IBM Systems - iSeries

統合操作環境
i5/OS PASE

バージョン 5 リリース 4

IBM

IBM Systems - iSeries

統合操作環境
i5/OS PASE

バージョン 5 リリース 4

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、 65 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 i5/OS (製品番号 5722-SS1) バージョン 5、リリース 4 モディフィケーション 0 に適用されます。また、改訂版で断りがない限り、それ以降のすべてのリリースおよびモディフィケーションに適用されます。このバージョンは、すべての RISC モデルで稼働するとは限りません。また CISC モデルでは稼働しません。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM Systems - iSeries
Integrated operating environments
i5/OS PASE
Version 5 Release 4

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2006.2

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

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

目次

i5/OS PASE

IBM® i5/OS™ ポータブル・アプリケーション・ソリューション環境 (i5/OS PASE) は最小限の労力で IBM AIX® アプリケーションを IBM iSeries™ サーバーに移植することを可能にします。

AIX または Linux® などのオペレーティング・システムの管理の複雑さに煩わされることなく、選択したアプリケーションを実行できる統合されたランタイム環境が i5/OS PASE によって提供されます。 i5/OS PASE には、強力なスクリプト環境を提供する、業界標準のシェルやユーティリティーも備えられています。

注: コード例を使用することにより、 62 ページの『コードに関する特記事項』 の条件に同意することになります。

V5R4 の新機能

このページでは、このリリースで i5/OS PASE に加えられた変更点を取り上げています。

- i5/OS PASE for V5R4M0 は、AIX 5.2 for i5/OS PASE V5R3M0 ではなく、AIX 5.3 から派生したものです。
- 以下のコンパイラー製品は、i5/OS PASE の V5R4M0 上での実行のサポートが発表されています。
  - IBM XL C/C++ Enterprise Edition for AIX バージョン 7.0
  - IBM XL C Enterprise Edition for AIX バージョン 7.0
  - IBM XL Fortran Enterprise Edition for AIX バージョン 9.1
- 以下のユーティリティーが新規追加または変更されました。
  - apt (Java™ 注釈処理ツールの QShell apt コマンドを実行)
  - pack200 (Java アーカイブ圧縮ツールの QShell pack200 コマンドを実行)
  - unpack200 (Java アーカイブ解凍ツールの QShell unpack200 を実行
- 新規および変更された i5/OS PASE ランタイム機能を以下に示します。
  - _OPEN_CCSID (CCSID for i5/OS PASE を使用して開く)
- 新しいロケールが追加されました。
- 新しい例が追加されました。

新規および変更箇所の識別方法

技術的な変更が加えられた箇所を識別しやすくするために、以下の情報が使用されています。

- » は、新規または変更された情報の開始箇所を示しています。
- « は、新規または変更された情報の終了箇所を示しています。

新機能またはこのリリースでの変更内容の他の情報については、「iSeries 最初にお読みください」を参照してください。

関連概念

11 ページの『i5/OS PASE での AIX コンパイラーのインストール』
このトピックのステップに従って、i5/OS PASE に AIX コンパイラーをインストールすることができます。

関連情報

i5/OS PASE シェルおよびユーティリティー

i5/OS PASE で使用するためのランタイム機能

i5/OS PASE ロケール

印刷可能な PDF

これを使用して、この情報の PDF を表示し、印刷することができます。

本書の PDF 版を表示またはダウンロードするには、i5/OS PASE を選択します。

PDF ファイルの保存

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

1. ブラウザーで PDF を右マウス・ボタン・クリックする (上部のリンクを右マウス・ボタン・クリック)。
2. PDF をローカルに保存するオプションをクリックします。
3. PDF を保存したいディレクトリーに進みます。
4. 「**保存**」をクリックします。

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 がシステムにインストールされていることが必要です。このアプリケーションは、 Adobe Web サイト (www.adobe.com/products/acrobat/readstep.html) から無料でダウンロードできます。

---

i5/OS PASE の紹介

i5/OS ポータブル・アプリケーション・ソリューション環境 (i5/OS PASE) を使用すれば、わずかな変更で、またはまったく変更しなくても、 i5/OS 上の多数の AIX アプリケーション・バイナリーを実行することができ、さらにプラットフォーム・ソリューション・ポートフォリオを効果的に拡張することができます。

クロスプラットフォーム・アプリケーションの開発と展開は、有効なビジネス・コンピューティング環境において極めて重要な構成要素です。加えて、システムによって提供される機能が使いやすいことや統合しやすいことも重要です。 iSeries はまさに、これらすべての条件を満たしています。ビジネスは日ごとにオープン・コンピューティング環境へと移行しつつあるので、これら多様な目標を達成するには時間や費用がかかり、困難な場合があります。たとえば、AIX オペレーティング・システム上で実行し、その機能を利用する使い慣れたアプリケーションを使用したいが、 AIX と i5/OS オペレーティング・システムの両方を管理するのは面倒だと思うことがあるかもしれません。

そのようなとき、i5/OS ポータブル・アプリケーション・ソリューション環境 (i5/OS PASE) が役に立ちます。i5/OS PASE を使用すれば、わずかな変更で、またはまったく変更しなくても、 i5/OS 上の多数の AIX アプリケーション・バイナリーを実行することができ、さらにプラットフォーム・ソリューション・ポートフォリオを効果的に拡張することができます。

i5/OS PASE の概要

i5/OS PASE は、i5/OS 上で実行している AIX アプリケーション用の統合ランタイム環境です。

これは AIX のアプリケーション・バイナリー・インターフェース (ABI) をサポートしており、 AIX 共用ライブラリー、シェル、およびユーティリティーによって提供されているサポートの幅広いサブセットを提供しています。i5/OS PASE は IBM PowerPC® のマシン・インストラクションの直接処理をサポートしているため、マシン・インストラクションのエミュレートのみを行う環境の弊害がありません。

i5/OS PASE アプリケーションには、以下の特徴があります。

- C、C++、Fortran、または PowerPC アセンブラーで作成できます。
- AIX PowerPC アプリケーションと同じバイナリー実行可能フォーマットを使用します。
- i5/OS ジョブで実行します。
- ファイル・システム、セキュリティー、およびソケットといった i5/OS システム機能を使用します。

i5/OS PASE は i5/OS での UNIX® オペレーティング・システムではないので注意してください。i5/OS PASE は、ほとんどあるいはまったく変更しなくても、 i5/OS 上で AIX プログラムを実行するように設計されています。 AIX または Linux などの他の環境からのプログラムは、 i5/OS PASE で実行するための最初のステップとして、 AIX でコンパイルできるように作成する必要があります。

i5/OS PASE 統合ランタイムは、iSeries 上のライセンス内部コード カーネルで実行します。システムは、i5/OS PASE と他のランタイム環境 (ILE や Java など) の間で多くの共通 i5/OS 機能を統合します。i5/OS PASE は AIX システム呼び出しの幅広いサブセットをインプリメントします。 i5/OS PASE のシステム・サポートでは、 i5/OS PASE プログラムがアクセスできるメモリーを制御し、特権のないマシン・インストラクションだけを使用するように制限することにより、システム・セキュリティーと整合性を強化します。

最小限の労力での迅速なアプリケーション展開

多くの場合、AIX プログラムはほとんどあるいはまったく変更せずに、 i5/OS PASE で実行できます。どの程度の AIX プログラミング・スキルが必要かは、 AIX プログラムの設計に応じて異なります。さらに、プログラム設計で (CL コマンドなどで) i5/OS アプリケーションの統合をさらに行うことにより、アプリケーション・ユーザーが構成に向ける注意を最小限にとどめることができます。

i5/OS PASE では、 i5/OS のマーケットでの成功から益を得たいソリューション開発者のために、別の移植オプションを用意しています。 i5/OS PASE が移植時間を著しく短縮する手段を提供することにより、市場参入までの期間が改善され、ソリューション開発者は投資に見合ったものを得ることができます。

i5/OS での AIX テクノロジーの幅広いサブセット

i5/OS PASE では、以下のような AIX テクノロジーの幅広いサブセットに基づいた、アプリケーション・ランタイムがインプリメントされています。

- 標準の C および C++ ランタイム (スレッド・セーフと非スレッド・セーフの両方)
- Fortran ランタイム (スレッド・セーフと非スレッド・セーフの両方)
- `pthreads` スレッド化パッケージ
- データ変換用の `iconv` サービス
- バークレー・ソフトウェア・ディストリビューション (BSD) 同等サポート
- Motif ウィジェット・セットが含まれている X Window システム・クライアント・サポート
- 疑似端末 (PTY) サポート

アプリケーションは、i5/OS PASE によってサポートされているレベルと互換性のある AIX のレベルで実行している AIX ワークステーションで、開発およびコンパイルされます。それから、i5/OS で実行されます。

もう一つの方法として、 i5/OS PASE 環境でサポートされているコンパイラー製品の 1 つをインストールして、 i5/OS PASE 内で完全にアプリケーションを開発、コンパイル、ビルド、および実行することもできます。

i5/OS PASE には、Korn、Bourne、C の各シェルと、強力なスクリプト環境を提供する約 200 個のユーティリティーも含まれています。

i5/OS PASE は、AIX および i5/OS オペレーティング・システム用の共通プロセッサー・テクノロジーでの IBM インベストメントを使用します。 PowerPC プロセッサーは、 i5/OS モードから AIX モードへ切り替えて、 i5/OS PASE ランタイムでアプリケーションを実行します。

i5/OS PASE で実行するアプリケーションは、 i5/OS 統合ファイル・システムおよび DB2 Universal Database™ for iSeries に統合されます。それらは Java および 統合言語環境® (ILE) アプリケーションを呼び出す (または呼び出される) ことができます。一般にそれらは、セキュリティー、メッセージ処理、通信、およびバックアップとリカバリーなど、 i5/OS 操作環境のすべての面を活用できます。同時に、AIX インターフェースから派生したアプリケーション・インターフェースも活用します。

関連資料

9 ページの『AIX ソースのコンパイル』
i5/OS PASE でのインストールをサポートする AIX コンパイラー製品のうちの一つをインストールして i5/OS PASE 環境でプログラムをコンパイルすることができます。

関連情報

i5/OS PASE シェルおよびユーティリティー

i5/OS PASE の使用がアプリケーション開発において役立つケース

i5/OS PASE に対する特定アプリケーションの適正を判別するために、API 分析を使用することができます。i5/OS PASE は、一部の環境では最良のソリューションとはなりません。

i5/OS PASE では、AIX アプリケーションを iSeries サーバーへ移植する方法を、非常に柔軟に決定できます。当然のことながら、i5/OS PASE は選択できるいくつかのオプションの 1 つにすぎません。

API の分析

アプリケーションが i5/OS PASE に適しているかどうかを判別する場合、まず最初に行うのはアプリケーションの分析であり、それが使用する API、ライブラリー、およびユーティリティーと、アプリケーションが i5/OS でどれほど効率的に実行されるかを分析します。 IBM Virtual Innovation Center for Hardware は、アプリケーションを分析し障害の可能性を述べるフリーな移植評価ツール、API 分析ツールを使用してこのエリアの手助けをします。 i5/OS PASE にアプリケーションを移植する際のプロシージャーで、この分析ツールをどのように組み込むかについては、 8 ページの『i5/OS PASE で実行するプログラムの準備』を参照してください。

i5/OS PASE アプリケーションにすることが可能なものの特性

以下に、i5/OS PASE を使用するかどうかを決定する際に考慮できる、いくつかの有用なガイドラインを示します。

- **その AIX アプリケーションは高度な計算主体か?**

i5/OS PASE は、高度に最適化された数学ライブラリーを提供することによって、 iSeries サーバー上で計算主体のアプリケーションを実行するために適した環境を提供します。

- **そのアプリケーションは i5/OS PASE だけでサポートされている (または ILE で一部しかサポートされていない) 機能 (たとえば、 fork()、X Window システム、疑似端末 (PTY) サポートなど) にかなり依存しているか?**

  i5/OS PASE は fork() および exec() のサポートを提供しています。これらは、現在の i5/OS システムにはありません。(ただし、 spawn() を使用した場合は例外です。この場合、fork() 関数が exec() 関数に組み込まれます。)
- **そのアプリケーションは複雑な AIX システム・ベースのビルド・プロセスまたはテスト環境を使用しているか?**

  i5/OS PASE では AIX システム・ベースのビルド・プロセスが使えます。これは特に、新しいオペレーティング・システムに簡単には移せない、既存の複雑なプロセスがある場合に役立ちます。
- **そのアプリケーションは ASCII 文字セットに依存しているか?**

  i5/OS PASE では、これらを必要とするアプリケーションを十分にサポートしています。
- **そのアプリケーションは多くのポインター操作を行うか、または整数からポインターへの変換 (キャスト) を行うか?**

  i5/OS PASE では、パフォーマンスへの影響が少ない 32 ビットと 64 ビットの両方の AIX アドレッシング・モデルがサポートされており、整数をポインターに変換できます。

i5/OS PASE が必ずしも最善のソリューションではない場合

ILE から呼び出さなければならない、多数の呼び出し可能インターフェースを提供するコードや、以下のいずれかの特性を持つコードの場合は、一般に i5/OS PASE は適していません。

- 呼び出しごとに i5/OS PASE を開始または終了するか、またはすでにアクティブな i5/OS PASE プログラムで i5/OS PASE プロシージャーを呼び出す (Qp2CallPase API を使用して) かのいずれよりも、パフォーマンスの点で優れた呼び出しおよび戻りが必要なコード。
- ILE 呼び出し元とライブラリー・コードの間で、メモリーまたはネーム・スペースを共用する必要があるコード。 i5/OS PASE プログラムは、呼び出し元の ILE コードと暗黙的にメモリーおよびネーム・スペースを共用することはありません。 (ただし、i5/OS PASE から呼び出される ILE コードでは、i5/OS PASE メモリーを共用または使用できます。)

関連情報

API 分析ツール

IBM Virtual Innovation Center for Hardware

i5/OS PASE のインストール

このトピックの説明に従って、ご使用のサーバーに i5/OS PASE をインストールすることができます。

i5/OS PASE は、すべての iSeries サーバーで、課金なしで利用できます。 i5/OS PASE のインストールは、推奨されています。拡張ドメイン・ネーム・サーバー (DNS) や ILE C++ コンパイラーなどの一部のシステム・ソフトウェアでは、i5/OS PASE サポートが必要です。

サーバーに i5/OS PASE をインストールするには、以下のステップを実行します。

1. i5/OS コマンド行で、`GO LICPGM` と入力します。
2. 11 (ライセンス・プログラムのインストール) を選択します。

3. オプション 33 (5722SS1 - ポータブル・アプリケーション・ソリューション環境) を選択します。
4. **任意:** 追加ロケールをインストールします。

   i5/OS PASE 製品では、 i5/OS にインストールされる言語フィーチャーに関連付けられたロケール・オブジェクトのみがインストールされます。サーバーの言語フィーチャーに組み込まれていないロケールが必要であれば、追加の i5/OS 言語フィーチャーをオーダーしてインストールすることが必要です。詳しくは、『i5/OS PASE グローバリゼーション』および『i5/OS PASE ロケール (i5/OS PASE locales)』を参照してください。

i5/OS PASE にアプリケーションを移植するソフトウェア開発者のためのライセンスの注::

i5/OS PASE には、i5/OS システム上に AIX ランタイム・ライブラリーのサブセットがあります。 i5/OS に同梱されているライブラリー・コードはすべて、i5/OS のライセンスで使用できます。ただし、このライセンスは、 i5/OS PASE に同梱されていない AIX ライブラリーに対するライセンスを意味するものではありません。すべての AIX 製品のライセンスは、IBM によって個別に交付されます。

独自のアプリケーションを i5/OS PASE に移植しようとするとき、そのアプリケーションが、 i5/OS PASE に同梱されていない AIX ライブラリーに依存しているという場合があるかもしれません。そのような場合は、これらのライブラリーを i5/OS システムに移植する前に、それらのライブラリーがどのソフトウェア製品で提供されているのかを確認し、そのソフトウェア製品のライセンス許諾条件を調べる必要があります。場合によっては、IBM やサード・パーティーと連絡をとり、アプリケーションが依存する付加的なミドルウェアを i5/OS システムに移植する必要があります。移植を行うときは、それを開始する前に、移植しようとしているコードに関係するすべてのライセンス許諾条件をよく調べてください。 IBM に帰属すると思われるライブラリーに関してライセンス許諾条件の情報が必要な場合は、 IBM の営業担当員、いずれかの IBM ポーティング・センター、ロチェスターの Custom Technology Center、または PartnerWorld® for Developers に相談してください。

関連概念

54 ページの『グローバリゼーション』
i5/OS PASE は AIX のランタイムをベースにしているため、 i5/OS PASE プログラムでは、AIX でサポートされている、ロケール、文字ストリング処理、日時サービス、メッセージ・カタログ、および文字エンコード変換といった、数多くの一連のプログラミング・インターフェースを使用することができます。

関連情報

i5/OS PASE ロケール

i5/OS PASE の計画

i5/OS PASE での作業を開始する際には、このトピックでリストされている点を知っておくと役に立つでしょう。

i5/OS PASE は、i5/OS 上に、最小限の手間で iSeries サーバーに AIX アプリケーションを移植できるようにする、 AIX ランタイム環境を提供します。実際、多くの AIX プログラムは、変更を加えなくても i5/OS PASE で稼働します。これは、i5/OS PASE が、AIX 上で使用可能なものと同じ共用ライブラリーを数多く備えており、 pSeries® AIX PowerPC プロセッサーで稼働するのと同じように直接 iSeries PowerPC プロセッサーで稼働する、広範な AIX ユーティリティーのサブセットを備えているからです。

i5/OS PASE での作業を開始するにあたっては、次のいくつかの点に留意してください。

- **AIX バイナリーのターゲット・リリースと、バイナリーが稼働する i5/OS PASE のリリースとの間には、相互関係があります。**

i5/OS PASE アプリケーションを AIX 上でコンパイルする場合、 AIX で作成するアプリケーション・バイナリーには、そのアプリケーションが実行される i5/OS PASE のバージョンとの互換性が必要です。次の表は、i5/OS PASE の各バージョンと互換性がある AIX バイナリーのバージョンを示しています。たとえば、AIX リリース 5.1 用に作成される 32 ビット・アプリケーションは、i5/OS PASE V5R4、V5R3、または OS/400® PASE V5R2 では稼働しますが、OS/400 PASE V5R1 では稼働しません。同様に、AIX リリース 4.3 用に作成される 64 ビット・アプリケーションは、 OS/400 PASE V5R1 で稼働しますが、i5/OS PASE V5R4、V5R3、または OS/400 PASE V5R2 では稼働しません。

AIX のリリース	OS/400 V5R1	OS/400 V5R2	i5/OS V5R3	i5/OS V5R4
4.3 (32 ビット)	X	X	X	X
4.3 (64 ビット)	X	-	-	-
5.1 (32 または 64 ビット)	-	X	X	X
5.2 (32 または 64 ビット)	-	-	X	X
5.3 (32 または 64 ビット)	-	-	-	X

- **i5/OS PASE では、i5/OS 上に AIX カーネルがありません。**

  代わりに、共用ライブラリーが必要とする低レベル・システム関数は、すべて、i5/OS カーネルか統合 i5/OS 機能にルーティングされます。この点で、i5/OS PASE は、AIX プラットフォームと i5/OS プラットフォームとの間の架け橋となります。共用ライブラリー内の API では、AIX の場合と同じ構文が使用されますが、i5/OS PASE プログラムは i5/OS ジョブの中で実行され、他のすべての i5/OS ジョブとまったく同じように i5/OS によって管理されます。
- **ほとんどのケースにおいて、i5/OS PASE で呼び出される API は、 AIX 上とまったく同じように動作します。**

