイト) までのストリーム・ファイル・サイズが使用可能となります。

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。

以下の状態は、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。

- 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。
- 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。例えば、アプリケーションがコンパイル時に _LARGE_FILES を定義した場合、stat() API に対する呼び出しは stat64() API にマップされ、stat() 構造は 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。

関連情報

統合ファイル・システム API

stat64()--Get File Information (Large File Enabled) API

stat()--Get File Information API

API のパス名規則

統合ファイル・システムまたは ILE C 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。 これは、API でパス名を指定する際の考慮すべき規則の要約です。

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。
- 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。

- i5/OS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。たとえば、以下のようにします。

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

- 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
  ```

- 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。
- 統合ファイル・システム・コマンドのパス名とは異なり、アスタリスク (*)、疑問符 (?)、単一引用符 (')、引用符 (")、および波形記号 (~) には特別の意味はありません。これらの文字は、名前の中で単に別の文字として扱われます。 i5/OS の特殊値と混同しないようにするため、パス名の先頭文字を単一アスタリスク (*) にはしないでください。この規則の例外となる 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)。

関連資料

82 ページの『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。

関連情報

ジャーナル処理の終了 (QjoEndJournal) API

ジャーナル処理の開始 (QjoStartJournal) API

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 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。

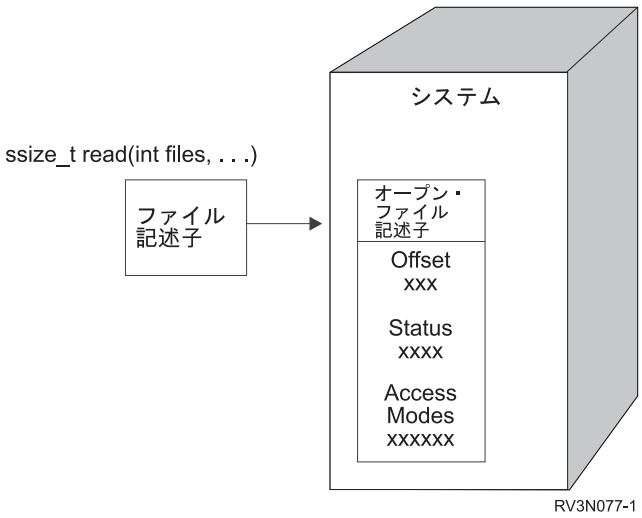

図 11. ファイル記述子とオープン・ファイル記述

ILE C 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 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。そのためには、givedescriptor()、takedescriptor()、sendmsg()、または recvmsg() 関数を使用して、ジョブ間でファイル記述子を渡します。

関連情報

givedescriptor()-- ディスクリプター・アクセスを別のジョブに渡す API

takedescriptor()-- 別のジョブからソケット・アクセスを受け取る API

sendmsg()-- ソケットによるメッセージ送信

recvmsg()-- ソケットによるメッセージ受信

ソケット・プログラミング

ソケット API

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 サーバーおよびファイル・サーバーのジョブについては、特別な考慮事項があります。通常これらのジョブは、必ずしもジョブのためのユーザー・プロファイルを所有しないユーザーのために機能を実行します。 NFS 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。他のファイル・サーバー・ジョブは、サーバーに接続されたユーザーの要求を実行します。

システムにおける権限は、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。

関連情報

chmod()-- ファイル権限変更 API

fchmod()-- ディスクリプターによるファイル権限変更 API

umask()-- ジョブの権限マスクの設定 API

統合ファイル・システム API

セキュリティー参照

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。

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。

関連情報

ソケット・プログラミング

write()-- ディスクリプターへの書き込み API

read()-- ディスクリプターからの読み取り API

close()-- ファイルまたはソケット・ディスクリプターのクローズ API

unlink()-- ファイル・リンク解除 API

リンクの除去 (RMVLNK) コマンド

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各文字が 16 ビット形式に変換され、すべての文字が標準のコード化表現となります。統合ファイル・システムのインターフェースによる入力で名前を使用する際に、呼び出し側により使用されるコード・ページから、適切なコード化形式に変換されます。出力時の名前変換において、元の名前で使用している文字が変換後のコード・ページに含まれていない場合は結果がエラーとなるか、印刷不能な文字で情報が返されます。

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実行しないでください。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。

関連概念

17 ページの『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。

112 ページの『自動名前変換の概要』
「ルート」(/) や CASE (*MONO) により作成される UDFS などの大文字と小文字を区別しないファイル・システムは、Unicode 規格 4.0 形式で格納される名前をサポートします。システムは、名前に使用されている追加の文字をサポートするために自動名前変換を実行します。

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。

関連情報

open()-- ファイルのオープン API

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。

注: このコーディング例を使用することによって、 165 ページの『コードに関するライセンス情報および特記事項』の条件に合意することになります。

```
#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;
}
```

System i 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。

フォルダーの作成

フォルダーを作成するには、次のステップに従ってください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。
2. 新しいフォルダーの追加先となるファイル・システムまたはファイル・システム内のフォルダーを右クリックして、「**新規フォルダー**」を選択します。
3. フォルダー名とフォルダーの属性を「**新規フォルダー**」ダイアログで指定します。
   - 「ルート」(/)、QOpenSys、またはユーザー定義ファイル・システムに作成するフォルダーの場合、「**名前変更とリンク解除の制限**」、「**フォルダーで作成されたオブジェクトの監査**」、および「**フォルダーで作成されたオブジェクトのスキャン**」のフィールドの値を指定できます。
   - 他のファイル・システムに作成するフォルダーの場合、「**フォルダーで作成されたオブジェクトの監査**」フィールドの値を指定できます。
4. 「**OK**」をクリックします。

System i プラットフォーム上にフォルダーを作成する際、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。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。

関連タスク

157 ページの『オブジェクトをスキャンするかどうかの設定』
「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムで、オブジェクトをスキャンするかどうかを指定できます。以下のステップに従って、スキャン・オプションを設定します。

関連情報

ジャーナル管理

ファイルまたはフォルダーの除去

ファイルまたはフォルダーを除去するには、次のステップに従ってください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。除去する対象が表示されるまで、展開を続けます。
2. 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。 確認パネルが表示され、選択した削除対象項目のリストが示されます。削除を取り消す項目があれば、消去します。それから、確認パネルの「**削除**」をクリックします。

   注: フォルダーを削除する場合、フォルダーの中身もすべて削除されます。

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。 ただし、別のファイル・システムにオブジェクトを移動させると、そのオブジェクトが現在保管されているファイル・システムの特性を利用することはできなくなる場合があります。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。

また、以下の状態について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。例えば、QDLS と QOPT のファイル・システムは、階層ファイル・システム (HFS) API とコマンドをサポートすることによって、文書およびフォルダーの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、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。

System i ナビゲーター では、オブジェクトを新しい場所にドラッグすることによって、ファイルやフォルダーを別のファイル・システムに移動できます。あるいは、コピー・アンド・ペーストやカット・アンド・ペースト機能を使用して移動することも可能です。

CL コマンドを使用して別のファイル・システムにオブジェクトを移動するには、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. オブジェクトの移動 (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化しないためです。しかし、オブジェクトのコピー (CPY) コマンドを使用し、OWNER(*KEEP) パラメーターを使用することによってもオブジェクトの所有権を保つ

ことができます。この方法が有効なのは、ファイル・システムがオブジェクトの所有者の設定をサポートしている場合だけであることに注意してください。 MOV または CPY を使用する場合には、次のことに注意してください。

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

このため、オブジェクトを元のファイル・システムに戻そうとしても、単に移動またはコピー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。

関連概念

25 ページの『ファイル・システム』
ファイル・システムは、論理単位として編成された記憶域の特定のセグメントへのアクセスを提供します。システムの論理単位とは、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。

関連資料

27 ページの『ファイル・システムの比較』
これらの表は、各ファイル・システムの機能と制限事項の要約です。

関連情報

ディレクトリーの作成 (CRTDIR) コマンド

オブジェクトの移動 (MOV) コマンド

オブジェクトのコピー (COPY) コマンド

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、次のステップに従ってください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。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。
2. 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。
3. 「**許可**」ダイアログで「**追加**」をクリックします。
4. 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。
5. 「**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。
6. 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。
7. 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。
8. 「**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

System i 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、System i プラットフォームと 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、次のステップに従ってください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」の順に展開します。