  ただし、一部の API は、i5/OS PASE では違う動作をするか、あるいは i5/OS PASE ではサポートされていません。このため、i5/OS PASE プログラムの準備の計画を立てる際は、 API 分析ツールを使用したコード全体の分析をまず行う必要があります。このツールを使用することにより、 AIX アプリケーションを i5/OS PASE に移植する際に考慮すべきプログラムの修正のタイプについて、総括的な概要を知ることができます。
- **AIX プラットフォームと i5/OS プラットフォームのいくつかの違いについても考慮してください。**
  - AIX には、一般に大/小文字の区別がありますが、特定の i5/OS ファイル・システムにはこの区別がありません。
  - AIX ではデータ・エンコードに ASCII を使用するのが一般的ですが、i5/OS では通常拡張 2 進化 10 進交換コードが使用されます。 i5/OS PASE プログラムからの ILE コードの呼び出しの詳細を管理したい場合には、これは考慮事項となります。たとえば、i5/OS PASE から任意の ILE プロシージャーへの呼び出しを行う場合には、ストリングに対して文字エンコード変換を処理するように、明示的に i5/OS PASE プログラムをコーディングする必要があります。i5/OS PASE のランタイム・サポートには、文字エンコード変換のための iconv_open()、iconv()、および iconv_close() 関数が含まれています。

    注: iconv() インターフェースのインプリメントは、i5/OS PASE と ILE で独立しており、それぞれ固有の変換テーブルがあります。 i5/OS PASE iconv() 関数でサポートされる変換は、統合ファイル・システムに保管されるバイト・ストリーム・ファイルとして保管されるため、ユーザーによって変更および拡張することができます。

– AIX アプリケーションは、行 (ファイルやシェル・スクリプト内の行など) の終わりが LF 改行になっていることを予期します。しかし、パーソナル・コンピューター (PC) ソフトウェアおよび i5/OS ソフトウェアでは、通常は行の最後に CRLF 改行が使用されます。
– AIX 上で使用される一部のスクリプトやプログラムは、標準のユーティリティーにハードコーディングされたパスを使用する場合があるため、 i5/OS PASE で使用するパスを反映して、パスに変更を加えることが必要になる場合があります。詳細については、『i5/OS PASE とのプログラムの互換性の分析』を参照してください。

これらの問題のいくつかは、i5/OS PASE によって自動的に処理されます。たとえば、通常はファイル記述子 (バイト・ストリーム・ファイルまたはソケット) に読み書きされるデータに対しては何の変換も実行されませんが、システムによって提供される i5/OS PASE ランタイム・サービス (i5/OS オプション 33 に同梱されている共用ライブラリーのシステム呼び出しやランタイム機能すべてを含む) を使用する場合、i5/OS PASE は、必要に応じて ASCII から EBCDIC への変換を実行します。

ILE 関数や API への呼び出しを行う i5/OS PASE プログラムの機能を拡張する場合には、 _ILECALL などの他の低レベル関数を使用できます。しかし、前述のとおり、データ変換を処理する必要がある場合があります。また、プログラムにこれらの拡張をコーディングするには、付加的なヘッダーおよびエクスポート・ファイルの使用が必要になります。

関連概念

『i5/OS PASE とのプログラムの互換性の分析』
C アプリケーションの iSeries サーバーへの移植性を評価する最初のステップには、アプリケーションで使用されるインターフェースを分析することが関係しています。

---

i5/OS PASE で実行するプログラムの準備

i5/OS で効果的に稼働する AIX プログラムを準備するためのステップは、プログラムの性質や、 i5/OS システム固有のインターフェースや機能を使用する必要が実際にあるかどうかによって変わります。

i5/OS PASE に アプリケーションを移植しようとする場合は、まずアプリケーションが AIX コンパイラーを使用してコンパイルできることを確認してください。場合によっては、この要件を満たすためにプログラムを修正する必要があります。

i5/OS PASE とのプログラムの互換性の分析

C アプリケーションの iSeries サーバーへの移植性を評価する最初のステップには、アプリケーションで使用されるインターフェースを分析することが関係しています。

この API 分析により、アプリケーション内で使用されるインターフェースが業界標準でないことや、i5/OS でサポートされていないことが明らかになります。また、インターフェースが業界標準に準拠しているとしても、 AIX または Linux マシンとは i5/OS のアーキテクチャーが異なるため、別の方法でサポートされていることも確認できます。

API 分析ツールは、フロントエンドおよびバックエンドのプロセスで構成されます。フロントエンド・プロセスでは、コンパイル済みのアプリケーションをスキャンして、アプリケーションで使用されるインターフェース (外部関数およびデータ) を抽出し、それらのすべてのインターフェースのリストを生成します。バックエンド・プロセスでは、このインターフェースのリストを入力として使用し、典型的なシステム API およびそれらのサポートから成るデータベースとインターフェースとを比較します。

API 分析ツールのフロントエンド・プロセスはシェル・スクリプトです。これは `nm` または `dump` コマンドを使用して、アプリケーションの外部シンボル・テーブルからシンボル情報を見つけます。

シンボルからストリップされたバイナリーには、分析するツールに関する、動的バインディング情報が十分に含まれていることがあります。静的にバインドされたバイナリーでは、ライブラリー・インターフェースを分析の対象に含めませんが、システム呼び出しの依存関係は分析用に引き続き公開します。

コンパイル前に実行する追加の分析

API 分析ツールから収集する情報に加えて、以下の情報も収集する必要があります。

- アプリケーションで使用されるライブラリーのリストの取得

  分析ツールでは、アプリケーションで使用される標準 API の一部についてフィードバックを提供しますが、多数の共通 API セットを探すわけではありません。ライブラリー分析は、アプリケーションで使用されるミドルウェア API のいくつかを識別するのに役立ちます。ご使用のコマンドおよび共用オブジェクトのそれぞれに対して以下のコマンドを実行して、アプリケーションで必要なライブラリーのリストを入手することができます。

  ```
  dump -H binary_name
  ```

- ハードコーディングされたパス名の検査

  クリデンシャルを変更するプログラムを実行する場合、または i5/OS PASE 環境変数が PASE_EXEC_QOPENSYS=N であるときでもコマンドまたはスクリプトを実行させる場合は、ハードコーディングされたパス名を変更しなければならないことがあります。

  /usr/bin/ksh は絶対パス (ルートで始まる) であるため、それが見つからない場合、またはそれがバイト・ストリーム・ファイルでない場合は、 i5/OS PASE は /QOpenSys ファイル・システムを検索して、パス名 /QOpenSys/usr/bin/ksh を探します。 QShell ユーティリティー・プログラムはバイト・ストリーム・ファイルではないため、オリジナルの (絶対) パスが QShell ユーティリティー・プログラムに対するシンボリック・リンク (/usr/bin/sh など) である場合でも i5/OS PASE は /QOpenSys ファイル・システムを探します。

  関連概念

  6 ページの『i5/OS PASE の計画』
  i5/OS PASE での作業を開始する際には、このトピックでリストされている点を知っておくと役に立つでしょう。

  関連情報

  API 分析ツール

AIX ソースのコンパイル

i5/OS PASE でのインストールをサポートする AIX コンパイラー製品のうちの一つをインストールして i5/OS PASE 環境でプログラムをコンパイルすることができます。

プログラムが AIX インターフェースのみを使用する場合、必須の AIX ヘッダーとコンパイルし、AIX ライブラリーとリンクして、 i5/OS PASE 用のバイナリーを作成します。 i5/OS PASE は、AIX システムが提供する共用ライブラリーと静的に結合するアプリケーションはサポートしません。

i5/OS PASE プログラムの構造と、PowerPC 用の AIX プログラムの構造は同一です。

i5/OS PASE (オペレーティング・システムのオプション 33) にはコンパイラーが含まれません。 AIX システムを使用して i5/OS PASE プログラムをコンパイルするか、または i5/OS PASE でのインストールをサポートする AIX コンパイラー製品のうちの一つをオプションでインストールしてi5/OS PASE 環境でプログラムをコンパイルすることができます。

pSeries サーバーでの AIX コンパイラーの使用

PowerPC 用の AIX ABI と互換性がある出力を生成する AIX コンパイラーおよびリンカーを使用して、i5/OS PASE プログラムを作成することができます。i5/OS PASE は、PowerPC には存在しない POWER™ アーキテクチャー指示 (キャッシュ管理の IBM POWER 指示は存在する) を使用するバイナリーの指示エミュレーション・サポートを提供します。

i5/OS PASE での AIX コンパイラーの使用

i5/OS PASE は、i5/OS PASE 環境での以下の別々に使用可能な AIX コンパイラーのインストールをサポートします。

- IBM XL C/C++ Enterprise Edition for AIX バージョン 7.0 (5724-I11)
- IBM XL C Enterprise Edition for AIX バージョン 7.0 (5724-I10)
- IBM XL Fortran Enterprise Edition for AIX バージョン 9.1 (5724-I08)

これらの製品を使用して、 iSeries サーバー上の i5/OS PASE 環境内で AIX アプリケーションの開発、コンパイル、ビルドおよびを実行をすべて行うことができます。

開発ツール

i5/OS PASE には、AIX で使用する多くの開発ツール (例: `ld`、`ar`、`make`、`yacc`) が付属しています。i5/OS PASE で使用できる、他のソースからの AIX ツール (たとえば、オープン・ソース・ツール `gcc` など) も多数あります。

また、iSeries Tools for Developers PRPQ (5799-PTL) にも、 iSeries アプリケーションを開発、構築、移植する上で役立つ多彩なツールが含まれています。この PRPQ についての詳細は、Web サイト IBM Virtual Innovation Center for Hardware (英語) を参照してください。

ポインターの処理に関するコンパイラーの注意事項

- `xlc` コンパイラーは、 `-qlngdbl128` と `-qalign=natural` を組み合わせて使用することにより、 (長倍精度実数型の) 16 バイト調整の限定サポートを提供します。 *ILEpointer* 型は、マシン・インターフェース (MI) ポインターが構造内で 16 バイトに調整されるようにするために、これらのコンパイラー・オプションを必要とします。オプション `-qldbl128` を使用すると、長倍精度実数型は強制的に 128 ビット型になります。この場合、長倍精度実数フィールドの `printf` のような操作を処理するために libc128.a を使用する必要があります。

  `xlc` コマンドの代わりに `xlc128` コマンドを使用すると、簡単にオプション `-qlngdbl128` を入手し、libc128.a とリンクすることができます。
- `xlc/xlC` コンパイラーには現在、静的変数または自動変数の 16 バイト調整を強制する手段がありません。このコンパイラーは、構造内の 128 ビット長倍精度実数フィールドの相対調整を保証するだけです。 `malloc` の i5/OS PASE バージョンは常に 16 バイト調整ストレージを提供するので、スタック・ストレージの 16 バイト調整を行うことができます。
- また、ヘッダー・ファイル as400_types.h も、 64 ビット整数である `long long` 型に依存します。 `xlc` コンパイラー・オプション `-qlonglong` はこの形状を保証します (これは、`xlc` コンパイラーを実行するすべてのコマンドでデフォルトであるわけではありません)。

例

以下の例は、AIX システムで i5/OS PASE プログラムをコンパイルする際に使用するためのものです。i5/OS PASE にインストールされたコンパイラーを使用してプログラムのコンパイルを行っているのであれ

ば、i5/OS システム固有のヘッダー・ファイルまたは i5/OS システム固有のエクスポートの場所についてのコンパイラー・オプションを指定する必要はありません。これらのファイルは i5/OS システム上のデフォルト・パス位置 /usr/include/ および /usr/lib/ にあるからです。

例 1

以下の AIX システム上のコマンドを使用すると、testpgm という i5/OS PASE プログラムが作成されます。これは、libc.a によってエクスポートされる i5/OS システム固有のインターフェースを使用できます。

```
xlc -o testpgm -qldbl128 -qlonglong -qalign=natural
         -bI:/mydir/as400_libc.exp testpgm.c
```

この例では、i5/OS システム固有のヘッダー・ファイルが AIX ディレクトリー /usr/include にコピーされ、 i5/OS システム固有のエクスポート・ファイルが AIX ディレクトリー /mydir にコピーされることを前提としています。

例 2

以下の例では、i5/OS システム固有のヘッダー・ファイルおよびエクスポート・ファイルが /pase/lib にあることを前提としています。

```
xlc -o as400_test -qldbl128 -qlonglong -qalign=natural -H16
          -l c128
          -I /pase/lib
          -bI:/pase/lib/as400_libc.exp as400_test.c
```

例 3

以下の例では、同じオプションを使用して、例 2 と同じプログラムを作成しています。ただし、xlc_r コマンドがマルチスレッド・プログラムで使用され、コンパイル済みアプリケーションがスレッド・セーフのランタイム・ライブラリーとリンクするようになっています。

```
xlc_r -o as400_test -qldbl128 -qlonglong -qalign=natural -H16
            -l c128
            -I /pase/lib
            -bI:/pase/lib/as400_libc.exp as400_test.c
```

この例では、IBM DB2 Universal Database (UDB) for iSeries コール・レベル・インターフェース (CLI) 用の i5/OS PASE サポートを使用する場合、build コマンドで -bI:/pase/include/libdb400.exp も指定する必要があります。

-bI ディレクティブは、パラメーターを ld コマンドに渡すようにコンパイラーに命令します。このディレクティブは、ライブラリーからエクスポートした記号を含むエクスポート・ファイルが、アプリケーションによってインポートされるように指定します。

関連概念

2 ページの『i5/OS PASE の概要』
i5/OS PASE は、i5/OS 上で実行している AIX アプリケーション用の統合ランタイム環境です。

関連情報

i5/OS PASE シェルおよびユーティリティー

i5/OS PASE での AIX コンパイラーのインストール

このトピックのステップに従って、i5/OS PASE に AIX コンパイラーをインストールすることができます。

以下の別個に入手可能な AIX コンパイラーのいずれかを、 i5/OS PASE 環境にインストールすることができます。

- IBM XL C/C++ Enterprise Edition for AIX バージョン 7.0 (5724-I11)
- IBM XL C Enterprise Edition for AIX バージョン 7.0 (5724-I10)
- IBM XL Fortran Enterprise Edition for AIX バージョン 9.1 (5724-I08)

これらの製品を使用して、iSeries サーバー上の i5/OS PASE 環境内で AIX アプリケーションの開発、コンパイル、ビルドおよび実行を行うことができます。

関連概念

1 ページの『V5R4 の新機能』
このページでは、このリリースで i5/OS PASE に加えられた変更点を取り上げています。

関連情報

XL C/C++ Enterprise Edition for AIX

XL C Enterprise Edition for AIX

AIX コンパイラーのインストール:

i5/OS PASE は、 AIX システム上にアプリケーションをインストールするために標準的に使用される、AIX `smit` や `installp` ユーティリティーをサポートしていません。 XL C/C++ Enterprise Edition for AIX バージョン 7.0 または XL C Enterprise Edition for AIX 製品のインストールは、各コンパイラーのインストール・メディアにある「非デフォルト・インストール」スクリプトを使用して行います。

iSeries i5/OS PASE に XL C/C++ Enterprise Edition for AIX バージョン 7.0 または XL C Enterprise Edition for AIX などの製品をインストールするには、以下のステップを実行します。

1. 必要な前提条件を満たしていることを確認します。 コンパイラーを正常にインストールして使用するには、コンパイラー・インストール・メディアに加えて、以下のものがご使用の iSeries サーバーにインストールされている必要があります。
   - 5722SS1 オプション 33 - i5/OS PASE 自体
   - 5722SS1 オプション 13 - システム・オープンネス・インクルード。/usr/include 統合ファイル・システム・ディレクトリーにあるコンパイラー・ヘッダー・ファイルを含む。
   - Perl。コンパイラー・インストール・スクリプトには Perl が必要です。 Perl のインストールには、以下の 2 つの方法があります。
     - 5799PTL - iSeries Tools for Developers PRPQ。 Perl は (他の多くの有用な開発ツールと共に)、別個に入手可能な iSeries Tools For Developers PRPQ に入っています。
     - http://www.cpan.org/ports/#os400 - i5/OS PASE 用の Perl Port バイナリー配布。
2. コンパイラー製品インストール CD を iSeries CD-ROM 装置に挿入します。
3. i5/OS に *ALLOBJ 権限のあるユーザー・プロファイルでサインオンします。 コンパイラー製品ファイルは、このユーザー・プロファイルに所有されます。
4. CL コマンド `call qp2term` を入力して、対話式 i5/OS PASE 端末セッションを開始します。
5. 以下のコマンドを入力して、適当なコンパイラー・インストール・スクリプトを復元します。

コンパイラー	コマンド
XL C/C++ Enterprise Edition for AIX 用	`cd /` `restore -qf /QOPT/CDROM/USR/SYS/INST.IMA/VACPP.NDI ./usr/vacpp/bin/vacppndi`

コンパイラー	コマンド
XL C Enterprise Edition for AIX 用	cd / restore -qf /QOPT/CDROM/USR/SYS/INST.IMA/VAC.NDI ./usr/vac/bin/vacndi
XL Fortran Enterprise Edition for AIX 用	cd / restore -qf /QOPT/CDROM/USR/SYS/INST.IMA/XLF.NDI ./usr/lpp/xlf/bin/xlfndi

6. インストール・スクリプトを実行して、コンパイラーをインストールします。 コンパイラーの宛先ディレクトリーは、コマンドの -b オプションで指定します。コンパイラーのための推奨されるディレクトリー名が以下の表にあるコマンドで使用されています。別のディレクトリーを選択する場合、(大/小文字の区別があるファイル名を使えるように) そのディレクトリーは /QOpenSys ツリーになければならないことに注意してください。

コンパイラー	コマンド
XL C/C++ Enterprise Edition for AIX 用	/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/vacpp/bin/vacppndi -i -d /QOPT/CDROM/USR/SYS/INST.IMA -b /QOpenSys/xlc70
XL C Enterprise Edition for AIX 用	/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/vac/bin/vacndi -i -d /QOPT/CDROM/USR/SYS/INST.IMA -b /QOpenSys/xlc70
XL Fortran Enterprise Edition for AIX 用	/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/lpp/xlf/bin/xlfndi -i -d /QOPT/CDROM/USR/SYS/INST.IMA -b /QOpenSys/xlf91
注: コマンドは 1 つの長いコマンドとして入力してください。	

これでコンパイラーはインストールされ、i5/OS PASE で使用できるようになりました。

xlC などの XL C/C++ Enterprise Edition for AIX コンパイラー・コマンドは、ディレクトリー /QOpenSys/xlc70/usr/vacpp/bin/ にあります。このディレクトリーは $PATH 環境変数に加えることができます。

XL C/C++ Enterprise Edition for AIX コンパイラー文書は、ディレクトリー /QOpenSys/xlc70/usr/vacpp/pdf/en_US/ に、Adobe Acrobat 形式で存在しています。

xlc や cc などの XL C Enterprise Edition for AIX コンパイラー・コマンドは、ディレクトリー /QOpenSys/xlc70/usr/vac/bin/ にあります。このディレクトリーは $PATH 環境変数に加えることができます。

XL C Enterprise Edition for AIX コンパイラー文書は、ディレクトリー /QOpenSys/xlc70/usr/vac/pdf/en_US/ に、Adobe Acrobat 形式で存在しています。

xlf などの XL Fortran for AIX コンパイラー・コマンドは、ディレクトリー /QOpenSys/xlf91/usr/bin/ にあります。このディレクトリーは $PATH 環境変数に加えることができます。

XL Fortran for AIX コンパイラー文書は、ディレクトリー /QOpenSys/xlf91/usr/share/man/info/en_US/xlf/pdf/ に、Adobe Acrobat 形式で存在しています。

PTF 更新手順:

XL C/C++ Enterprise Edition for AIX バージョン 7.0 または XL C Enterprise Edition for AIX 製品のためのプログラム一時修正 (PTF) のインストールは、最初のコンパイラー・インストールで使用したものと同じ「非デフォルト・インストール」スクリプトを使用して実行します。