2. 「**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。
3. 自動的に変換されるようにするファイル拡張子を、「**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。
4. 自動的に変換されるようにするすべてのファイル拡張子について、ステップ 3 を繰り返します。
5. 「**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別のシステムに送信するには、次のステップに従ってください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。送信する対象が表示されるまで、展開を続けます。
2. 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。
3. 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。
4. 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。
5. 「**OK**」 をクリックして、ファイルまたはフォルダーを送信します。

関連タスク

『ファイルまたはフォルダー送信のためのオプション変更』

システムからファイルまたはフォルダーを別のシステムに送信する際にサブフォルダーを含めるか、また既存のファイルを置き換えるかどうかを定義できます。さらに、システムが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できます。ファイル送信のためのオプションを変更するには、次のステップに従ってください。

ファイルまたはフォルダー送信のためのオプション変更

システムからファイルまたはフォルダーを別のシステムに送信する際にサブフォルダーを含めるか、また既存のファイルを置き換えるかどうかを定義できます。さらに、システムが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できます。ファイル送信のためのオプションを変更するには、次のステップに従ってください。

1. 『他のシステムへのファイルまたはフォルダーの送信』のステップを完了します。
2. 「**オプション**」タブをクリックします。デフォルト・オプションでは、ファイルをパッケージして送信するときにサブフォルダーを組み込んで、既存のファイルを送信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またはフォルダーを送信するか、あるいは「**スケジュール**」をクリックしてファイルまたはフォルダーを送信する時刻を設定します。
7. 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。 スケジューラー機能によって、都合のよい日時に柔軟に作業が実行できるようになります。

ファイル共用の作成

*ファイル共用*は、i5/OS NetServer が System i 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、System i プラットフォーム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、次のステップに従ってください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。
2. 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。
3. 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。
4. 「**新規共用**」を選択します。
5. 表示される「ファイル共用」ダイアログで、新規ファイル共用の属性を指定し、「**OK**」をクリックします。

ファイル共用の変更

*ファイル共用*は、i5/OS NetServer が System i 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、System i プラットフォーム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、次のステップに従ってください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。
2. 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。
3. 変更するファイル共用を含むフォルダーを右クリックし、「**共用**」を選択します。
4. 変更するファイル共用の名前を選択します。
5. 表示される「ファイル共用」ダイアログで、ファイル共用の属性を変更し、「**OK**」をクリックして変更を確定します。

ファイル共用の解除

*ファイル共用*は、i5/OS NetServer が System i 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、System i プラットフォーム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。 System i ナビゲーター を使用して既存のファイル共用を停止することができます。

ファイル共用を解除するには、次のステップに従ってください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。
2. 共用を停止するファイル共用が含まれているファイル・システムを展開します。
3. 共用を停止する共用ディレクトリーを右クリックし、「**共用**」 → 「**共用の停止**」の順に選択します。
4. 次に表示される「共用の停止 (Stop sharing)」ウィンドウで、「**OK**」をクリックします。

新規のユーザー定義ファイル・システムの作成

ユーザー定義ファイル・システム (UDFS) とは、その属性を作成して定義するファイル・システムのことです。UDFS はシステム上の補助記憶域プール (ASP)、および独立 ASP 中に存在します。

新しいユーザー定義ファイル・システム (UDFS) を作成するには、次のステップに従ってください。

1. Systems Director Navigator for i で、**「ユーザー接続」 → *ご使用のシステム* → 「ファイル・システム」 → 「統合ファイル・システム」 → 「ルート」 → 「dev」**の順に展開します。
2. 新しい UDFS を入れる補助記憶域プール (ASP) をクリックします。
3. 「ファイル」メニューから**「新規の UDFS」**を選択します。
4. 「新規のユーザー定義ファイル・システム」ダイアログで、UDFS 名、記述 (オプション)、監査値、デフォルトのファイル・フォーマット、デフォルトのスキャン属性、デフォルトのディスク・スペース割り振り、デフォルトのメモリー割り振り、および新規の UDFS 中のファイルが大文字と小文字を区別するファイル名をもつかどうかを指定します。