PTF をインストールする前に、このトピック内の前述のステップを使用してコンパイラーのインストールを実行しておく必要があります。

iSeries i5/OS PASE に XL C/C++ Enterprise Edition for AIX バージョン 7.0 または XL C Enterprise Edition for AIX などの製品用の PTF をインストールするには、以下のステップを実行します。

1. インストールする PTF パッケージ・ファイルを入手します。 コンパイラーの PTF パッケージの圧縮 TAR イメージは、 XL C/C++ Enterprise Edition Web サイトのサポート・ダウンロード・セクションからダウンロードできます。
2. PTF パッケージ・ファイルを解凍し、次いで untar します。 圧縮 TAR イメージを /QOpenSys/vacptf/ ディレクトリーにダウンロードしたら、 QP2TERM コマンド行から以下のコマンドを使用してこれを実行できます。

   ```
   cd /QOpenSys/ptf
   uncompress <filename.tar.Z>
   tar -xvf <filename.tar>
   ```

3. インストールする PTF パッケージのリストを含むファイルを作成します。 QP2TERM コマンド行から以下のコマンドを使用してこれを実行できます。

   ```
   cd /QOpenSys/ptf
   ls *.bff > ptflist.txt
   ```

4. インストール・スクリプトを実行して、PTF をインストールします。更新するコンパイラーに応じて、以下のコマンドの 1 つを QP2TERM コマンド行から入力します。

コンパイラー	コマンド
XL C/C++ Enterprise Edition for AIX 用	/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/vacpp/bin/vacppndi -d /QOpenSys/ptf -b /QOpenSys/xlc70 -u /QOpenSys/ptf/ptflist.txt
XL C Enterprise Edition for AIX 用	/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/vac/bin/vacndi -d /QOpenSys/ptf -b /QOpenSys/xlc70 -u /QOpenSys/ptf/ptflist.txt
XL Fortran Enterprise Edition for AIX 用	/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/lpp/xlf/bin/xlfndi -d /QOpenSys/ptf -b /QOpenSys/xlf91 -u /QOpenSys/ptf/ptflist.txt
注: コマンドは 1 つの長いコマンドとして入力してください。	

   インストール・スクリプトは、PTF 更新前に存在していたコンパイラー・ファイルの圧縮 TAR バックアップを作成します。この説明で示されるディレクトリーを使用する場合、このファイル名は /QOpenSys/xlc70.backup.tar.Z または /QOpenSys/xlf91.backup.tar.Z となります。 PTF 更新のインストールまたは PTF 更新そのもので問題が発生した場合、このバックアップから復元して、PTF 更新をアンインストールすることができます。

関連情報

XL C/C++ Enterprise Edition for AIX

iSeries サーバーへの i5/OS PASE プログラムのコピー

i5/OS PASE で実行する AIX バイナリーを統合ファイル・システムにコピーします。

統合ファイル・システムで使用できるファイル・システムはすべて、 i5/OS PASE 内で使用することができます。

オペレーティング・システム間でファイルを移動するときは、アプリケーションでの大/小文字の区別、および AIX が使用する改行文字と i5/OS が使用する改行文字の違いを理解しておく必要があります。これらの違いによって問題が発生する場合があります。

ファイル転送プロトコル (FTP)、Server Message Block (SMB)、またはリモート・ファイル・システムを使用することにより、i5/OS PASE と iSeries サーバーの間でプログラムや関連ファイルの転送を行うことができます。

関連資料

18 ページの『ヘッダー・ファイルのコピー』
このトピックの説明に従って、iSeries サーバーから AIX マシンにヘッダー・ファイルをコピーすることができます。

20 ページの『エクスポート・ファイルのコピー』
このトピックの説明に従って、iSeries サーバーから AIX ディレクトリーにエクスポート・ファイルをコピーすることができます。

関連情報

統合ファイル・システム

大/小文字の区別

アプリケーションが大/小文字の区別を行う場合、/QOpenSys ファイル・システム、または大/小文字を区別するものとして作成したユーザー定義のファイル・システムにファイルを移動します。

AIX および Linux などのオペレーティング・システムのインターフェースは一般に、大/小文字を区別します。 i5/OS では、大/小文字の区別がない場合もあります。特に、大/小文字の区別があることによって、既存のコードとの混乱が生じる可能性があるいくつかの状況を把握しておく必要があります。

ディレクトリーまたはファイル単位の大/小文字の区別は、 i5/OS で使用しているファイル・システムに依存します。 /QOpenSys ファイル・システムでは大/小文字の区別があり、大/小文字の区別があるユーザー定義のファイル・システム (UDFS) を作成することができます。

例

以下は、大/小文字の区別から生じる可能性のある問題の例です。

例 1

この例では、シェルが readdir() による戻り値に対して総称名接頭部の文字比較を行います。ただし、QSYS.LIB ファイル・システムは大文字のディレクトリー項目を戻すため、どの項目も小文字の総称名接頭部とは一致しません。

```
$ ls -d /qsys.lib/v4r5m0.lib/qwobj*
/qsys.lib/v4r5m0.lib/qwobj* not found
$ ls -d /qsys.lib/v4r5m0.lib/QWOBJ*
/qsys.lib/v4r5m0.lib/QWOBJ.FILE
```

例 2

この例は最初の例と類似していますが、シェルではなく find ユーティリティーによって比較が行われている点が異なります。

```
$ find /qsys.lib/v4r5m0.lib/ -name 'qwobj*' -print

$ find /qsys.lib/v4r5m0.lib/ -name 'QWOBJ*' -print
/qsys.lib/v4r5m0.lib/QWOBJ.FILE
```

例 3

ps ユーティリティーはユーザー名に大/小文字の区別があることを予期しているため、 -u オプションによって指定された大文字の名前と i5/OS PASE ランタイム機能 getpwuid() によって戻される小文字の名前との一致は認識しません。

```
$ ps -uTIMMS -f
UID PID PPID C STIME TTY TIME CMD
$ ps -utimms -f
UID PID PPID C STIME TTY TIME CMD
timms 617 570 0 10:54:00 - 0:00 /QOpenSys/usr/bin/-sh -i
timms 660 617 0 11:14:56 - 0:00 ps -utimms -f
```

関連情報

ファイル・システムの比較

統合ファイル・システムにおける改行文字

AIX と i5/OS では、テキスト・ファイル内 (たとえばファイルおよびシェル・スクリプト内) で使用する改行文字が異なります。

i5/OS PASE プログラムのソースの AIX アプリケーションでは、行 (ファイルやシェル・スクリプト内の行など) の終わりに LF 改行を入れることが求められています。ただし、PC ソフトウェアや一般的な i5/OS ソフトウェアでは、行の最後が CRLF 改行になっている場合も少なくありません。

```
awk '{ gsub( /¥r$/, "" ); print $0 }' < oldfile > newfile
```

FTP での CRLF の使用

この違いが問題の原因となる 1 つの例は、ファイル転送プロトコル (FTP) を使ってソース・ファイルとシェル・スクリプトを AIX から iSeries に転送するケースです。 FTP の標準では、テキスト・モードで送信され、行の最後に CRLF 改行が使用されたデータが必要です。一方 AIX では、テキスト・モードのインバウンド・ファイルを処理する際に、 FTP ユーティリティーによって CR 改行が除去されます。i5/OS FTP は、必ずデータ・ストリームに示されているとおりに書き込みを行い、必ずテキスト・モード用に CRLF を残すため、これが i5/OS PASE のランタイムやユーティリティーでは問題の原因になります。

この問題を防ぐため、可能な場合は、AIX オペレーティング・システムからの転送にはバイナリー・モードを使用してください。また、パーソナル・コンピューターから転送されるテキスト・ファイルでも、ほとんどの場合に CRLF 区切り文字が使用されています。そのような場合は、ファイルをまず AIX に転送するようにすれば、問題を修正できます。現行ディレクトリーのファイルから CR を除去するための手段として、以下のコマンドを使用することができます。

```
awk '{ gsub( /¥ r$/, "" ); print $0 }' < oldfile > newfile
```

iSeries や PC のエディターでの CRLF の使用

iSeries サーバーやワークステーションのエディター (Windows® のメモ帳エディターなど) でファイルやシェル・スクリプトを編集する場合にも問題は生じます。これらのエディターで使用される改行区切り文字は CRLF であって、i5/OS PASE で使用される LF ではないからです。

改行区切り文字として CRLF を用いないエディターは数多くあります (例: ez エディター)。

関連情報

IBM Virtual Innovation Center for Hardware

ファイルの転送

このトピックで説明されている 3 つの方法のうちのいずれかの方法によって、i5/OS PASE と iSeries サーバーの間でプログラムや関連ファイルの転送を行うことができます。

- ファイル転送プロトコルを使用したプログラムのコピー
- Server Message Block を使用したプログラムのコピー
- リモート・ファイル・システムを使用したプログラムのコピー

ファイル転送プロトコルを使用したプログラムのコピー

i5/OS ファイル転送プロトコル (FTP) デーモンおよびクライアントを使用することによって、i5/OS 統合ファイル・システムの間でファイルの転送を行うことができます。ファイルの転送を行う際はバイナリー・モードを使用します。 FTP サブコマンド `binary` を使用してこのモードを設定してください。

ファイルを統合ファイル・システムに配置する際には、命名形式 1 (i5/OS FTP コマンドの `NAMEFMT 1` サブコマンド) を使用する必要があります。この形式により、パス名を使用することができ、ストリーム・ファイルにファイルを転送します。命名形式 1 を使用するには、以下のいずれかを行います。

- パス名を使ってディレクトリーを変更します。

  これで、セッションが自動的に命名形式 1 になります。この方法を使用すると、最初のディレクトリーの前にスラッシュ (/) が付けられます。たとえば、次のようになります。

  ```
  cd /QOpenSys/usr/bin
  ```

- リモート・クライアントの場合は FTP サブコマンド `quote site namefmt 1` を使用し、ローカル・クライアントの場合は `namefmt 1` を使用します。

Server Message Block を使用したプログラムのコピー

i5/OS は、Server Message Block (SMB) クライアント・コンポーネントおよびサーバー・コンポーネントをサポートします。 NetServer™ を構成して実行すると、 i5/OS PASE は /QNTC ファイル・システムを使用してネットワーク内の SMB サーバーにアクセスできます。 AIX プラットフォームまたは Linux プラットフォームでこれと同じサービスを提供するには、SAMBA サーバーが必要となります。構成済みで操作可能なシステム (AIX など) をインストールすると、 i5/OS PASE で使用可能なディレクトリーおよびファイルを作成することができます。

リモート・ファイル・システムを使用したプログラムのコピー

i5/OS では、統合ファイル・システムのファイル・スペース内のマウント・ポイントに、ネットワーク・ファイル・システム (NFS) をマウントすることができます。 AIX では、分散ファイル・システム (DFS™) と Andrew File System (AFS®) に加えて NFS もサポートしており (DFS から NFS に、また AFS から NFS に変換するプログラムを使用)、 i5/OS はこれらのファイル・システムのエクスポートおよびマウントが行えます。これにより、i5/OS PASE アプリケーションもこれらのファイル・システムを使用することができます。 i5/OS ユーザー・プロファイルのユーザー ID 番号およびグループ ID 番号を使用して、アクセス対象のディレクトリー・パスおよびファイルに対するセキュリティー権限が検証されます。複数のプラットフォームで同一のユーザーとなるように意図されたユーザー・プロファイルには、すべてのシステム上で同じユーザー ID が含まれるようにします。

i5/OS は NFS サーバーとして使用されるときに最もその性能を発揮します。そのように使用する場合、AIX システムから i5/OS 統合ファイル・システムのディレクトリーにマウントする必要があり、AIX はプログラムを作成する際に i5/OS に直接書き込みます。

注: i5/OS NFS は現在、マルチスレッド・アプリケーションではサポートされていません。

関連情報

FTP

i5/OS 機能を使用するための i5/OS PASE プログラムのカスタマイズ

AIX アプリケーションで、システム提供の i5/OS PASE 共用ライブラリーでは直接サポートされていない i5/OS の機能を利用したい場合は、いくらかの付加的なステップを実行してアプリケーションを準備する必要があります。

準備を行うには、以下のステップを完了してください。

1. i5/OS システム固有の機能へのアクセスを調整する、すべての必要な i5/OS PASE ランタイム機能が呼び出されるように、 AIX アプリケーションをコーディングします。
2. AIX システム上の i5/OS PASE プログラムをコンパイルする場合は、カスタマイズしたアプリケーションをコンパイルする前に、以下のステップを実行する必要があります。
   a. AIX システムに i5/OS システム固有の必須ヘッダー・ファイルをコピーします。
   b. AIX システムに i5/OS システム固有の必須エクスポート・ファイルをコピーします。

関連概念

32 ページの『i5/OS PASE プログラムからの i5/OS プログラムおよびプロシージャーの呼び出し』
i5/OS PASE では、ILE プロシージャー、Java プログラム、OPM プログラム、i5/OS API、および i5/OS 機能への統合アクセスを持つ CL コマンドを呼び出すためのメソッドを提供します。

43 ページの『i5/OS PASE プログラムと i5/OS の相互作用』
i5/OS の機能を使用するように i5/OS PASE プログラムをカスタマイズする場合は、プログラムがそれらの機能とどのように相互作用するかを考慮する必要があります。

関連情報

i5/OS PASE で使用するためのランタイム機能

ヘッダー・ファイルのコピー

このトピックの説明に従って、iSeries サーバーから AIX マシンにヘッダー・ファイルをコピーすることができます。

i5/OS PASE は、標準 AIX ランタイムに、i5/OS システム固有のサポート用のヘッダー・ファイルを追加します。これらのヘッダー・ファイルは、i5/OS PASE および i5/OS オペレーティング・システムによって提供されます。

これらのヘッダー・ファイルを、 iSeries サーバーから AIX マシンのヘッダー・ファイル検索パスにコピーしてください。

ヘッダー・ファイルは、AIX ディレクトリー /usr/include、またはコンパイラーのヘッダー・ファイル検索パスにある他の任意のディレクトリーにコピーできます。

/usr/include 以外のディレクトリーを使用する場合は、 AIX コンパイラー・コマンドの -I オプションを使用して、そのディレクトリーをヘッダー・ファイル検索パスに追加できます。

i5/OS PASE ヘッダー・ファイルのコピー

i5/OS PASE ヘッダー・ファイルは、以下の i5/OS ディレクトリーにあります。

/QOpenSys/QIBM/ProdData/OS400/PASE/include

i5/OS PASE では、次のようなヘッダー・ファイルが提供されています。

ヘッダー・ファイル	説明
as400_protos.h	このヘッダー・ファイルは、i5/OS PASE システム固有の各種機能を ILE に提供します。
as400_types.h	このヘッダー・ファイルは、ILE への呼び出し用の固有の i5/OS パラメーター・タイプを宣言します。 このヘッダー・ファイルは、16 バイト・マシン・インターフェース (MI) ポインターに、タイプ `ILEpointer` を宣言します。この宣言は、長倍精度実数型が 128 ビット・フィールドであることに依存します。 as400_types.h で宣言される他のタイプは、long long 型が 64 ビット整数であることに依存します。 as400_types.h で宣言されるタイプのサイズや位置合わせが適正なものとなるようにするため、 AIX コンパイラーは、オプション `-qlngdbl128`、`-qalign=natural`、および `-qlonglong` を指定して実行する必要があります。
os400msg.h	このヘッダー・ファイルは、i5/OS メッセージを送受信するための関数を宣言します。

i5/OS ヘッダー・ファイルのコピー

i5/OS PASE アプリケーションで他の i5/OS 機能にアクセスする計画であれば、使用する i5/OS 機能のためのヘッダー・ファイルを、開発マシンにコピーしておくと便利な場合があります。一般的には、i5/OS プログラムまたはプロシージャーは、 i5/OS PASE アプリケーションから直接実行することはできないことに注意してください。詳しくは、『i5/OS PASE プログラムからの i5/OS プログラムおよびプロシージャーの呼び出し』を参照してください。

i5/OS システムで提供されるヘッダー・ファイルは、/QIBM/include ディレクトリーにあります。

アプリケーションで何らかの i5/OS API ヘッダー・ファイルが必要とされる場合は、まずヘッダー・ファイルを EBCDIC から ASCII に変換し、その変換したファイルを AIX ディレクトリーにコピーする必要があります。

EBCDIC のテキスト・ファイルを ASCII に変換する 1 つの方法として、 i5/OS PASE の `Rfile` ユーティリティーを使用できます。

次に示す例では、i5/OS PASE の `Rfile` ユーティリティーで i5/OS ヘッダー・ファイル /QIBM/include/qusec.h を読み取り、そのデータを i5/OS PASE コード化文字セット ID (CCSID) に変換した後、各行から行末のブランクを除去して、その結果として生成されたものをバイト・ストリーム・ファイル `ascii_qusec.h` に書き込みます。

```
Rfile -r /QIBM/include/qusec.h > ascii_qusec.h
```

関連概念

44 ページの『データベース』
i5/OS PASE は DB2® UDB for iSeries コール・レベル・インターフェース (CLI) をサポートしています。 AIX および i5/OS 上の DB2 CLI は互いに適切なサブセットではないので、いくつかのインターフェースで多少の違いがあり、あるインプリメンテーションで存在する API が、別のインプリメンテーションでは存在しない場合もあります。

32 ページの『i5/OS PASE プログラムからの i5/OS プログラムおよびプロシージャーの呼び出し』
i5/OS PASE では、ILE プロシージャー、Java プログラム、OPM プログラム、i5/OS API、および i5/OS 機能への統合アクセスを持つ CL コマンドを呼び出すためのメソッドを提供します。

関連タスク

32 ページの『ILE プロシージャーの呼び出し』
このトピックの説明に従って、ILE プロシージャーを準備し、i5/OS PASE プログラムから呼び出すことができます。

関連資料

14 ページの『iSeries サーバーへの i5/OS PASE プログラムのコピー』
i5/OS PASE で実行する AIX バイナリーを統合ファイル・システムにコピーします。

エクスポート・ファイルのコピー

このトピックの説明に従って、iSeries サーバーから AIX ディレクトリーにエクスポート・ファイルをコピーすることができます。

i5/OS システム固有の機能にアクセスする必要のあるアプリケーションを作成するには、以下の i5/OS ディレクトリーにあるエクスポート・ファイルを使用することをお勧めします。

/QOpenSys/QIBM/ProdData/OS400/PASE/lib

これらのファイルを任意の AIX ディレクトリーにコピーできます。 AIX システム上の共用ライブラリーにない記号を定義するには、 AIX `ld` コマンド (または compiler コマンド) で `-bI:` オプションを使用します。

i5/OS PASE は以下のエクスポート・ファイルを提供します。

エクスポート・ファイル	機能
as400_libc.exp	このファイルは、libc.a にある、i5/OS システム固有の機能用のエクスポート・ファイルです。 as400_libc.exp ファイルは、 libc.a の i5/OS PASE バージョンからのすべてのエクスポート (そのライブラリーの AIX バージョンによってエクスポートされないもの) を定義します。
libdb400.exp	このファイルは、i5/OS データベース機能用のエクスポート・ファイルです。 libdb400.exp ファイルは、 (DB2 UDB for iSeries コール・レベル・インターフェース (CLI) がサポートする) i5/OS PASE libdb400.a ライブラリーからのエクスポートを定義します。

関連概念

44 ページの『データベース』
i5/OS PASE は DB2 UDB for iSeries コール・レベル・インターフェース (CLI) をサポートしています。 AIX および i5/OS 上の DB2 CLI は互いに適切なサブセットではないので、いくつかのインターフェースで多少の違いがあり、あるインプリメンテーションで存在する API が、別のインプリメンテーションでは存在しない場合もあります。

関連資料

14 ページの『iSeries サーバーへの i5/OS PASE プログラムのコピー』
i5/OS PASE で実行する AIX バイナリーを統合ファイル・システムにコピーします。

i5/OS 機能にアクセスするための i5/OS PASE API

i5/OS PASE は、 ILE コードおよびその他の i5/OS 関数にアクセスするためのいくつかの API を提供しています。どの API を使用するかは、コンパイラーをどの程度機能させるかではなく、どれだけの準備と構成を行うかに依存します。

関連情報

i5/OS PASE API

i5/OS 環境での i5/OS PASE プログラムの使用

i5/OS PASE プログラムは他の i5/OS プログラムを呼び出すことができ、他の i5/OS プログラムは i5/OS PASE プログラムのプロシージャーを呼び出すことができます。

i5/OS PASE プログラムおよびプロシージャーの実行

このトピックでは、ジョブでの i5/OS PASE プログラムの開始、および ILE プログラムからの i5/OS PASE プロシージャーの呼び出しに関する情報と例が提供されています。

i5/OS PASE プログラムは、以下のいくつかの方法で実行できます。

- i5/OS ジョブ内で
- i5/OS PASE 対話式シェル環境から
- ILE プロシージャーからの呼び出し先プログラムとして

注: i5/OS PASE プログラムを i5/OS 上で実行する場合、 i5/OS PASE 環境変数は ILE 環境変数に依存していないことを覚えておく必要があります。一方の環境で変数を設定しても、他方の環境には影響を与えません。

i5/OS PASE プログラムでの作業を可能にする ILE プロシージャー

i5/OS PASE には、ILE コードが i5/OS PASE サービスに (i5/OS PASE プログラム内に特別なプログラミングすることなく) アクセスできるようにするための、いくつかの ILE プロシージャー API が用意されています。

- Qp2dlclose
- Qp2dlerror
- Qp2dlopen
- Qp2dlsym
- Qp2errnop
- Qp2free
- Qp2jobCCSID
- Qp2malloc
- Qp2paseCCSID
- Qp2ptrsize

ILE スレッドへの接続

i5/OS PASE で作成されていないスレッド (例: Java スレッドや ILE pthread_create で作成されたスレッド) で実行される ILE コードから、 i5/OS PASE プログラム内のプロシージャーを呼び出すことが可能です。Qp2CallPase は、自動的に ILE スレッドを i5/OS PASE に (対応する i5/OS PASE pthread 構造体を作成することによって) 接続させます。ただしこれは、i5/OS PASE プログラムの開始時に i5/OS PASE 環境変数 PASE_THREAD_ATTACH が Y に設定された場合のみです。

i5/OS PASE から i5/OS プログラムに結果を戻す

i5/OS _RETURN() 関数を使用すると、i5/OS PASE プログラムを呼び出し、 i5/OS PASE 環境を終了させることなく結果を戻すことが可能です。これによって、i5/OS PASE プログラムを開始し、QP2SHELL2 (QP2SHELL ではない) または Qp2RunPase API が戻された後、そのプログラムの中でプロシージャーを (Qp2CallPase を使用して) 呼び出すことが可能になります。

関連概念

31 ページの『環境変数の処理』
i5/OS PASE 環境変数は ILE 環境変数に依存しません。一方の環境で変数を設定しても、他方の環境には影響を与えません。

関連情報

i5/OS PASE ILE プロシージャー API

_RETURN()--i5/OS PASE を終了せずに戻る

QP2SHELL() を使用した i5/OS PASE プログラムの実行

任意の i5/OS コマンド行から i5/OS PASE プログラムを実行する場合や、任意の高水準言語プログラム、バッチ・ジョブ、または対話式ジョブからプログラムを実行する場合は、QP2SHELL または QP2SHELL2 プログラムを使用します。

これらのプログラムは、呼び出し元のジョブの中で i5/OS PASE プログラムを実行します。プログラムでは、i5/OS PASE プログラムの名前がパラメーターとして渡されます。

QP2SHELL() プログラムは、新しい活動化グループで i5/OS PASE プログラムを実行します。
QP2SHELL2() プログラムは、呼び出し側の活動化グループで実行されます。

次の例では、i5/OS のコマンド行から `ls` コマンドを実行します。

```
call qp2shell parm('/QOpenSys/bin/ls' '/')
```

CL 変数を使用して QP2SHELL() に値を渡す場合、変数はヌル終了でなければなりません。たとえば、上のサンプルは次のような方法でコーディングする必要があります。

```
PGM  DCL  VAR(&CMD)    TYPE(*CHAR)   LEN(20)    VALUE('/QOpenSys/bin/ls')
DCL  VAR(&PARM1)  TYPE(*CHAR)   LEN(10)    VALUE('/')
DCL  VAR(&NULL)   TYPE(*CHAR)   LEN(1)     VALUE(X'00')

                  CHGVAR VAR(&CMD) VALUE(&CMD *TCAT &NULL)
                  CHGVAR VAR(&PARM1) VALUE(&PARM1 *TCAT &NULL)

                  CALL PGM(QP2SHELL) PARM(&CMD &PARM1)

ENDIT:
ENDPGM
```

関連情報

QP2SHELL() および QP2SHELL2()--i5/OS PASE シェル・プログラムの実行

QP2TERM() での i5/OS PASE プログラムの実行

この i5/OS プログラムを使用して、対話式シェル環境で i5/OS PASE プログラムを実行します。

QP2TERM() プログラムで i5/OS PASE 対話式端末セッションを開始します。

以下のコマンドは、デフォルトの Korn シェル・プロンプト (/QOpenSys/usr/bin/sh) を画面に表示します。

```
call qp2term
```

このプロンプトから、 i5/OS PASE プログラムを個別のバッチ・ジョブとして実行します。 QP2TERM() は、対話式ジョブを使用して、バッチ・ジョブでファイル stdin、stdout、および stderr の出力の表示および入力の受け入れを行います。

Korn シェルがデフォルトですが、実行する任意の i5/OS PASE プログラムへ渡す任意の引数ストリングの他に、そのプログラムのパス名を指定することもできます。

QP2TERM() で開始する対話式セッションからは、任意の i5/OS PASE プログラムおよび任意のユーティリティーを実行でき、 stdout および stderr が、端末の画面に表示およびスクロールされます。

関連情報

QP2TERM()--i5/OS PASE 端末セッションの実行

i5/OS PASE シェルおよびユーティリティー

i5/OS プログラム内からの i5/OS PASE プログラムの実行

このトピックのステップに従って、他の ILE プロシージャー内から Qp2CallPase() および Qp2CallPase2() ILE プロシージャーを呼び出して、 i5/OS PASE プログラムを開始し、実行することができます。以下に例を示します。

i5/OS PASE プログラムを実行するには、Qp2RunPase() API を使用します。 プログラム名、引数ストリング、および環境変数を指定してください。

Qp2RunPase() API は、呼び出し元のジョブの中で i5/OS PASE プログラムを実行します。これは i5/OS PASE プログラム (必要な共用ライブラリーをすべて含む) をロードし、プログラムに制御を渡します。

この API では、QP2SHELL() や QP2TERM() に比べ、より幅広く i5/OS PASE の実行方法を制御できます。

関連情報

Qp2RunPase()--i5/OS PASE プログラムの実行

例: i5/OS プログラム内からの i5/OS PASE プログラムの実行:

このトピックで示されている例は、i5/OS PASE プログラムを呼び出す ILE プログラム、およびその ILE プログラムによって呼び出される i5/OS PASE プログラムを示しています。

注: コード例を使用することにより、 62 ページの『コードに関する特記事項』 の条件に同意することになります。

例 1: i5/OS PASE プログラムを呼び出す ILE プログラム

以下の ILE プログラムは、i5/OS PASE プログラムを呼び出します。このサンプルに続いて、このプログラムが呼び出す i5/OS PASE コードのサンプルを示します。

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

/* include file for QP2RunPase(). */

#include <qp2user.h>

/*****************************************
  Sample:
```

```
  A simple ILE C program to invoke an i5/OS
  PASE program using QP2RunPase() and
  passing one string parameter.
  Example compilation:
    CRTCMOD MODULE(MYLIB/SAMPLEILE) SRCFILE(MYLIB/QCSRC)
    CRTPGM PGM(MYLIB/SAMPLEILE)
****************************************/

void main(int argc, char*argv[])
 {
  /* Path name of PASE program */
  char *PasePath = "/home/samplePASE";
  /* Return code from QP2RunPase() */
  int rc;
  /* The parameter to be passed to the
     i5/OS PASE program */
  char *PASE_parm = "My Parm";
  /* Argument list for i5/OS PASE program,
     which is a pointer to a list of pointers */
  char **arg_list;
  /* allocate the argument list */
  arg_list =(char**)malloc(3 * sizeof(*arg_list));
  /* set program name as first element. This is a UNIX convention */
  arg_list[0] = PasePath;
  /* set parameter as first element */
  arg_list[1] = PASE_parm;
  /* last element of argument list must always be null */
  arg_list[2] = 0;
  /* Call i5/OS PASE program. */
   rc = Qp2RunPase(PasePath, /* Path name */
      NULL,           /* Symbol for calling to ILE, not used in this sample */
      NULL,           /* Symbol data for ILE call, not used here */
      0,              /* Symbol data length for ILE call, not used here */
      819,            /* ASCII CCSID for i5/OS PASE */
      arg_list,       /* Arguments for i5/OS PASE program */
      NULL);          /* Environment variable list, not used in this sample */
 }
```

例 2: ILE プログラムで呼び出される i5/OS PASE プログラム

上記の ILE プログラムにより、以下の i5/OS PASE プログラムが呼び出されます。

```
#include <stdio.h>

/****************************************
  Sample:
  A simple i5/OS PASE Program called from
  ILE using QP2RunPase() and accepting
  one string parameter.
  The ILE sample program expects this to be
  located at /home/samplePASE. Compile on
  AIX, then ftp to i5/OS.
  To ftp use the commands:
  > binary
  > site namefmt 1
  > put samplePASE /home/samplePASE
****************************************/

int main(int argc, char *argv[])
{
    /* Print out a greeting and the parameter passed in. Note argv[0] is the program
       name, so, argv[1] is the parameter */
     printf("Hello from i5/OS PASE program %s. Parameter value is ¥"%s¥".¥n", argv[0], argv[1]);

     return 0;
}
```

i5/OS プログラム内からの i5/OS PASE プロシージャーの呼び出し

他の ILE プロシージャー内から Qp2CallPase() および Qp2CallPase2() ILE プロシージャーを呼び出して、i5/OS PASE 環境がすでに稼働しているジョブで i5/OS PASE プログラムを実行することができます。

最初に Qp2RunPase() API が、ジョブの中で i5/OS PASE プログラムを開始および実行します。そのジョブで i5/OS PASE がすでにアクティブになっている場合は、エラーが戻されます。

i5/OS PASE プログラムがすでに実行されているジョブの中で i5/OS PASE プロシージャーを呼び出すには、 Qp2CallPase() および Qp2CallPase2() API を使用します。

関連情報

Qp2CallPase()--i5/OS PASE プロシージャーの呼び出し

例 1: i5/OS プログラム内からの i5/OS PASE プロシージャーの呼び出し:

このトピックの例は、i5/OS PASE プロシージャーを呼び出す ILE プログラムを示しています。

注: コード例を使用することにより、 62 ページの『コードに関する特記事項』 の条件に同意することになります。

```
#include <stdio.h>
#include <qp2shell2.h>
#include <qp2user.h>
#define JOB_CCSID 0

int main(int argc, char *argv[])
{
    QP2_ptr64_t id;
    void *getpid_pase;
    const QP2_arg_type_t signature[] = { QP2_ARG_END };
    QP2_word_t result;

    /*
     * Call QP2SHELL2 to run the i5/OS PASE program
     * /usr/lib/start32, which starts i5/OS PASE in
     * 32-bit mode (and leaves it active on return)
     */
    QP2SHELL2("/usr/lib/start32");

    /*
     * Qp2dlopen opens the global name space (rather than
     * loading a new shared executable) when the first
     * argument is a null pointer.  Qp2dlsym locates the
     * function descriptor for the i5/OS PASE getpid
     * subroutine (exported by shared library libc.a)
     */
    id = Qp2dlopen(NULL, QP2_RTLD_NOW, JOB_CCSID);
    getpid_pase = Qp2dlsym(id, "getpid", JOB_CCSID, NULL);

    /*
     * Call Qp2CallPase to run the i5/OS PASE getpid
     * function, and print the result.  Use Qp2errnop
     * to find and print the i5/OS PASE errno if the
     * function result was -1
     */
    int rc = Qp2CallPase(getpid_pase,
                         NULL,        // no argument list
                         signature,
                         QP2_RESULT_WORD,
                         &result)
    printf("i5/OS PASE getpid() = %i¥n", result);
    if (result == -1)
        printf("i5/OS errno = %i¥n", *Qp2errnop());
```

```
    /*
     *  Close the Qp2dlopen instance, and then call
     *  Qp2EndPase to end i5/OS PASE in this job
     */
    Qp2dlclose(id);
    Qp2EndPase();
    return 0;
}
```

例 2: i5/OS PASE プロシージャーの呼び出しでポインター引数を使用する i5/OS ILE プログラム:

この例では、i5/OS ILE プログラムは 2 つの異なる手法を使用して、呼び出す i5/OS PASE プロシージャーでのメモリー・ストレージの割り振りと共有を行っています。

注: 以下のコード例を使用することにより、 62 ページの『コードに関する特記事項』 の条件に同意することになります。

```
/* Name: ileMain.c
 *
 * Call an i5/OS PASE procedure from ILE
 *
 * This example uses the Qp2dlopen, Qp2dlsym, and Qp2CallPase2 ILE
 * functions to call an i5/OS PASE function passing in parameters
 *
 * Compile like so:
 *
 * CRTBNDC PGM(mylib/ilemain)
 *         SRCFILE(mylib/mysrcpf)
 *         TERASPACE(*YES *TSIFC)
 */
#include <stdio.h>
#include <stddef.h>
#include <errno.h>
#include <qp2user.h>
/* Use EBCDIC default job CCSID in Qp2dlopen and Qp2dlsym calls */
#define JOB_CCSID 0

/* start i5/OS PASE in this process */
void startPASE(void) {
    /* start64 starts the 64 bit version of i5/OS PASE */
    char *start64Path="/usr/lib/start64";
    char *arg_list[2];

    arg_list[0] = start64Path;
    arg_list[1] = NULL;
    Qp2RunPase(start64Path,
               NULL,
               NULL,
               0,
               819,
               (char**)&arg_list,
               NULL);

}

/* open a shared library */

QP2_ptr64_t openlib(char * libname) {
    QP2_ptr64_t id;
    int * paseErrno;

    /* Qp2dlopen dynamically loads the specified library returning an
     * id value that can be used in calls to Qp2dlsym and Qp2dlcose */
    id = Qp2dlopen(libname,
                   (QP2_RTLD_NOW |
```

```
                     QP2_RTLD_MEMBER ),
                    JOB_CCSID);
    if (id == 0) {
        printf("Qp2dlopen failed.  ILE errno=%i¥n", errno);
        if ((paseErrno=Qp2errnop()) != NULL)
            printf("Qp2dlopen failed.  i5/OS PASE errno=%i¥n", *paseErrno);
        printf("Qp2dlopen failed.  Qp2dlerror = %s¥n", Qp2dlerror());
    }

    return(id);
}

/* find an exported symbol */

void * findsym(const QP2_ptr64_t id, const char * functionname) {
    void * symbol;
    int * paseErrno;

    /* Qp2dlsym locates the function descriptor for the
     * specified function */
    symbol = Qp2dlsym(id, functionname, JOB_CCSID, NULL);
    if (symbol == NULL) {
        printf("Qp2dlsym failed.  ILE errno = %i¥n", errno);
        if ((paseErrno=Qp2errnop()) != NULL)
            printf("Qp2dlsym failed.  i5/OS PASE errno=%i¥n", *paseErrno);
        printf("Qp2dlsym failed.  Qp2dlerror = %s¥n", Qp2dlerror());
    }
    return(symbol);
}

/* call i5/OS PASE procedure */
int callPASE(const void * functionsymbol,
             const void * arglist,
             const QP2_arg_type_t * signature,
             const QP2_result_type_t result_type,
             void * buf,
             const short buflen) {
    int * paseErrno;
    int rc;

    /* Call Qp2CallPase2 to run the unction function */
    rc = Qp2CallPase2(functionsymbol,
                      arglist,
                      signature,
                      result_type,
                      buf,
                      buflen);
    if (rc != 0) {
        printf("Qp2CallPase failed.  rc=%i, ILE errno=%i¥n", rc, errno);
        if ((paseErrno=Qp2errnop()) != NULL)
            printf("Qp2CallPase failed.  i5/OS PASE errno=%i¥n", *paseErrno);
        printf("Qp2CallPase failed.  Qp2dlerror=%s¥n", Qp2dlerror());
    }
}


int main(int argc, char *argv[])
{
    /* we will call a function in i5/OS PASE named "paseFunction"
     * the prototype for the function looks like this:
     *  int paseFunction(void * input, void * output ) */

    /* "signature" is the argument signature for the PASE routine "paseFunction" */
    const QP2_arg_type_t signature[] = {QP2_ARG_PTR64, QP2_ARG_PTR64, QP2_ARG_END};

    /* "paseFunctionArglist" are the arguments for the PASE routine "paseFunction" */
    struct {
```

```
|          QP2_ptr64_t inputPasePtr;
|          QP2_ptr64_t outputPasePtr;
|      } paseFunctionArglist;
|
|
|      /* "inputString" will be one of the arguments to the PASE routine
|       * "paseFunction" we will call
|       * This is the string "input" in ASCII */
|      const char inputString[] = {0x69, 0x6e, 0x70, 0x75, 0x74, 0x00};
|
|      /* "outputILEPtr" will be a pointer to storage malloc'd from PASE heap */
|      char * outputILEPtr;
|
|      /* "id" is the identifier for the library opened by Qp2dlopen */
|      QP2_ptr64_t id;
|
|      /* "paseFunction_ptr" is the pointer to the routine "paseFunction" in PASE */
|      void * paseFunction_ptr;
|
|      /* "inputAndResultBuffer" is the buffer of storage shared between ILE and PASE
|       * by Qp2CallPase2.  This buffer contains space for the PASE function result */
|      struct {
|          QP2_dword_t result;
|          char inputValue[6];
|      } inputAndResultBuffer;
|
|      int rc;
|      int * paseErrno;
|
|      /* start i5/OS PASE in this process */
|      startPASE();
|
|      id = openlib("/home/joeuser/libpasefn.a(shr64.o)");
|
|      if (id !=0) {
|          /* Locate the symbol for "paseFunction" */
|          paseFunction_ptr = findsym(id, "paseFunction");
|
|          if (paseFunction_ptr != NULL) {
|
|              /* set input arguments for the call to paseFunction() */
|
|              /* copy the inputString into the inputAndResultBuffer */
|              strcpy(inputAndResultBuffer.inputValue, inputString);
|
|              /* by setting inputPasePtr argument to the offset of the
|               * inputValue by-address argument data in the
|               * inputAndResultbuffer structure and OR'ing that with
|               * QP2_ARG_PTR_TOSTACK QP2CallPase2 will "fixup" the
|               * actual argument pointer passed to the PASE function
|               * to point to the address (plus the offset) of the
|               * copy of the inputAndResultbuffer that Qp2CallPase2
|               * copies to i5/OS PASE storage */
|              paseFunctionArglist.inputPasePtr =
|              (QP2_ptr64_t)((offsetof(inputAndResultBuffer, inputValue))
|                             | QP2_ARG_PTR_TOSTACK);
|
|              /* allocate memory from the i5/OS PASE heap for an output
|               * argument.  Qp2malloc will also set the i5/OS PASE address
|               * of the allocated storage in the outputPasePtr
|               * argument */
|              outputILEPtr = Qp2malloc(10, &(paseFunctionArglist.outputPasePtr));
|
|              /* Call the function in i5/OS PASE */
|              rc = callPASE(paseFunction_ptr,
|                             &paseFunctionArglist,
|                             signature,
```