   注: **「デフォルトのディスク・スペース割り振り」**および**「デフォルトのメモリー割り振り」**は、V6R1 またはそれ以降のリリースでのみ使用できます。

ユーザー定義ファイル・システムのマウント

UDFS に保管されたデータにアクセスまたは表示するには、IPL する度に UDFS をマウントする必要があります。

UDFS をマウントすると、そのフォルダー階層のマウント・ポイントの下に存在するすべてのファイル・システム、ディレクトリー、またはオブジェクトが隠されます。これで、UDFS をアンマウントするまで、それらのファイル・システム、ディレクトリー、またはオブジェクトがアクセス不能になります。統合ファイル・システム中のすべてのデータへのアクセスを保守するようするには、UDFS を空きフォルダーにマウントしてください。UDFS をマウントした後では、UDFS 中のファイルはそのフォルダー内からアクセスできます。フォルダーに対する変更は、隠されたフォルダーに対してではなく、UDFS に対しての変更となります。

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。

ユーザー定義ファイル・システム (UDFS) をマウントするには、次のステップに従ってください。

1. Systems Director Navigator for i で、**「ユーザー接続」 → *ご使用のシステム* → 「ファイル・システム」 → 「統合ファイル・システム」 → 「ルート」 → 「dev」**の順に展開します。
2. マウントしたい UDFS が含まれている補助記憶域プール (ASP) をクリックします。
3. Systems Director Navigator for i の右側ペインの **「UDFS 名」**欄で、マウントする UDFS を右クリックします。
4. **「マウント」**を選択します。
5. 表示される「UDFS のマウント」ダイアログで、マウントするディレクトリーのパス、アクセスのタイプ (読み取り専用、読み取り/書き込み)、およびユーザーとグループ ID の設定を許可するかどうかを指定します。それから、**「OK」**をクリックします。

ドラッグしたい場合、同じシステム上の統合ファイル・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。

ユーザー定義ファイル・システムのアンマウント

UDFS をマウントすると、そのフォルダー階層のマウント・ポイントの下に存在するすべてのファイル・システム、ディレクトリー、またはオブジェクトが隠されます。これで、UDFS をアンマウントするまで、それらのファイル・システム、ディレクトリー、またはオブジェクトがアクセス不能になります。

ユーザー定義ファイル・システム (UDFS) をアンマウントするには、次のステップに従ってください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」 → 「**ルート**」 → 「**dev**」の順に展開します。
2. アンマウントしたい UDFS が含まれている補助記憶域プール (ASP) をクリックします。
3. System i ナビゲーター の右側ペインの「**名前**」欄で、アンマウントする UDFS を右クリックします。
4. 「**アンマウント**」を選択します。 アンマウント確認パネルが表示され、選択したアンマウント対象の UDFS が示されます。
5. アンマウントを取り消す UDFS があれば、消去します。それから、「**アンマウント**」をクリックします。

動的にマウントされたファイル・システムの処理

「動的マウント情報」機能を使用することにより、動的にマウントされるファイル・システムのうち現在マウントされているものを確かめ、それらのプロパティーを表示し、それらのいずれかをアンマウントすることができます。

この機能を使うには、次のステップを実行してください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。
2. 「**統合ファイル・システム**」を右クリックします。
3. ポップアップ・メニューで、「**動的マウント情報**」を選択します。
4. 「動的マウント情報」ウィンドウが開き、現在マウントされているすべてのファイル・システムのリストが示されます。このウィンドウに、マウントされたファイル・システムの名前、そのファイル・システムがマウントされたリモート・システム、およびマウント・タイプが表示されます。サポートされるマウント・タイプは、ユーザー定義ファイル・システム (UDFS)、ネットワーク・ファイル・システム・バージョン 2 (NFSv2)、ネットワーク・ファイル・システム・バージョン 3 (NFSv3)、およびネットワーク・ファイル・システム・バージョン 4 (NFSv4) です。リストに示されているファイル・システムのいずれもアンマウント可能で、特定のファイル・システムのプロパティーを表示できます。

   注: NFSv4 は V6R1 と以降のリリースでのみ使用できます。

   - ファイル・システムをアンマウントするには、リストからファイル・システムを選択し、「**アンマウント**」をクリックします。「アンマウントの確認」ウィンドウが開きます。表示されるファイル・システムが、アンマウントするシステムかどうかを確認します。アンマウントを取り消す項目があれば、消去します。それから、「**アンマウント**」をクリックし、操作を確定します。
   - ファイル・システムのプロパティーを表示するには、リストからファイル・システムを選択し、「**プロパティー**」をクリックします。「マウント・プロパティー」ウィンドウが開きます。
     - ユーザー定義ファイル・システム (UDFS) の場合、「マウント・プロパティー」ウィンドウには「**一般**」タブが含まれています。ここに表示されるプロパティーは、名前、UDFS がマウントされるパス、マウント・タイプ、マウント時刻、ファイル・システムが読み取り専用かどうか、ユーザーおよびグループ設定を許可するかどうかです。

– ネットワーク・ファイル・システム (NFS) の場合、「マウント・プロパティー」ウィンドウには「**一般**」タブと「**拡張**」タブが含まれています。「**一般**」タブに表示されるプロパティーは、名前、リモート・サーバー名、NFS がマウントされるパス、マウント・タイプ、マウント時刻、ファイル・システムが読み取り専用かどうか、およびユーザーとグループ設定を許可するかどうかです。「**拡張**」タブに表示されるプロパティーは、マウント・タイプ、タイムアウト値、読み取りバッファー・サイズ、書き込みバッファー・サイズ、再試行回数、再送信の試行回数、正規オブジェクト属性最小/最大保持時間、フォルダー属性最小/最大保持時間、オープン時に属性の強制リフレッシュをするかどうか、属性および名前のキャッシングを許可するかどうかです。

オブジェクトをスキャンするかどうかの設定

「ルート」(/)、QOpenSys、およびユーザー定義ファイル・システムで、オブジェクトをスキャンするかどうかを指定できます。以下のステップに従って、スキャン・オプションを設定します。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。対象のオブジェクトが表示されるまで、展開を続けます。
2. 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。
3. 「**セキュリティー**」タブをクリックします。
4. 「**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。

オプションに関する詳細は、以下のセクション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。

- 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。

- 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:

1. 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。

2. 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。

オブジェクトのチェックイン

ポップアップ・メニューの「**チェックイン**」オプションまたは「**プロパティー**」ページを使用して、フォルダー内のファイルやすべての適格なオブジェクトをチェックインすることができます。

以下の要件を満たすオブジェクトのチェックインが可能です。

- オブジェクト・タイプがオブジェクトのチェックイン (CHKIN) コマンドにサポートされている。
- オブジェクトが現在チェックアウトされている。

ポップアップ・メニューよりオブジェクトをチェックインするには、次のステップに従ってください。

注: この方式は System i ナビゲーター V6R1 またはそれ以降のシステムでのみ使用できます。それより前のバージョンの場合は、「**プロパティー**」ページ 方式を使用してください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。
2. チェックインするファイル、または中身をすべてチェックインするフォルダーを右クリックします。
3. ポップアップ・メニューで、「**チェックイン**」を選択します。

「**プロパティー**」ページよりオブジェクトをチェックインするには、次のステップに従ってください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。
2. チェックインするファイル、または中身をすべてチェックインするフォルダーを右クリックします。

   注: フォルダーの全内容をチェックインする機能は、V6R1 またはそれ以降のシステムでのみ使用できます。
3. ポップアップ・メニューで、「**プロパティー**」を選択します。
4. 表示される「**プロパティー**」ウィンドウで、「**使用**」タブをクリックします。
5. ファイルまたはフォルダー内の全オブジェクトをチェックインします。
   - ファイルをチェックインするには、「**チェックイン**」をクリックします。
   - フォルダー内の全オブジェクトをチェックインするには、「**チェックイン**」をクリックします。確認ウィンドウが表示されます。「**続行**」をクリックし、チェックイン操作を続けます。チェックインするオブジェクトの数によっては、この操作の完了に長時間かかる場合があります。

オブジェクトのチェックアウト

ポップアップ・メニューの「**チェックアウト**」オプションまたは「**プロパティー**」ページを使用して、フォルダー内のファイルやすべての適格なオブジェクトをチェックアウトすることができます。

以下の要件を満たすオブジェクトのチェックアウトが可能です。

- オブジェクト・タイプがオブジェクトのチェックアウト (CHKOUT) コマンドにサポートされている。
- オブジェクトが現在チェックインされている。

ポップアップ・メニューよりオブジェクトをチェックアウトするには、次のステップに従ってください。

注: この方式は System i ナビゲーター V6R1 またはそれ以降のシステムでのみ使用できます。それより前のバージョンの場合は、「**プロパティー**」ページ 方式を使用してください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。
2. チェックアウトするファイル、または中身をすべてチェックアウトするフォルダーを右クリックします。
3. ポップアップ・メニューで、「**チェックアウト**」を選択します。

「**プロパティー**」ページよりオブジェクトをチェックアウトするには、次のステップに従ってください。

1. System i ナビゲーター で、「**ユーザー接続**」 → ***ご使用のシステム*** → 「**ファイル・システム**」 → 「**統合ファイル・システム**」の順に展開します。
2. チェックアウトするファイル、または中身をすべてチェックアウトするフォルダーを右クリックします。

   注: フォルダーの全内容をチェックアウトする機能は、V6R1 またはそれ以降のシステムでのみ使用できます。
3. ポップアップ・メニューで、「**プロパティー**」を選択します。
4. 表示される「**プロパティー**」ウィンドウで、「**使用**」タブをクリックします。
5. ファイルまたはフォルダー内の全オブジェクトをチェックアウトします。
   - ファイルをチェックアウトするには、「**チェックアウト**」をクリックします。
   - フォルダー内の全オブジェクトをチェックアウトするには、「**チェックアウト**」をクリックします。確認ウィンドウが表示されます。「**続行**」をクリックし、チェックアウト操作を続けます。チェックアウトするオブジェクトの数によっては、この操作の完了に長時間かかる場合があります。

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。

RPC には、外部データ表示 (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 System i プラットフォーム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。

ネットワーク選択 API

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。

以下の 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。

関連情報

API ファインダー

名前からアドレスへの変換 API

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。

関連情報

API ファインダー

eXternal Data Representation (XDR) API

以下の API によって、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。

API	説明
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。

関連情報

API ファインダー

認証 API