```
                            QP2_RESULT_DWORD,
                            &inputAndResultBuffer,
                            sizeof(inputAndResultBuffer));
                if (rc != 0) {

                    printf("output from paseFunction = >%s<¥n",
                           (char*)outputILEPtr);
                    printf("return code from paseFunction = %d¥n",
                           (int)inputAndResultBuffer.result);
                } /* rc != 0 */
            } /* paseFunction_ptr != NULL */
        } /* id != 0 */

        /* Close the Qp2dlopen instance, and then call Qp2EndPase
         * to end i5/OS PASE in this job */
        Qp2dlclose(id);
        Qp2EndPase();
        return 0;
    }

    Source code for the i5/OS Procedure paseFunction that is called by the ileMain.c program:

    /* i5/OS PASE function to be called from ILE
     *
     * Compile with something like:
     *  xlc -q64 -c -o paseFunction.o paseFunction.c
     *  ld -b64 -o shr64.o -bnoentry -bexpall -bM:SRE -lc paseFunction.o
     *  ar -X64 -r /home/joeuser/libpasefn.a shr64.o
     *
     * The ILE side of this example expects to find libpasefn.a in
     * /home/joeuser/libpasefn.a
     *
     * The compiler options -qalign=natural and -qldbl128 are
     * necessary only when interacting with i5/OS ILE programs
     * to force relative 16-byte alignment of type long double
     * (used inside type ILEpointer)
     */

    #include <stdlib.h>
    #include <stdio.h>
    int paseFunction(void * inputPtr, void * outputPtr)
    {
        /* An output string to return from i5/OS PASE to ILE *
         * this is the string "output" in EBCDIC             */
        const char outputValue[] = {0x96, 0xa4, 0xa3, 0x97, 0xa4, 0xa3, 0x00};


        printf("Entered paseFunction The input is >%s<¥n",
               (char*)inputPtr);

        /* copy the output results to the outputPtr argument */
        memcpy(outputPtr, outputValue, sizeof(outputValue));


        return(52); /* return something more interesting than 0 */
    }
```

例 2 の ILE の部分で使用されるさまざまな関数

- **startPASE() 関数**

  i5/OS PASE をプロセスで使用する前に、これを開始しておく必要があります。これは、API (たとえば QP2SHELL、QP2TERM、または Qp2RunPase) を使用して i5/OS PASE アプリケーションのメインエントリー・ポイントを呼び出すことにより自動的に行うことができます。

しかし、この例は (メインエントリー・ポイントではなく) 共用ライブラリーからエクスポートされた i5/OS PASE 関数を呼び出しているので、i5/OS PASE を手動で開始する必要があります。これを行うには、/usr/lib/start32 (32 ビット・バージョンの i5/OS PASE を開始する) および /usr/lib/start64 (64 ビット・バージョンの i5/OS PASE を開始する) という 2 つの i5/OS PASE 開始ユーティリティーを使用できます。

i5/OS プロセスはそれぞれ 1 つの i5/OS PASE インスタンスしか実行させておくことができないという点にご注意ください。 Qp2ptrsize() API を使用すると、i5/OS PASE がすでに実行されているかどうかを判別することができます。

- i5/OS PASE が現在プロセスでアクティブになっていない場合、Qp2ptrsize() は 0 を戻します。
- i5/OS PASE が 32 ビット・モードでアクティブになっている場合、Qp2ptrsize() は 4 を戻します。
- i5/OS PASE が 64 ビット・モードでアクティブになっている場合、Qp2ptrsize() は 8 を戻します。

- **openlib() 関数および findsym() 関数**

これらの関数は Qp2dlopen() および Qp2dlsym() を使用して、i5/OS 共用ライブラリーをオープンし、呼び出したい関数のへのポインターを取得します。これらの関数は多くのプラットフォームにおける dlopen() ルーチンおよび dlsym() ルーチンと類似しています。

- **Qp2CallPase2 呼び出しのための引数のセットアップ**

callPASE() 関数によって Qp2CallPase2() を呼び出す前に、main() ルーチンは、ILE と i5/OS PASE 関数の間のインターフェースを定義する以下の変数をセットアップします。

- signature-array 変数は、i5/OS PASE 関数の引数を定義します。配列中のエレメントは一般に、qsysinc/h.qp2user 組み込みファイル内の #define を使用して設定されます。
- paseFunctionArglist 構造には、i5/OS PASE ランタイムが引数 (関数の呼び出し時に i5/OS PASE 関数に渡される) にマップする ILE 変数が含まれています。 paseFunctionArglist のメンバーは、シグニチャー配列で宣言されている i5/OS PASE 関数のシグニチャーと対応します。
- inputAndResultBuffer 構造には、関数の呼び出し時に i5/OS PASE ランタイムが ILE と i5/OS PASE の間の一種の共用バッファーとして使用する ILE 変数が含まれています。

  構造の最初のメンバー (この例の result) には、i5/OS PASE 関数からの戻り値が含まれます。この変数は、Qp2CallPase2 API の呼び出しの 4 つ目の引数として提供される結果タイプと一致しなければなりません。この最初のエレメントの後に来るものはいずれも、関数の呼び出し時に i5/OS PASE 環境にコピーされるストレージを表します。

  この例の inputAndResultBuffer 構造の inputValue エレメントには、i5/OS PASE 関数の最初の引数によって指し示されるアドレス渡しの引数データが入ります。
- この例では、呼び出されている i5/OS PASE 関数のポインター引数の、2 つの異なる設定方法が使用されています。
  - 関数の 2 番目の引数 paseFunctionArglist.outputPasePtr は、Qp2malloc() 関数を呼び出すことによって設定されます。 Qp2malloc() は、i5/OS PASE ランタイム・ヒープからメモリーを割り振り、割り振られたストレージに、ILEpointer と i5/OS PASE ポインターの両方を戻します。
  - 最初の引数 paseFunctionArglist.inputPasePtr は、inputAndResultBuffer 構造の inputValue エレメントのオフセットに設定されます (OR によって qp2user.h #define QP2_ARG_PTR_TOSTACK とつながれます)。

| これは、i5/OS PASE ランタイムに、アドレス (inputAndResultBuffer.inputValue が i5/OS PASE メモリーにコピーされた) による i5/OS PASE 関数の呼び出しによって提供される実際のポインター値を変更するように命令します。

- **callPASE() 関数**

  この関数は、Qp2CallPase2() API、および main() ルーチンで設定される引数を使用して、i5/OS PASE 関数を呼び出します。

- **プロセスでの i5/OS PASE の終了**

  i5/OS PASE 関数を呼び出した後、i5/OS PASE 共有ライブラリーをアンロードするために Qp2dlclose() API が呼び出され、この例の最初に呼び出された start64 プログラムを終了するために Qp2EndPase() が呼び出されます。

Java からの i5/OS PASE ネイティブ・メソッドの使用

i5/OS PASE 環境で実行される i5/OS PASE ネイティブ・メソッドを、 Java プログラムから使用することができます。

i5/OS PASE ネイティブ・メソッドのサポートには、 i5/OS PASE ネイティブ・メソッドからのすべての iSeries Java ネイティブ・インターフェース (JNI) の使用、およびネイティブ iSeries JVM から i5/OS PASE ネイティブ・メソッドを呼び出す機能が含まれます。

関連情報

IBM i5/OS PASE native methods for Java

環境変数の処理

i5/OS PASE 環境変数は ILE 環境変数に依存しません。一方の環境で変数を設定しても、他方の環境には影響を与えません。

ただし、i5/OS PASE プログラムの実行に使用する方法に応じて、 ILE から i5/OS PASE に変数をコピーすることができます。

対話式 i5/OS PASE セッションの環境変数

ILE 環境変数は、QP2SHELL() および QP2TERM() を使用して開始される場合にのみ、 i5/OS PASE に渡されます。 i5/OS PASE を開始する前に、環境変数の処理 (WRKENVVAR) コマンドを使用して、環境変数の変更、追加、または削除を行います。

呼び出し先 i5/OS PASE セッションの環境変数

i5/OS PASE が (Qp2RunPase() API を使用した) プログラム呼び出しから開始される場合、環境変数に対する完全制御が与えられます。 i5/OS PASE プログラムの呼び出し元 ILE 環境と関係のない環境変数を渡すことができます。

CL コマンドを実行する前に ILE に環境変数をコピーする

systemCL ランタイム機能にオプションを指定して CL コマンドを実行する前に、 ILE 環境に i5/OS PASE 環境変数をコピーすることができます。これは、i5/OS PASE `system` ユーティリティーのデフォルトの動作でもあります。

関連資料

21 ページの『i5/OS PASE プログラムおよびプロシージャーの実行』
このトピックでは、ジョブでの i5/OS PASE プログラムの開始、および ILE プログラムからの i5/OS PASE プロシージャーの呼び出しに関する情報と例が提供されています。

関連情報

QP2SHELL() および QP2SHELL2()--i5/OS PASE シェル・プログラムの実行

QP2TERM()--i5/OS PASE 端末セッションの実行

systemCL()--i5/OS PASE 用の CL コマンドの実行

i5/OS PASE 環境変数

i5/OS PASE プログラムからの i5/OS プログラムおよびプロシージャーの呼び出し

i5/OS PASE では、ILE プロシージャー、Java プログラム、OPM プログラム、i5/OS API、および i5/OS 機能への統合アクセスを持つ CL コマンドを呼び出すためのメソッドを提供します。

i5/OS プログラムおよびプロシージャーの一般構成要件

i5/OS PASE プログラム環境から i5/OS 環境に呼び出しを行う場合、一般に i5/OS プログラムが活動化グループの *CALLER でコンパイルされていることを確認する必要があります。それには、以下の理由があります。

- i5/OS PASE (Qp2RunPase API によって呼び出される) を開始した活動化グループ内で実行するコードだけが、Qp2CallPase などの ILE API を使用して i5/OS PASE プログラムと対話できる。
- ILE ランタイムは、マルチスレッド・ジョブ内の活動化グループを破壊する必要がある場合、 (i5/OS PASE `fork` が作成するすべてのジョブはマルチスレッド対応)、ジョブ全体を終了してしまう場合がある (i5/OS PASE を終了することもある)。 ACTGRP(*CALLER) を使用すると、ジョブを終了しようとする前に、ジョブが終了してしまうことを避けられます。

systemCL ランタイム機能を使用して CL コマンド (CALL コマンドを含む) をマルチスレッド対応でない別個のジョブで実行すると、マルチスレッド対応のジョブで実行することに伴う問題を避けることができます。

関連タスク

18 ページの『i5/OS 機能を使用するための i5/OS PASE プログラムのカスタマイズ』
AIX アプリケーションで、システム提供の i5/OS PASE 共用ライブラリーでは直接サポートされていない i5/OS の機能を利用したい場合は、いくらかの付加的なステップを実行してアプリケーションを準備する必要があります。

ILE プロシージャーの呼び出し

このトピックの説明に従って、ILE プロシージャーを準備し、i5/OS PASE プログラムから呼び出すことができます。

i5/OS PASE プログラムから ILE プロシージャーを呼び出す際には、まず ILE プロシージャーを、テラスペース用に使用可能化し、テキストを適切な CCSID に変換し、変数および構造をセットアップして準備する必要があります。

- **テラスペース用に ILE プロシージャーを使用可能にする**

  i5/OS PASE から呼び出す ILE モジュールをコンパイルする際に、常にテラスペース・オプションを *YES に設定する必要があります。 ILE モジュールがこの仕方でコンパイルされていないと、 i5/OS

PASE アプリケーションのジョブ・ログに MCH4433 エラー・メッセージ (ターゲット・プログラム &2 のストレージ・モデルは無効です (Invalid storage model for target program &2)) が記録されます。

- **テキストを適切な CCSID に変換する**

  ILE と i5/OS PASE の間で渡されるテキストは、事前に適切な CCSID に変換しておかなければならない場合があります。この変換を行わないと、文字変数の中に判読できない値が入ってしまいます。

- **変数と構造をセットアップする**

  i5/OS PASE プログラムから ILE を呼び出すには、変数と構造をセットアップする必要があります。必要なヘッダー・ファイルを AIX システムに確実にコピーし、シグニチャー、結果タイプ、および引数リスト変数をセットアップしなければなりません。

  – **ヘッダー・ファイル:** ILE を呼び出すには、i5/OS PASE プログラムにヘッダー・ファイル as400_types.h および as400_protos.h が含まれていなければなりません。 as400_type.h ヘッダー・ファイルには、 i5/OS システム固有のインターフェースで使用されるタイプの定義が含まれています。

  – **シグニチャー:** シグニチャー構造には、順序の説明と、 i5/OS PASE と ILE の間で渡される引数のタイプが含まれます。呼び出そうとしている ILE プロシージャーによって指示されるタイプのエンコードは、 as400_types.h header ファイルにあります。シグニチャーに 4 バイト以下の固定小数点引数、または 8 バイト以下の浮動小数点引数が含まれる場合、次のプラグマを使用して、ILE C コードをコンパイルする必要があります。

    ```
    #pragma argument(ileProcedureName, nowiden)
    ```

    このプラグマを使用しない場合、 ILE への標準 C リンクで、1 バイトまたは 2 バイトの整数引数を 4 バイトに、 4 バイトの浮動小数点引数を 8 バイトに拡張する必要があります。

  – **結果タイプ:** 結果タイプは C の戻りタイプの動作と類似しており、複雑ではありません。

  – **引数リスト:** 引数リストは正しい順序のフィールドを持つ構造でなければなりません。そのタイプはシグニチャー配列のエントリーによって指定されます。 size_ILEarglist() および build_ILEarglist() API を使用することにより、シグニチャーに基づいて引数リストを動的に作成することができます。

i5/OS PASE プログラムから ILE プロシージャーを呼び出すには、コード内で以下の API 呼び出しを行います。

1. i5/OS PASE を開始したプロシージャーに関連する ILE 活動化グループに、結合プログラムをロードします。これを行うには、_ILELOAD() API を使用します。

   i5/OS PASE を開始した活動化グループで結合プログラムがすでにアクティブになっている場合、このステップは不要になる場合があります。この場合、_ILESYM のステップに進むことができます。活動化マーク・パラメーターにゼロを指定し、現行の活動化グループのすべてのアクティブ結合プログラムに含まれるすべての記号を検索します。

2. ILE 結合プログラムを活動化するときにエクスポート済み記号を検索し、記号のデータまたはプロシージャーに 16 バイトのタグ付きポインターを戻します。これを行うには、_ILESYM() API を使用します。

3. ILE プロシージャーを呼び出して、 i5/OS PASE プログラムから ILE プロシージャーに制御を転送します。これを行うには、_ILECALL() または _ILECALLX() API を使用します。

関連資料

18 ページの『ヘッダー・ファイルのコピー』
このトピックの説明に従って、iSeries サーバーから AIX マシンにヘッダー・ファイルをコピーすることができます。

関連情報

size_ILEarglist()--i5/OS PASE() 用の ILE 引数リスト・サイズの計算

build_ILEarglist()--i5/OS PASE 用の ILE 引数リストの作成

_ILELOADX()--i5/OS PASE 用の ILE 結合プログラムのロード

_ILESYMX()--i5/OS PASE 用のエクスポート済み ILE シンボルの検索

_ILECALLX()--i5/OS PASE 用の ILE プロシージャーの呼び出し

ILE 概念 PDF

例: ILE プロシージャーの呼び出し:

このトピックのコード例は、サービス・プログラムの一部である ILE プロシージャーを呼び出すための i5/OS PASE コードと、プログラムを作成するためのコンパイラー・コマンドを示しています。

以下のコード例の中には、サービス・プログラムの一部である ILE プロシージャーを呼び出すための i5/OS PASE コードと、 プログラムを作成するためのコンパイラー・コマンドを示す 2 つのプロシージャーがあります。 ILE プロシージャーの処理方法は異なりますが、どちらも同じ ILE プロシージャーを呼び出します。最初のプロシージャーは、i5/OS PASE システムが提供するメソッドを使用した、 _ILECALL API のデータ構造の構築を示しています。 2 番目のプロシージャーは、手動による引数リストの作成を示しています。

注: コード例を使用することにより、 62 ページの『コードに関する特記事項』 の条件に同意することになります。

例 1: i5/OS PASE C コード

以下の例には、コードを説明するコメントが含まれています。例を入力したり検討したりする際には、これらのコメントを必ずお読みください。

```
/* Name: PASEtoILE.c
 *
 * You must use compiler options -qalign=natural and -qldbl128
 * to force relative 16-byte alignment of type long double
 * (used inside type ILEpointer)
 *
 */
#include <stdlib.h>
#include <malloc.h>
#include <sys/types.h>
#include <stdio.h>
#include "as400_types.h"
#include "as400_protos.h"

/*
 * init_pid saves the process id (PID) of the process that
 * extracted the ILEpointer addressed by ILEtarget.
 * init_pid is initialized to a value that is not a
 * valid PID to force initialization on the first
 * reference after the exec() of this program
 *
 * If your code uses pthread interfaces, you can
 * alternatively provide a handler registered using
 * pthread_atfork() to re-initialize ILE procedure
 * pointers in the child process and use a pointer or
 * flag in static storage to force reinitialization
 * after exec()
 */

pid_t  init_pid = -1;
ILEpointer*ILEtarget;  /* pointer to ILE procedure */
```

```
/*
 * ROUND_QUAD finds a 16-byte aligned memory
 * location at or beyond a specified address
 */

#define ROUND_QUAD(x) (((size_t)(x) + 0xf) & ~0xf)

/*
 * do_init loads an ILE service program and extracts an
 * ILEpointer to a procedure that is exported by that
 * service program.
 */

void do_init()
{
   static char ILEtarget_buf[sizeof(ILEpointer) + 15];
   int actmark;
   int rc;

   /* _ILELOAD() loads the service program */
   actmark = _ILELOAD("SHUPE/ILEPASE", ILELOAD_LIBOBJ);
   if (actmark == -1)
   abort();

   /*
    * xlc does not guarantee 16-byte alignment for
    * static variables of any type, so we find an
    * aligned area in an oversized buffer. _ILESYM()
    * extracts an ILE procedure pointer from the
    * service program activation
    */

   ILEtarget = (ILEpointer*)ROUND_QUAD(ILEtarget_buf);
   rc = _ILESYM(ILEtarget, actmark, "ileProcedure");
   if (rc == -1)
   abort();

   /*
    * Save the current PID in static storage so we
    * can determine when to re-initialize (after fork)
    */
   init_pid = getpid();
}

/*
 * "aggregate" is an example of a structure or union
 * data type that is passed as a by-value argument.
 */
typedef struct {
    char        filler[5];
} aggregate;

/*
 * "result_type" and "signature" define the function
 * result type and the sequence and type of all
 * arguments needed for the ILE procedure identified
 * by ILEtarget
 *
 * NOTE: The fact that this argument list contains
 * fixed-point arguments shorter than 4 bytes or
 * floating-point arguments shorter than 8 bytes
 * implies that the target ILE C procedure is compiled
 * with #pragma argument(ileProcedureName, nowiden)
 *
 * Without this pragma, standard C linkage for ILE
 * requires 1-byte and 2-byte integer arguments to be
```

```
 * widened to 4-bytes and requires 4-byte floating-point
 * arguments to be widened to 8-bytes
 */
static result_type_tresult_type = RESULT_INT32;
static arg_type_tsignature[] =
{
    ARG_INT32,
    ARG_MEMPTR,
    ARG_FLOAT64,
    ARG_UINT8,      /* requires #pragma nowiden in ILE code */
    sizeof(aggregate),
    ARG_INT16,
    ARG_END
};

/*
 * wrapper_1 accepts the same arguments and returns
 * the same result as the ILE procedure it calls. This
 * example does not require a customized or declared structure
 * for the ILE argument list. This wrapper uses malloc
 * to obtain storage. If an exception or signal occurs,
 * the storage may not be freed. If your program needs
 * to prevent such a storage leak, a signal handler
 * must be built to handle it, or you can use the methods
 * in wrapper_2.
 */
int wrapper_1(int arg1, void *arg2, double arg3,
                            char arg4, aggregate arg5, short arg6)
{
    int result;
    /*
     * xlc does not guarantee 16-byte alignment for
     * automatic (stack) variables of any type, but
     * PASE malloc() always returns 16-byte aligned storage.
     * size_ILEarglist() determines how much storage is
     * needed, based on entries in the signature array
     */
    ILEarglist_base *ILEarglist;
    ILEarglist = (ILEarglist_base*)malloc( size_ILEarglist(signature) );

    /*
     * build_ILEarglist() copies argument values into the ILE
     * argument list buffer, based on entries in the signature
     * array.
     */
    build_ILEarglist(ILEarglist,
             &arg1,
             signature);

    /*
     * Use a saved PID value to check if the ILEpointer
     * is set. ILE procedure pointers inherited by the
     * child process of a fork() are not usable because
     * they point to an ILE activation group in the parent
     * process
     */
    if (getpid() != init_pid)
    do_init();

    /*
     * _ILECALL calls the ILE procedure. If an exception or signal
     * occurs, the heap allocation is orphaned (storage leak)
     */
    _ILECALL(ILEtarget,
         ILEarglist,
         signature,
         result_type);
```

```
    result = ILEarglist->result.s_int32.r_int32;
    if (result == 1) {
       printf("The results of the simple wrapper is: %s¥n", (char *)arg2);
    }
    else if (result == 0) printf("ILE received other than 1 or 2 for version.¥n");
    else printf("The db file never opened.¥n");
    free(ILEarglist);
    return result;
}

/*
 * ILEarglistSt defines the structure of the ILE argument list.
 * xlc provides 16-byte (relative) alignment of ILEpointer
 * member fields because ILEpointer contains a 128-bit long
 * double member. Explicit pad fields are only needed in
 * front of structure and union types that do not naturally
 * fall on ILE-mandated boundaries
 */
typedef struct {
    ILEarglist_base base;
    int32 arg1;
    /* implicit 12-byte pad provided by compiler */
    ILEpointer arg2;
    float64 arg3;
    uint8 arg4;
    char filler[7]; /* pad to 8-byte alignment */
    aggregate arg5; /* 5-byte aggregate (8-byte align) */
    /* implicit 1-byte pad provided by compiler */
    int16 arg6;
} ILEarglistSt;

/*
 * wrapper_2 accepts the same arguments and returns
 * the same result as the ILE procedure it calls. This
 * method uses a customized or declared structure for the
 * ILE argument list to improve execution efficiency and
 * avoid heap storage leaks if an exception or signal occurs
 */
int wrapper_2(int arg1, void *arg2, double arg3,
                        char arg4, aggregate arg5, short arg6)
{
    /*
     * xlc does not guarantee 16-byte alignment for
     * automatic (stack) variables of any type, so we
     * find an aligned area in an oversized buffer
     */
    char ILEarglist_buf[sizeof(ILEarglistSt) + 15];
    ILEarglistSt *ILEarglist = (ILEarglistSt*)ROUND_QUAD(ILEarglist_buf);
    /*
     * Assignment statements are faster than calling
     * build_ILEarglist()
     */
    ILEarglist->arg1 = arg1;
    ILEarglist->arg2.s.addr = (address64_t)arg2;
    ILEarglist->arg3 = arg3;
    ILEarglist->arg4 = arg4;
    ILEarglist->arg5 = arg5;
    ILEarglist->arg6 = arg6;
    /*
     * Use a saved PID value to check if the ILEpointer
     * is set. ILE procedure pointers inherited by the
     * child process of a fork() are not usable because
     * they point to an ILE activation group in the parent
     * process
     */
    if (getpid() != init_pid)
    do_init();
```

```
    /*
     * _ILECALL calls the ILE procedure. The stack may
     * be unwound, but no heap storage is orphaned if
     * an exception or signal occurs
     */
    _ILECALL(ILEtarget,
        &ILEarglist->base,
        signature,
        result_type);
    if (ILEarglist->base.result.s_int32.r_int32 == 1)
      printf("The results of best_wrapper function is: %s¥n", arg2);
    else if ( ILEarglist->base.result.s_int32.r_int32 == 0)
    printf("ILE received other than 1 or 2 for version.¥n");
    else printf("The db file never opened.¥n");
    return ILEarglist->base.result.s_int32.r_int32;
    }
    void main () {
      int version,
                   result2;
      char dbText[ 25 ];
      double dblNumber = 5.999;
      char justChar = 'a';
      short shrtNumber = 3;
      aggregate agg;
      strcpy( dbText, "none" );

      for (version =1; version <=  2; version
        ++) {if(version="=" 1) {
          result2="simple_wrapper(version," dbText, dblNumber, justChar, agg, shrtNumber);
        } else {
          result2="best_wrapper(version," dbText, dblNumber, justChar, agg, shrtNumber);
        }
    }
}
```

例 2: ILE C コード

ここでは、i5/OS システムでこの例の ILE C コードを作成する方法が示されます。コードの作成先のライブラリーにはソース物理ファイルが必要です。この ILE の例にもコメントが含まれています。これらのコメントは、コードを理解する上で重要です。ソースを入力したり検討したりする際には、これらのコメントを検討する必要があります。

```
#include <stdio.h>
#include <math.h>
#include <recio.h>
#include <iconv.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>

typedef struct {
    char        filler[5];
} aggregate;

#pragma mapinc("datafile","SHUPE/PASEDATA(*all)","both",,,"")
#include "datafile"
#pragma argument(ileProcedure, nowiden) /* not necessary */

/*
 * The arguments and function result for this ILE procedure
 * must be equivalent to the values presented to _ILECALL
 * function in the i5/OS PASE program
 */
int ileProcedure(int         arg1,
                 char        *arg2,
                 double      arg3,
```

```
                char        arg4[2],
                aggregate   arg5,
                short       arg6)
{
    char         fromcode[33];
    char         tocode[33];
    iconv_t      cd;     /* conversion descriptor */
    char         *src;
    char         *tgt;
    size_t       srcLen;
    size_t       tgtLen;
    int          result;

    /*
     * Open a conversion descriptor to convert CCSID 37
     * (EBCDIC) to CCSID 819 (ASCII), that is used for
     * any character data returned to the caller
     */
    memset(fromcode, 0, sizeof(fromcode));
    strcpy(fromcode, "IBMCCSID000370000000");
    memset(tocode, 0, sizeof(tocode));
    strcpy(tocode, "IBMCCSID00819");
    cd = iconv_open(tocode, fromcode);
    if (cd.return_value == -1)
    {
        printf("iconv_open failed¥n");
        return -1;
    }
     /*
      * If arg1 equals one, return constant text (converted
      * to ASCII) in the buffer addressed by arg2. For any
      * other arg1 value, open a file and read some text,
      * then return that text (converted to ASCII) in the
      * buffer addressed by arg2
      */
    if (arg1 == 1)
    {
        src = "Sample 1 output text";
        srcLen = strlen(src) + 1;
        tgt = arg2; /* iconv output to arg2 buffer */
        tgtLen = srcLen;
        iconv(cd, &src, &srcLen, &tgt, &tgtLen);

        result = 1;
    }
    else
    {
        FILE *fp;
        fp = fopen("SHUPE/PASEDATA", "r");
        if (!fp) /* if file open error */
        {
             printf("fopen(¥"SHUPE/PASEDATA¥", ¥"r¥") failed, "
                    "errno = %i¥n", errno);
             result = 2;
        }
        else
        {
             char buf[25];
             char *string;
             errno = 0;
             string = fgets(buf, sizeof(buf), fp);
             if (!string)
             {
                  printf("fgets() EOF or error, errno = %i¥n", errno);
                  buf[0] = 0; /* null-terminate empty buffer */
             }
             src = buf;
```

```
            srcLen = strlen(buf) + 1;
            tgt = arg2; /* iconv output to arg2 buffer */
            tgtLen = srcLen;
            iconv(cd, &src, &srcLen, &tgt, &tgtLen);

            fclose(fp);
         }
         result = 1;
    }
      /*
       * Close the conversion descriptor, and return the
       * result value determined above
       */
      iconv_close(cd);
      return result;
}
```

例 3: プログラムを作成するためのコンパイラー・コマンド

i5/OS PASE プログラムをコンパイルする際に、コンパイラー・オプション `-qalign=natural` および `-qldbl128` を使用して、長倍精度実数型の相対 16 バイト調整を強制しなければなりません。これは、ILEpointer 型の中で使用されます。この調整は i5/OS の ILE で必要です。オプション `-bI:` を使用する場合は、 as400_libc.exp の保管先パス名を入力する必要があります。

```
xlc -o PASEtoILE -qldbl128 -qalign=natural
       -bI:/afs/rich.xyz.com/usr1/shupe/PASE/as400_libc.exp
       PASEtoILE.c
```

ILE C モジュールとサービス・プログラムをコンパイルする際には、テラスペース・オプションを使用します。このオプションを使用しないと、i5/OS PASE は ILE C モジュールやサービス・プログラムとの対話が行えません。

```
CRTCMOD MODULE(MYLIB/MYMODULE)
        SRCFILE(MYLIB/SRCPF)
        TERASPACE(*YES *TSIFC)

CRTSRVPGM SRVPGM(MYLIB/MYSRVPGM)
        MODULE(MYLIB/MOMODULE)
```

最後に、DDS をコンパイルして、少なくとも 1 つのデータ・レコードを伝搬する必要があります。

```
CRTPF FILE(MYLIB/MYDATAFILE)
      SRCFILE(MYLIB/SRCDDSF)
      SRCMBR(MYMEMBERNAME)
```

i5/OS PASE からの i5/OS プログラムの呼び出し

i5/OS PASE アプリケーションを作成する際に、既存の i5/OS プログラム (*PGM オブジェクト) を利用することができます。さらに、systemCL() 機能を使用して CL CALL コマンドを実行することができます。

i5/OS PASE プログラム内から i5/OS プログラムを呼び出すには、 _PGMCALL ランタイム機能を使用します。

このメソッドの処理速度は systemCL() ランタイム機能より高速ですが、 (PGMCALL_ASCII_STRINGS を指定しない限り) 文字ストリング引数の自動変換は実行せず、異なるジョブのプログラムを呼び出すための機能も備わっていません。

関連タスク

42 ページの『i5/OS PASE からの i5/OS コマンドの実行』
i5/OS 機能を使用する制御言語 (CL) コマンドを実行することにより、 i5/OS PASE プログラムの機能を拡張することができます。

関連情報

_PGMCALL()--i5/OS PASE 用の i5/OS プログラムの呼び出し

例: i5/OS PASE からの i5/OS プログラムの呼び出し:

このトピックの例をご覧いただくことにより、_PGMCALL ランタイム機能を使用した i5/OS PASE プログラムでのプログラムの呼び出しについて学ぶことができます。

以下の例は、 _PGMCALL ランタイム機能を使用して i5/OS PASE プログラムからプログラムを呼び出す方法を示しています。

以下の例には、コードを説明するコメントが含まれています。例を入力したり検討したりする際には、これらのコメントを必ずお読みください。

注: コード例を使用することにより、 62 ページの『コードに関する特記事項』 の条件に同意することになります。

```
/* This example uses the i5/OS PASE _PGMCALL function to call the i5/OS
API QSZRTVPR. The QSZRTVPR API is used to retrieve information about
i5/OS software product loads. Refer to the QSZRTVPR API documentation
for specific information regarding the input and output parameters needed
to call the API */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "as400_types.h"
#include "as400_protos.h"

int main(int argc, char * argv[])
{

   /* i5/OS API's (including QSZRTVPR) typically expect character
      parameters to be in EBCDIC.  However, character constants in
      i5/OS PASE programs are typically in ASCII. So, declare some
      CCSID 37 (EBCDIC) character parameter constants that will be
      needed to call QSZRTVPR */

   /* format[] is input parameter 3 to QSZRTVPR and is
      initialized to the text 'PRDR0100' in EBCDIC */
   const char format[] =
      {0xd7, 0xd9, 0xc4, 0xd9, 0xf0, 0xf1, 0xf0, 0xf0};

   /* prodinfo[] is input parameter 4 to QSZRTVPR and is
      initialized to the text '*OPSYS *CUR  0033*CODE     ' in EBCDIC

      This value indicates we want to check the code load for Option 33
      of the currently installed i5/OS release */
   const char prodinfo[] =
      {0x5c, 0xd6, 0xd7, 0xe2, 0xe8, 0xe2, 0x40, 0x5c, 0xc3,
       0xe4, 0xd9, 0x40, 0x40, 0xf0, 0xf0, 0xf3, 0xf3, 0x5c,
       0xc3, 0xd6, 0xc4, 0xc5, 0x40, 0x40, 0x40, 0x40, 0x40};

   /* installed will be compared with the "Load State" field of the
      information returned by QSZRTVPR and is initialized to the text
      '90' in EBCDIC */
   const char installed[] = {0xf9, 0xf0};

   /* rcvr is the output parameter 1 from QSZRTVPR */
   char rcvr[108];

   /* rcvrlen is input parameter 2 to QSZRTVPR */
   int rcvrlen = sizeof(rcvr);
```

```
   /* errcode is input parameter 5 to QSZRTVPR */
   struct {
      int bytes_provided;
      int bytes_available;
      char msgid[7];
   } errcode;


   /* qszrtvpr_pointer will contain the i5/OS 16-byte tagged system
      pointer to QSZRTVPR */
   ILEpointer qszrtvpr_pointer;

   /* qszrtvpr_argv6 is the array of argument pointers to QSZRTVPR */
   void *qszrtvpr_argv[6];

   /* return code from _RSLOBJ2 and _PGMCALL functions */
   int rc;

   /* Set the i5/OS pointer to the QSYS/QSZRTVPR *PGM object */
   rc = _RSLOBJ2(&qszrtvpr_pointer,
                 RSLOBJ_TS_PGM,
                 "QSZRTVPR",
                 "QSYS");

   /* initialize the QSZRTVPR returned info structure */
   memset(rcvr, 0, sizeof(rcvr));

   /* initialize the QSZRTVPR error code structure */
   memset(&errcode, 0, sizeof(errcode));
   errcode.bytes_provided = sizeof(errcode);

   /* initialize the array of argument pointers for the QSZRTVPR API */
   qszrtvpr_argv[0] = &rcvr;
   qszrtvpr_argv[1] = &rcvrlen;
   qszrtvpr_argv[2] = &format;
   qszrtvpr_argv[3] = &prodinfo;
   qszrtvpr_argv[4] = &errcode;
   qszrtvpr_argv[5] = NULL;

   /* Call the i5/OS QSZRTVPR API from i5/OS PASE */
   rc = _PGMCALL(&qszrtvpr_pointer,
                 (void*)&qszrtvpr_argv,
                 0);

   /* Check the contents of bytes 63-64 of the returned information.
      If they are not '90' (in EBCDIC), the code load is NOT correctly
      installed */
   if (memcmp(&rcvr[63], &installed, 2) != 0)
      printf("i5/OS Option 33 is NOT installed¥n");
   else
      printf("i5/OS Option 33 IS installed¥n");

   return(0);
}
```

i5/OS PASE からの i5/OS コマンドの実行

i5/OS 機能を使用する制御言語 (CL) コマンドを実行することにより、 i5/OS PASE プログラムの機能を拡張することができます。

systemCL ランタイム機能を使用して、i5/OS PASE プログラム内から i5/OS コマンドを実行します。

i5/OS PASE から i5/OS コマンドを実行する場合、 systemCL ランタイム機能は ASCII から EBCDIC への文字ストリング引数の変換を自動的に処理して、別のジョブでプログラムを呼び出せるようにします。

関連タスク

40 ページの『i5/OS PASE からの i5/OS プログラムの呼び出し』
i5/OS PASE アプリケーションを作成する際に、既存の i5/OS プログラム (*PGM オブジェクト) を利用することができます。さらに、systemCL() 機能を使用して CL CALL コマンドを実行することができます。

関連情報

systemCL()--i5/OS PASE 用の CL コマンドの実行

例: i5/OS PASE からの i5/OS コマンドの実行:

このトピックで提供されている例から、 i5/OS PASE プログラムで CL コマンドを実行する方法を学ぶことができます。

注: コード例を使用することにより、 62 ページの『コードに関する特記事項』 の条件に同意することになります。

以下の例は、i5/OS PASE プログラムからコマンドを呼び出す方法を示しています。

```
/* sampleCL.c
   example to demonstrate use of sampleCL to run a CL command
   Compile with a command similar to the following.
   xlc -o sampleCL -I /whatever/pase -bI:/whatever/pase/as400_libc.exp sampleCL.c
   Example program using QP2SHELL() follows.
   call qp2shell ('sampleCL' 'wrkactjob') */

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <as400_types.h>  /* PASE header */
#include <as400_protos.h> /* PASE header */

void main(int argc, char* argv[])
{
   int rc;

   if (argc!=2)
   {
      printf("usage: %s ¥"CL command¥"¥n", argv[0]);
      exit(1);
   }
   printf("running CL command: ¥"%s¥"¥n", argv[1]);