以下の API は TI-RPC アプリケーションに対する認証を提供します。

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。

関連情報

API ファインダー

トランスポート独立 RPC (TI-RPC) API

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。

関連情報

API ファインダー

TI-RPC 単純化 API

以下の単純化 API は、使用す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。

関連情報

API ファインダー

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・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()	汎用クライアント・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。

関連情報

API ファインダー

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。

関連情報

API ファインダー

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選択できます。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。

関連情報

API ファインダー

その他の 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。
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。

関連情報

API ファインダー

統合ファイル・システムの関連情報

統合ファイル・システムのトピック・コレクション関連の情報が、各製品マニュアルやその他の Information Center のトピック・コレクションで提供されています。以下の PDF ファイルのいずれも表示または印刷できます。

マニュアル

- i5/OS Network File System Support この資料は、一連の実アプリケーションを通してネットワーク・ファイル・システムを説明しています。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。

- WebSphere Development Studio: ILE C/C++ Language Reference この資料には、System i プラットフォームでの ILE C プログラムの設計、編集、コンパイル、実行、およびデバッグに必要な情報が記載されています。

- APPC プログラミング この資料では、System i プラットフォーム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。

- Recovering your system この資料には、System i プラットフォームのリカバリーおよび可用性オプションに関する一般的な情報が記載されています。

その他の情報

- **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 System i ソリューションをインプリメントすることができます。 経験事例の統合ファイル・システムのバックアップは、ファイルおよびファイル・システムと関連しています。
- 制御言語
- i5/OS グローバリゼーション
- アプリケーション・プログラミング・インターフェース
- ジャーナル管理
- コミットメント制御
- セキュリティー参照

コードに関するライセンス情報および特記事項

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。

強行法規で除外を禁止されている場合を除き、IBM、そのプログラム開発者、および供給者は「プログラム」および「プログラム」に対する技術的サポートがある場合にはその技術的サポートについて、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。

いかなる場合においても、IBM および IBM のサプライヤーならびに IBM ビジネス・パートナーは、その予見の有無を問わず発生した以下のものについて賠償責任を負いません。

1. データの喪失、または損傷。
2. 直接損害、特別損害、付随的損害、間接損害、または経済上の結果的損害
3. 逸失した利益、ビジネス上の収益、あるいは節約すべかりし費用

国または地域によっては、法律の強行規定により、上記の責任の制限が適用されない場合があります。

付録. 特記事項

本書は米国 IBM 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。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒242-8502
神奈川県大和市下鶴間1623番14号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

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 YBWA
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、IBM 機械コードのご使用条件、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。

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。サンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

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。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

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「統合ファイル・システム」には、プログラムを作成するユーザーが IBM i5/OS 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

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標または登録商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

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。

使用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

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。

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21