   /* process the CL command */
   rc = systemCL(argv[1], /* use first parameter for CL command */
                 SYSTEMCL_MSG_STDOUT
                 SYSTEMCL_MSG_STDERR ); /* collect messages */

   printf("systemCL returned %d. ¥n", rc);
   if (rc != 0)
   {
      perror("systemCL");
      exit(rc);
   }
}
```

i5/OS PASE プログラムと i5/OS の相互作用

i5/OS の機能を使用するように i5/OS PASE プログラムをカスタマイズする場合は、プログラムがそれらの機能とどのように相互作用するかを考慮する必要があります。

関連タスク

18 ページの『i5/OS 機能を使用するための i5/OS PASE プログラムのカスタマイズ』
AIX アプリケーションで、システム提供の i5/OS PASE 共用ライブラリーでは直接サポートされていない i5/OS の機能を利用したい場合は、いくらかの付加的なステップを実行してアプリケーションを準備する必要があります。

通信

i5/OS PASE は一般に、ソケット通信において AIX および Linux との互換性があります。

i5/OS PASE は、ソケット通信用に AIX と同じ構文をサポートします。これは、詳細な点では他のオペレーティング・システム (Linux など) と異なる場合があります。

i5/OS PASE ソケット・サポートは AIX のソケット・インプリメンテーションと比較できますが、 i5/OS PASE は (AIX カーネルのソケット・インプリメンテーションの代わりに) i5/OS のソケット・インプリメンテーションを使用するため、AIX の動作とは多少異なります。

i5/OS のソケット・インプリメンテーションは、UNIX 98 ソケットとバークレー・ソフトウェア・ディストリビューション (BSD) ソケットの両方をサポートします。ほとんどの場合、i5/OS PASE は AIX インプリメンテーションの動作を取り入れることによって、これらのスタイルの違いを解決します。

加えて、実行中のアプリケーションのユーザー・プロファイルには、ソケット API でレベル・パラメーターを IPPROTO_IP に、 option_value パラメーターを IP_OPTIONS に指定するための *IOSYSCFG 特殊権限がなければなりません。

関連情報

ソケット・プログラミング

バークレー・ソフトウェア・ディストリビューション (BSD) との互換性

UNIX 98 互換性

データベース

i5/OS PASE は DB2 UDB for iSeries コール・レベル・インターフェース (CLI) をサポートしています。AIX および i5/OS 上の DB2 CLI は互いに適切なサブセットではないので、いくつかのインターフェースで多少の違いがあり、あるインプリメンテーションで存在する API が、別のインプリメンテーションでは存在しない場合もあります。

そのため、以下の点を考慮する必要があります。

- AIX 上でコードの生成は行えるが、テストができない。そのため、AIX ではなく、 i5/OS PASE 内の別のプラットフォームでコードをテストしなければならない。
- ヘッダー・ファイル `sqlcli.h` は i5/OS バージョンのものを使ってコンパイルしなければならない。このヘッダー・ファイルの AIX バージョンを使用してコンパイルしたプログラムは、 i5/OS PASE では実行できません。

i5/OS のデフォルトのエンコード・システムが EBCDIC であるのに対して、 AIX は ASCII を基にします。この違いのために、i5/OS データベース (DB2 UDB for iSeries) と i5/OS PASE アプリケーションとの間のデータ変換が必要になる場合があります。

DB2 CLI が i5/OS PASE をインプリメントする際に、 i5/OS PASE システムが提供するライブラリー・ルーチンによって、文字データは自動的に ASCII から拡張 2 進化 10 進交換コード (EBCDIC) に、あるいは EBCDIC から ASCII に変換されます。この変換は、アクセスされるデータのタグ付き CCSID、および i5/OS PASE プログラムが実行されている ASCII CCSID に基づいて行われます。データベースがタグ

付けされる、つまり CCSID 65535 を使用してタグ付けされる場合、自動変換は行われません。これはデータのエンコード形式を識別するため、また必要な変換を実行するために、アプリケーションに残されます。

CCSID の処理

Qp2RunPase() API を使用する場合は、i5/OS PASE CCSID を明示的に指定する必要があります。

i5/OS PASE CCSID の制御は、API プログラム QP2TERM、QP2SHELL、または QP2SHELL2 を呼び出す前に、 ILE で以下の変数を設定することによって行えます。

- PASE_LANG
- QIBM_PASE_CCSID

ILE でこれらの変数の一方またはその両方が省略される場合、 QP2TERM、QP2SHELL、および QP2SHELL2 はデフォルトで、 i5/OS PASE CCSID および i5/OS PASE 環境変数 LANG を、ジョブの言語および CCSID 属性の i5/OS PASE に相当する最適なものを使用して設定します。

libc.a の拡張によって、i5/OS PASE アプリケーションは _SETCCSID() 関数を使用して、実行中のアプリケーションの CCSID を変更することができるようになります。

アプリケーションの CCSID を変更せずに、i5/OS PASE アプリケーションが DB2 CLI 内部変換をオーバーライドできるようにする拡張もあります。 SQLOverrideCCSID400() 関数は、オーバーライド CCSID の整数を、1 つのパラメーターとして受け取ります。

注: オーバーライドを有効にするには、CCSID オーバーライド関数 SQLOverrideCCSID400() を他のすべての SQLx() API の前に呼び出す必要があります。そうしない場合、要求は無視されます。

i5/OS PASE プログラムでの DB2 UDB for iSeries CLI の使用

i5/OS PASE プログラムで DB2 CLI を使用するには、ソースをコンパイルする前に、 sqlcli.h ヘッダー・ファイルと libdb400.exp エクスポート・ファイルを AIX システムにコピーする必要があります。 DB2 CLI ライブラリー・ルーチンは、i5/OS PASE 環境の libdb400.a にあり、 pthread インターフェースを使用してインプリメントされます。これはスレッド・セーフティーを提供します。 i5/OS PASE CLI 関数は多くの場合、対応する ILE CLI 関数を呼び出して、必要な操作を実行します。

注: DB2 CLI を i5/OS PASE プログラムで使用する場合、以下の点を考慮してください。

- SQLGetSubString は CLOB/DBCLOB フィールドをサブストリング化する場合、常に EBCDIC ストリングを戻す。 SQLGetSubString は LOB データ・タイプに対してのみ使用されます。
- 結果セット (テーブル・タイプ) の列 4 である SQLTables は常に EBCDIC として戻される。
- i5/OS PASE プログラムで図形文字データを扱うには、そのデータはプログラム内で wchar として入力されている必要があります。これによってデータベースは図形文字のみの 2 バイト文字を Unicode/UCS-2 に変換します。そうしないと、データベースはデータの CCSID と i5/OS ジョブの CCSID の間で変換を行います。データベースは、EBCDIC 図形文字と CCSID との間の変換 (Qp2RunPase() API または SQLOverrideCCSID400() API のいずれかによる) をサポートしません。

関連資料

18 ページの『ヘッダー・ファイルのコピー』
このトピックの説明に従って、iSeries サーバーから AIX マシンにヘッダー・ファイルをコピーすることができます。

20 ページの『エクスポート・ファイルのコピー』
このトピックの説明に従って、iSeries サーバーから AIX ディレクトリーにエクスポート・ファイルをコピーすることができます。

関連情報

QP2TERM()--i5/OS PASE 端末セッションの実行

QP2SHELL() および QP2SHELL2()--i5/OS PASE シェル・プログラムの実行

_SETCCSID()--i5/OS PASE CCSID の設定

SQLOverrideCCSID400()--i5/OS PASE 用の SQL CLI CCSID のオーバーライド

SQL CLI

例: i5/OS PASE プログラムでの DB2 UDB for iSeries CLI 関数の呼び出し:

このトピックの例は、DB2 UDB for iSeries SQL コール・レベル・インターフェースを使用して DB2 UDB for iSeries にアクセスする i5/OS PASE プログラムを示しています。

注: コード例を使用することにより、 62 ページの『コードに関する特記事項』 の条件に同意することになります。

```
/* i5/OS PASE DB2 UDB for iSeries example program
 *
 * To show an example of an i5/OS PASE program that accesses
 * i5/OS DB2 UDB via SQL CLI
 *
 * Program accesses iSeries Access data base, QIWS/QCUSTCDT, that
 * should exist on all systems
 *
 * Change system name, userid, and password in fun_Connect()
 * procedure to valid parms
 *
 * Compilation invocation:
 *
 * xlc -I./include -bI:./include/libdb400.exp -o paseclidb4 paseclidb4.c
 *
 * FTP in binary, run from QP2TERM() terminal shell
 *
 * Output should show all rows with a STATE column match of MN  */
/* Change Activity: */
/* End Change Activity */

#define SQL_MAX_UID_LENGTH 10
#define SQL_MAX_PWD_LENGTH 10
#define SQL_MAX_STM_LENGTH 255

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "sqlcli.h"

SQLRETURN fun_Connect( void );
SQLRETURN fun_DisConnect( void );
SQLRETURN fun_ReleaseEnvHandle( void );
SQLRETURN fun_ReleaseDbcHandle( void );
SQLRETURN fun_ReleaseStmHandle( void );
SQLRETURN fun_Process( void );
SQLRETURN fun_Process2( void );
void fun_PrintError( SQLHSTMT );

SQLRETURN nml_ReturnCode;
SQLHENV nml_HandleToEnvironment;
SQLHDBC nml_HandleToDatabaseConnection;
SQLHSTMT nml_HandleToSqlStatement;
```

```
SQLINTEGER Nmi_vParam;
SQLINTEGER Nmi_RecordNumberToFetch = 0;
SQLCHAR chs_SqlStatement01[ SQL_MAX_STM_LENGTH + 1 ];
SQLINTEGER nmi_PcbValue;
SQLINTEGER nmi_vParam;
char *pStateName = "MN";

void main( ) {
    static
      char*pszId = "main()";
        SQLRETURN nml_ConnectionStatus;
        SQLRETURN nml_ProcessStatus;

        nml_ConnectionStatus = fun_Connect();
        if ( nml_ConnectionStatus == SQL_SUCCESS ) {
           printf( "%s: fun_Connect() succeeded¥n", pszId );
      } else {
           printf( "%s: fun_Connect() failed¥n", pszId );
           exit( -1 );
      } /* endif */

        printf( "%s: Perform query¥n", pszId );
        nml_ProcessStatus = fun_Process();
        printf( "%s: Query complete¥n", pszId );
        nml_ConnectionStatus = fun_DisConnect();
        if ( nml_ConnectionStatus == SQL_SUCCESS ) {
           printf( "%s: fun_DisConnect() succeeded¥n", pszId );
      } else {
           printf( "%s: fun_DisConnect() failed¥n", pszId );
           exit( -1 );
      } /* endif */

        printf( "%s: normal exit¥n", pszId );
} /* end main */

SQLRETURN fun_Connect()
{
        static char *pszId = "fun_Connect()";
        SQLCHAR chs_As400System[ SQL_MAX_DSN_LENGTH ];
        SQLCHAR chs_UserName[ SQL_MAX_UID_LENGTH ];
        SQLCHAR chs_UserPassword[ SQL_MAX_PWD_LENGTH ];
        nml_ReturnCode = SQLAllocEnv( &nml_HandleToEnvironment );
        if ( nml_ReturnCode != SQL_SUCCESS ) {
           printf( "%s: SQLAllocEnv() succeeded¥n", pszId );
           fun_PrintError( SQL_NULL_HSTMT );
           printf( "%s: Terminating¥n", pszId );
           return SQL_ERROR;
      } else {
           printf( "%s: SQLAllocEnv() succeeded¥n", pszId );
      } /* endif */

      strcpy( chs_As400System, "AS4PASE" );
      strcpy( chs_UserName, "QUSER" );
      strcpy( chs_UserPassword, "QUSER" );
      printf( "%s: Connecting to %s userid %s¥n", pszId, chs_As400System, chs_UserName );

      nml_ReturnCode = SQLAllocConnect( nml_HandleToEnvironment,
                                        &nml_HandleToDatabaseConnection );

      if ( nml_ReturnCode != SQL_SUCCESS ) {
         printf( "%s: SQLAllocConnect¥n", pszId );
         fun_PrintError( SQL_NULL_HSTMT );
         nml_ReturnCode = fun_ReleaseEnvHandle();
         printf( "%s: Terminating¥n", pszId );
         return SQL_ERROR;
    } else {
         printf( "%s: SQLAllocConnect() succeeded¥n", pszId );
```

```
    } /* endif */

    nml_ReturnCode = SQLConnect( nml_HandleToDatabaseConnection,
                                 chs_As400System,
                                 SQL_NTS,
                                 chs_UserName,
                                 SQL_NTS,
                                 chs_UserPassword,
                                 SQL_NTS );
    if ( nml_ReturnCode != SQL_SUCCESS ) {
       printf( "%s: SQLConnect(%s) failed¥n", pszId, chs_As400System );
       fun_PrintError( SQL_NULL_HSTMT );
       nml_ReturnCode = fun_ReleaseDbcHandle();
       nml_ReturnCode = fun_ReleaseEnvHandle();
       printf( "%s: Terminating¥n", pszId );
       return SQL_ERROR;
  } else {
       printf( "%s: SQLConnect(%s) succeeded¥n", pszId, chs_As400System );
       return SQL_SUCCESS;
  } /* endif */
} /* end fun_Connect */

SQLRETURN fun_Process()
{
      static
        char*pszId = "fun_Process()";
      charcLastName[ 80 ];

      nml_ReturnCode = SQLAllocStmt( nml_HandleToDatabaseConnection,
                                     &nml_HandleToSqlStatement );
      if ( nml_ReturnCode != SQL_SUCCESS ) {
         printf( "%s: SQLAllocStmt() failed¥n", pszId );
         fun_PrintError( SQL_NULL_HSTMT );
         printf( "%s: Terminating¥n", pszId );
         return SQL_ERROR;
    } else {
         printf( "%s: SQLAllocStmt() succeeded¥n", pszId );
    } /* endif */

    strcpy( chs_SqlStatement01, "select LSTNAM, STATE " );
    strcat( chs_SqlStatement01, "from QIWS.QCUSTCDT " );
    strcat( chs_SqlStatement01, "where " );
    strcat( chs_SqlStatement01, "STATE = ? " );

    nml_ReturnCode = SQLPrepare( nml_HandleToSqlStatement,
                                 chs_SqlStatement01,
                                 SQL_NTS );
    if ( nml_ReturnCode != SQL_SUCCESS ) {
       printf( "%s: SQLPrepare() failed¥n", pszId );
       fun_PrintError( nml_HandleToSqlStatement );
       nml_ReturnCode = fun_ReleaseStmHandle();
       printf( "%s: Terminating¥n", pszId );
       return SQL_ERROR;
  } else {
       printf( "%s: SQLPrepare() succeeded¥n", pszId );
  } /* endif */

    Nmi_vParam = SQL_TRUE;
    nml_ReturnCode = SQLSetStmtOption( nml_HandleToSqlStatement,
                                       SQL_ATTR_CURSOR_SCROLLABLE,
                                       ( SQLINTEGER * ) &Nmi_vParam );
    if ( nml_ReturnCode != SQL_SUCCESS ) {
       printf( "%s: SQLSetStmtOption() failed¥n", pszId );
       fun_PrintError( nml_HandleToSqlStatement );
       nml_ReturnCode = fun_ReleaseStmHandle();
       printf( "%s: Terminating¥n", pszId );
       return SQL_ERROR;
```

```
} else {
    printf( "%s: SQLSetStmtOption() succeeded¥n", pszId );
} /* endif */

  Nmi_vParam = SQL_TRUE;
  nml_ReturnCode = SQLSetStmtOption( nml_HandleToSqlStatement,
                                     SQL_ATTR_FOR_FETCH_ONLY,
                                     ( SQLINTEGER * ) &Nmi_vParam );
  if ( nml_ReturnCode != SQL_SUCCESS ) {
     printf( "%s: SQLSetStmtOption() failed¥n", pszId );
     fun_PrintError( nml_HandleToSqlStatement );
     nml_ReturnCode = fun_ReleaseStmHandle();
     printf( "%s: Terminating¥n", pszId );
     return SQL_ERROR;
} else {
     printf( "%s: SQLSetStmtOption() succeeded¥n", pszId );
} /* endif */

  nmi_PcbValue = 0;
  nml_ReturnCode = SQLBindParam( nml_HandleToSqlStatement,
                                           1,
                                           SQL_CHAR,
                                           SQL_CHAR,
                                           2,
                                           0,
                                           ( SQLPOINTER ) pStateName,
                                           ( SQLINTEGER *) &nmi_PcbValue );
  if ( nml_ReturnCode != SQL_SUCCESS ) {
     printf( "%s: SQLBindParam() failed¥n", pszId );
     fun_PrintError( nml_HandleToSqlStatement );
     nml_ReturnCode = fun_ReleaseStmHandle();
     printf( "%s: Terminating¥n", pszId );
     return SQL_ERROR;
} else {
     printf( "%s: SQLBindParam() succeeded¥n", pszId );
} /* endif */

  nml_ReturnCode = SQLExecute( nml_HandleToSqlStatement );
  if ( nml_ReturnCode != SQL_SUCCESS ) {
     printf( "%s: SQLExecute() failed¥n", pszId );
     fun_PrintError( nml_HandleToSqlStatement );
     nml_ReturnCode = fun_ReleaseStmHandle();
     printf( "%s: Terminating¥n", pszId );
     return SQL_ERROR;
} else {
     printf( "%s: SQLExecute() succeeded¥n", pszId );
} /* endif */

  nml_ReturnCode = SQLBindCol( nml_HandleToSqlStatement,
                               1,
                               SQL_CHAR,
                               ( SQLPOINTER ) &cLastName,
                               ( SQLINTEGER ) ( 8 ),
                               ( SQLINTEGER * ) &nmi_PcbValue );
  if ( nml_ReturnCode != SQL_SUCCESS ) {
     printf( "%s: SQLBindCol() failed¥n", pszId );
     fun_PrintError( nml_HandleToSqlStatement );
     nml_ReturnCode = fun_ReleaseStmHandle();
     printf( "%s: Terminating¥n", pszId );
     return SQL_ERROR;
} else {
     printf( "%s: SQLBindCol() succeeded¥n", pszId );
} /* endif */

  do {
        memset( cLastName, '¥0', sizeof( cLastName ) );
        nml_ReturnCode = SQLFetchScroll( nml_HandleToSqlStatement,
```

```
                                        SQL_FETCH_NEXT,
                                        Nmi_RecordNumberToFetch );
          if ( nml_ReturnCode == SQL_SUCCESS ) {
             printf( "%s: SQLFetchScroll() succeeded, LastName(%s)¥n", pszId, cLastName);
        } else {
        }/*endif */
    } while ( nml_ReturnCode == SQL_SUCCESS );
    if ( nml_ReturnCode != SQL_NO_DATA_FOUND ) {
       printf( "%s: SQLFetchScroll() failed¥n", pszId );
       fun_PrintError( nml_HandleToSqlStatement );
       nml_ReturnCode = fun_ReleaseStmHandle();
       printf( "%s: Terminating¥n", pszId );
       return SQL_ERROR;
  } else {
       printf( "%s: SQLFetchScroll() completed all rows¥n", pszId );
  } /* endif */

    nml_ReturnCode = SQLCloseCursor( nml_HandleToSqlStatement );
    if ( nml_ReturnCode != SQL_SUCCESS ) {
       printf( "%s: SQLCloseCursor() failed¥n", pszId );
       fun_PrintError( nml_HandleToSqlStatement );
       nml_ReturnCode = fun_ReleaseStmHandle();
       printf( "%s: Terminating¥n", pszId );
       return SQL_ERROR;
  } else {
       printf( "%s: SQLCloseCursor() succeeded¥n", pszId );
  } /* endif */

    return SQL_SUCCESS;
} /* end fun_Process */

SQLRETURN fun_DisConnect()
{
     static
       char*pszId = "fun_DisConnect()";

     nml_ReturnCode = SQLDisconnect( nml_HandleToDatabaseConnection );
     if ( nml_ReturnCode != SQL_SUCCESS ) {
        printf( "%s: SQLDisconnect() failed¥n", pszId );
        fun_PrintError( SQL_NULL_HSTMT );
        printf( "%s: Terminating¥n", pszId );
        return 1;
   } else {
        printf( "%s: SQLDisconnect() succeeded¥n", pszId );
   } /* endif */

     nml_ReturnCode = fun_ReleaseDbcHandle();
     nml_ReturnCode = fun_ReleaseEnvHandle();

     return nml_ReturnCode;
} /* end fun_DisConnect */

SQLRETURN fun_ReleaseEnvHandle()
{
     static
       char*pszId = "fun_ReleaseEnvHandle()";

     nml_ReturnCode = SQLFreeEnv( nml_HandleToEnvironment );
     if ( nml_ReturnCode != SQL_SUCCESS ) {
        printf( "%s: SQLFreeEnv() failed¥n", pszId );
        fun_PrintError( SQL_NULL_HSTMT );
        return SQL_ERROR;
   } else {
        printf( "%s: SQLFreeEnv() succeeded¥n", pszId );
        return SQL_SUCCESS;
   } /* endif */
} /* end fun_ReleaseEnvHandle */
```

```
SQLRETURN fun_ReleaseDbcHandle()
{
     static
       char*pszId = "fun_ReleaseDbcHandle()";

     nml_ReturnCode = SQLFreeConnect( nml_HandleToDatabaseConnection );
     if ( nml_ReturnCode != SQL_SUCCESS ) {
        printf( "%s: SQLFreeConnect() failed¥n", pszId );
        fun_PrintError( SQL_NULL_HSTMT );
        return SQL_ERROR;
   } else {
        printf( "%s: SQLFreeConnect() succeeded¥n", pszId );
        return SQL_SUCCESS;
   } /* endif */
} /* end fun_ReleaseDbcHandle */

SQLRETURN fun_ReleaseStmHandle()
{
     static
       char*pszId = "fun_ReleaseStmHandle()";

     nml_ReturnCode = SQLFreeStmt( nml_HandleToSqlStatement, SQL_CLOSE );
     if ( nml_ReturnCode != SQL_SUCCESS ) {
        printf( "%s: SQLFreeStmt() failed¥n", pszId );
        fun_PrintError( nml_HandleToSqlStatement );
        return SQL_ERROR;
   } else {
        printf( "%s: SQLFreeStmt() succeeded¥n", pszId );
        return SQL_SUCCESS;
   } /* endif */
} /* end fun_ReleaseStmHandle */

void fun_PrintError( SQLHSTMT nml_HandleToSqlStatement )
{
     static
       char*pszId = "fun_PrintError()";

     SQLCHAR chs_SqlState[ SQL_SQLSTATE_SIZE ];
     SQLINTEGER nmi_NativeErrorCode;
     SQLCHAR chs_ErrorMessageText[ SQL_MAX_MESSAGE_LENGTH + 1 ];
     SQLSMALLINT nmi_NumberOfBytes;

     nml_ReturnCode = SQLError( nml_HandleToEnvironment,
                                nml_HandleToDatabaseConnection,
                                nml_HandleToSqlStatement,
                                chs_SqlState,
                                &nmi_NativeErrorCode,
                                chs_ErrorMessageText,
                                sizeof( chs_ErrorMessageText ),
                                &nmi_NumberOfBytes );

     if ( nml_ReturnCode != SQL_SUCCESS ) {
        printf( "%s: SQLError() failed¥n", pszId );
        return;
   } /* endif */

     printf( "%s: SqlState - %s¥n", pszId, chs_SqlState );
     printf( "%s: SqlCode - %d¥n", pszId, nmi_NativeErrorCode );
     printf( "%s: Error Message:¥n", pszId );
     printf( "%s: %s¥n", pszId, chs_ErrorMessageText );
} /* end fun_PrintError */
```

データ・エンコード

AIX や Linux など、ほとんどのオペレーティング・システムでは、ASCII 文字エンコードが使用されます。ほとんどの i5/OS 機能では、EBCDIC 文字エンコードが使用されます。いくつかの i5/OS オブジェク

ト・タイプに対してコード化文字セット ID (CCSID) の値を指定することにより、オブジェクト内の文字データに関する特定のエンコードを指定することができます。

i5/OS PASE バイト・ストリーム・ファイルには、CCSID 属性があります。 i5/OS PASE 外にあるほとんどのシステム・インターフェースはこの属性を使用して、ファイルから読み取られるテキスト・データ、およびファイルに書き込まれるテキスト・データを必要に応じて変換します。i5/OS PASE はストリーム・ファイルから読み取られるデータ、およびストリーム・ファイルに書き込まれるデータの CCSID 変換を行いませんが (AIX と整合性があります)、 i5/OS PASE プログラムによって作成されたバイト・ストリーム・ファイルの CCSID 属性を、現行の i5/OS PASE CCSID 値に設定して、その他のシステムの関数がファイル内の ASCII テキストを適切に処理できるようにします。

i5/OS PASE 共用ライブラリーに付属する AIX API を使用する場合、 i5/OS PASE はほとんどのデータ変換を処理します。i5/OS PASE プログラムは、i5/OS PASE ランタイムによって自動的に処理されないすべての文字データ変換について、共用ライブラリー `libiconv.a` で提供されている `iconv` 関数を使用することができます。たとえば、 i5/OS PASE アプリケーションは一般に、(_ILECALLX または _PGMCALL のいずれかを使用して) i5/OS API 関数を呼び出す前に、文字ストリングを EBCDIC に変換する必要があります。

関連概念

『ファイル・システム』
i5/OS PASE プログラムは、統合ファイル・システムを介してアクセス可能なすべてのファイルとリソースにアクセスできます。これには、QSYS.LIB および QOPT ファイル・システム内のオブジェクトも含まれます。

ファイル・システム

i5/OS PASE プログラムは、統合ファイル・システムを介してアクセス可能なすべてのファイルとリソースにアクセスできます。これには、QSYS.LIB および QOPT ファイル・システム内のオブジェクトも含まれます。

バッファーに入れられる入出力

外部装置との入出力は、i5/OS 上でバッファーに入れられます。これはデータ・ブロックを扱う入出力プロセッサーによって処理されます。これとは逆に、AIX や Linux などのオペレーティング・システムは通常、文字単位の (バッファーに入れられない) 入出力を操作します。 i5/OS では、特定の入出力シグナル (例: Enter キー、ファンクション・キー、システム要求) が、システムに割り込みを送信します。

データ変換サポート

i5/OS PASE プログラムは ASCII (または UTF-8) パス名を open() 関数に渡して、バイト・ストリーム・ファイルをオープンします。この場合、名前は i5/OS によって使用されるエンコード・スキーマに自動的に変換されますが、オープン・ファイルから読み書きされるデータは変換されません。

ファイル記述子の使用

i5/OS PASE ランタイムは通常、ファイル stdin、stdout、および stderr 用の ILE C ランタイム・サポートを使用します。これらのファイルは、i5/OS PASE および ILE プログラムに対して、一貫した動作を提供します。

i5/OS PASE および ILE C は、標準入出力 (stdin、stdout、および stderr) について同じストリームを使用します。i5/OS PASE は常に、ファイル記述子 0、1、および 2 を使用して、標準入出力にアクセスします。ただし、ILE C は常に stdin、stdout、および stderr の統合ファイル記述子を使用するとは限らないの

で、 i5/OS PASE は i5/OS PASE ファイル記述子と統合ファイル・システム内の記述子との間のマッピングを提供します。このマッピングにより、i5/OS PASE プログラムと ILE C プログラムは異なる記述子番号を使用して、同じオープン・ファイルにアクセスすることができます。

fcntl 関数の i5/OS PASE 拡張版である F_MAP_XPFFD を使用して、 i5/OS PASE 記述子を ILE 番号に割り当てることができます。これは、 i5/OS PASE によって作成されなかった ILE 記述子のファイル操作を i5/OS PASE アプリケーションが行わなければならない場合に役立ちます。

fstatx() 関数の i5/OS システム固有の拡張版である STX_XPFFD_PASE を使用すると、 i5/OS PASE プログラムは i5/OS PASE ファイル記述子用の統合ファイル・システム記述子番号を決定することができます。ファイル stdin、stdout、および stderr 用の ILE C ランタイム・サポートに付加されたすべての i5/OS PASE 記述子には特殊値 (負の数) が戻されます。

Qp2RunPase() API が呼び出されるときに ILE 環境変数 QIBM_USE_DESCRIPTOR_STDIO が Y または I に設定されている場合、 i5/OS PASE はファイル記述子 0、1、および 2 と統合ファイル・システムとの同期をとり、 i5/OS PASE プログラムと ILE C プログラムの両方で、ファイル stdin、stdout、および stderr に同じ記述子番号が使用されるようにします。このモードの作動中に i5/OS PASE コードまたは ILE C コードがファイル記述子 0、1、および 2 をクローズまたは再オープンする場合、その変更はどちらの環境の stdin、stdout、および stderr の処理にも影響を与えます。

i5/OS PASE ランタイムは一般に、i5/OS PASE ファイル記述子 (ソケットを含む) によって読み書きされるデータの文字エンコード変換を行いません。ただし、ILE C stdin から読み取られるデータ、または ILE C stdout および stderr に書き込まれるデータについて、 (i5/OS PASE CCSID とジョブ・デフォルト CCSID の間で) ASCII から EBCDIC への変換は行われます。

stdin、stdout、および stderr の自動変換は、 2 つの環境変数によって制御されます。

- 一般に適用される変数は、QIBM_USE_DESCRIPTOR_STDIO です。この変数を Y に設定すると、 ILE ランタイムはこれらのファイルに対してファイル記述子 0、1、または 2 を使用します。
- i5/OS PASE システム固有の環境変数は、QIBM_PASE_DESCRIPTOR_STDIO です。バイナリーの場合は B、テキストの場合は T の値が入ります。

i5/OS PASE stdin、stdout、および stderr の ASCII から EBCDIC への変換は、 ILE 環境変数 QIBM_USE_DESCRIPTOR_STDIO を Y に、 QIBM_PASE_DESCRIPTOR_STDIO を B に設定すると、使用不可になります (バイナリー・データは stdin から読み取られ、 stdout または stderr に書き込まれます)。 QIBM_PASE_DESCRIPTOR_STDIO のデフォルトは T (テキスト) です。この値を設定すると、EBCDIC から ASCII への変換が行われます。

関連概念

51 ページの『データ・エンコード』
AIX や Linux など、ほとんどのオペレーティング・システムでは、ASCII 文字エンコードが使用されます。ほとんどの i5/OS 機能では、EBCDIC 文字エンコードが使用されます。いくつかの i5/OS オブジェクト・タイプに対してコード化文字セット ID (CCSID) の値を指定することにより、オブジェクト内の文字データに関する特定のエンコードを指定することができます。

関連情報

統合ファイル・システム

グローバリゼーション

i5/OS PASE は AIX のランタイムをベースにしているため、 i5/OS PASE プログラムでは、AIX でサポートされている、ロケール、文字ストリング処理、日時サービス、メッセージ・カタログ、および文字エンコード変換といった、数多くの一連のプログラミング・インターフェースを使用することができます。

i5/OS PASE は、アプリケーションで使用するロケールの管理や、ロケールを区別する関数 (ctype や strcoll など) の実行において、 AIX ランタイムのインターフェースをサポートしています。これには、単一バイトとマルチバイト両方の文字エンコード方式のサポートも含まれます。

i5/OS PASE には AIX ロケールのサブセットが組み込まれており、これによって、業界標準のエンコード方式 (コード・セット ISO8859-x)、コード・セット IBM-1250、およびコード・セット UTF-8 を使用した、多くの国や言語のサポートが提供されます。 i5/OS PASE は、IBM-1252 ロケールと ISO 8859-15 ロケール (これらはいずれも単一バイトのエンコード方式を使用)、および UTF-8 ロケールという 3 つの異なる方法でユーロをサポートしています。

注: i5/OS PASE のロケール・サポートは、 ILE C プログラムで使用されるロケール・サポート (オブジェクト・タイプ *CLD および *LOCALE) のいずれの形式にも属しません。内部構造が異なる上、ILE C プログラム用に同梱されている既存のロケールに、ASCII をサポートするものはありません。

新規ロケールの作成

i5/OS PASE には、新規ロケールを作成するためのユーティリティーは同梱されていません。ただし、`localedef` ユーティリティーを使用すれば、 AIX システム上の i5/OS PASE で使用するロケールを作成することは可能です。

ロケールの変更

i5/OS PASE アプリケーションでロケールを変更する場合は、一般的には、新しいロケールのエンコード方式と一致させるために、i5/OS PASE CCSID も (_SETCCSID ランタイム機能を使用して) 変更するべきです。こうすることにより、すべての文字データ・インターフェース引数が、 i5/OS PASE ランタイムによって正しく解釈されるようになります (また、場合によっては、EBCDIC システム・サービスの呼び出し時に変換されます)。 CCSID がどのコード・セット名と対応するかを確認するには、 `cstoccsid` ランタイム機能を使用できます。

i5/OS PASE ランタイムは、 i5/OS PASE プログラムによって作成されるすべてのファイルの CCSID タグを、現行の i5/OS PASE CCSID 値 (プログラムの開始時や最新の _SETCCSID 値を使用したときに得られる) に設定します。

日本語、韓国語、繁体字の中国語、および簡体字の中国語をサポートする i5/OS PASE には、UTF-8 ロケールを使用してください。i5/OS には、これらの言語をサポートする他のロケールもありますが、システムは、i5/OS PASE CCSID を IBM-euc*XX* コード・セットのエンコードと対応させる設定をサポートしていません。アプリケーションが他のプラットフォームで実行されるときは、ファイル・データが他のエンコード方式 (Shift-JIS など) で保管されている可能性もあるため、 UTF-8 サポートを使用するには、そのようなファイル・データの変換が必要になることがあります。

i5/OS PASE 変換オブジェクトとロケールの保管場所

i5/OS PASE の変換オブジェクトとロケールは、 i5/OS 言語フィーチャー・コードと一緒にパッケージされています。ロケールは、i5/OS PASE がインストールされるときに、インストールされる i5/OS 言語フィーチャーに関連付けられているものだけが作成されます。

すべての i5/OS PASE ロケールでは、ASCII または UTF-8 の文字エンコード方式が使用されます。したがって、すべての i5/OS PASE ランタイムは、ASCII (または UTF-8) で動作します。

関連タスク

5 ページの『i5/OS PASE のインストール』
このトピックの説明に従って、ご使用のサーバーに i5/OS PASE をインストールすることができます。

関連情報

i5/OS グローバリゼーション

i5/OS PASE ロケール

_SETCCSID()--i5/OS PASE CCSID の設定

メッセージ・サービス

i5/OS PASE のシグナルと ILE のシグナルは独立しているため、一方のタイプのシグナルを出してもう一方のシグナル・タイプのハンドラーを直接呼び出すことはできません。

受け取った任意の ILE シグナルに対応する i5/OS PASE シグナルを POST するには、 i5/OS PASE Qp2SignalPase() API を使用できます。 QP2SHELL() プログラムと i5/OS PASE `fork()` 関数は、すべての ILE シグナルを対応する i5/OS PASE シグナルにマップするハンドラーを必ずセットアップします。

システムは、Qp2RunPase、Qp2CallPase、または Qp2CallPase2 API を実行する呼び出しのプログラム・メッセージ・キューに送信されるすべての i5/OS 例外メッセージを、自動的に、対応する i5/OS PASE シグナルに変換します。このようにして、i5/OS PASE アプリケーションは、システムによって変換された i5/OS PASE シグナルを処理することによって、すべての i5/OS 例外を処理することができます。

i5/OS PASE には、 i5/OS のメッセージ処理を直接制御できるようにする、次のようなランタイム機能があります。

- QMHSNDM
- QMHSNDM1
- QMHSNDPM
- QMHSNDPM1
- QMHSNDPM2
- QMHRCVM
- QMHRCVM1
- QMHRCVPM
- QMHRCVPM1
- QMHRCVPM2

これらの機能に関する詳細は、各ランタイム機能の説明を参照してください。

i5/OS メッセージ・サポート

i5/OS には、さまざまなコンテキストでのメッセージ・サポートがあります。

- **ジョブ・ログ**。ジョブ・ログには、i5/OS やアプリケーションが実行されたりコンパイルされたりしたときに発行された、すべてのメッセージが記録されています。ジョブ・ログを表示するには、コマンド行から `DSPJOBLOG` と入力してください。「ジョブ・ログの表示」表示画面が表示されたら、F10 を押し、続いて Shift + F6 を押します。これらのキーの組み合わせを押すと、「すべてのメッセージ表示」表示

画面が表示され、最新のメッセージに設定されます。特定のメッセージの詳細情報を表示するには、詳細を知りたいメッセージの上にカーソルを持っていき、F1 を押します。

- **アクティブ・ジョブの処理。** i5/OS 上のジョブやジョブ・スタックについて調べるには、アクティブ・ジョブの処理 (WRKACTJOB) コマンドが役立ちます。

関連情報

Qp2SignalPase()--i5/OS PASE シグナルの POST

i5/OS PASE で使用するためのランタイム機能

活動ジョブ処理 (WRKACTJOB)

実行管理機能

i5/OS PASE シグナルの処理

i5/OS PASE アプリケーションからの印刷出力

i5/OS PASE シェルからの出力の読み取りおよび書き込みを行うには、 QShell Rfile ユーティリティーを使用できます。

次の例では、ストリーム・ファイル mydoc.ps の内容を、スプールされているプリンター・ファイル QPRINT に未変換の ASCII データとして書き込み、 CL LPR コマンドを使用してそのスプールされたファイルを別のシステムに送信します。

```
before='ovrprtf qprint devtype(*userascii) spool(*yes)'¥
after="lpr file(qprint) system(usrchprt01) prtq('rchdps') transform(*no)"
cat -c mydoc.ps | Rfile -wbQ -c "$before" -C "$after" qprint
```

関連情報

Rfile - レコード・ファイルの読み書きをする

疑似端末

i5/OS PASE は、AT&T とバークレー・ソフトウェア・ディストリビューション (BSD) の両方のスタイルのデバイスをサポートしています。プログラミングの観点からすれば、これらのデバイスは、AIX 上で機能するのと同じように i5/OS PASE 上で機能します。

i5/OS PASE では、AT&T スタイルのデバイスに最大 1024 のインスタンス、 BSD スタイルのデバイスに最大 592 のインスタンスを持つことができます。システムが開始されると、最初の 32 のインスタンスが、各デバイス・タイプに自動的に作成されます。

i5/OS PASE での PTY デバイスの構成

AIX において、管理者は、smit を使用して使用可能な各タイプのデバイスの数を構成します。一方 i5/OS PASE では、これらのデバイスは次のような方法で構成されます。

- AT&T-style スタイルのデバイスの場合、i5/OS PASE は自動構成をサポートしています。最初の 32 のインスタンスが使用されている状態でアプリケーションが別のインスタンスを開こうとすると、 1024 のデバイスを限度として、統合ファイル・システム内に自動的に CHRSF デバイスが作成されます。
- BSD-style スタイルのデバイスの場合は、i5/OS PASE mknod ユーティリティーを使用して、手動で CHRSF デバイスを作成する必要があります。これを行うためには、BSD 従属デバイスと BSD 基本デバイスの主要な番号と、命名規則を知っている必要があります。次の例は、追加の BSD 疑似端末 (PTY) デバイスを作成する方法を示す、シェル・スクリプトです。この例では、グループ 16 にデバイスを作成します。

注: コード例を使用することにより、 62 ページの『コードに関する特記事項』 の条件に同意することになります。

```
#!/QOpenSys/usr/bin/ksh

prefix="pqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
bsd_tty_major=32949
bsd_pty_major=32948

if [ $# -lt 1 ]
then
    echo "usage: $(basename $0) ptyN "
    exit 10
fi

function mkdev {
  if [ ! -e $1 ]
  then
    mknod $1 c $2 $3
    chown QSYS $1
    chmod 0666 $1
  fi
}

while [ "$1" ]
do
  N=${1##pty}
  if [ "$N" = "$1" -o "$N" = "" -o $N -lt 0 -o $N -gt 36 ]
  then
      echo "skipping: ¥"$1¥": not valid, must be in the form ptyN where: 0 <= N <= 36"
      shift
      continue
  fi

  minor=$((N * 16))
  pre=$(expr "$prefix" : ".¥{$N¥}¥(.¥)")

  echo "creating /dev/[pt]ty${pre}0 - /dev/[pt]ty${pre}f"
  for i in 0 1 2 3 4 5 6 7 8 9 a b c d e f
  do
    echo ".¥c"
    mkdev /dev/pty${pre}${i} $bsd_pty_major $minor
    echo ".¥c"
    mkdev /dev/tty${pre}${i} $bsd_tty_major $minor
    minor=$((minor + 1))
  done
  echo ""

  shift
done
```

PTY デバイスについての詳細は、AIX 文書 Web サイト (英語) を参照してください。

セキュリティー

セキュリティーの視点から、 i5/OS PASE プログラムは、i5/OS 上の他のすべてのプログラムと同じセキュリティー制限に属しています。

i5/OS で i5/OS PASE を実行するためには、統合ファイル・システム内の AIX バイナリーに対する権限が必要です。また、プログラムがアクセスするそれぞれのリソースに適したレベルの権限を持っていることも必要です。適切なレベルの権限がなければ、それらのリソースにアクセスしようとすると、プログラムはエラーを戻します。

i5/OS PASE プログラムを実行する場合には、次の情報が特に重要です。

ユーザー・プロファイルと権限管理

システムの権限管理は、オブジェクトの 1 つであるユーザー・プロファイルに基づいています。システム上で作成されるすべてのオブジェクトは、特定のユーザーによって所有されます。オブジェクトに対するそれぞれの操作やアクセスは、ユーザーの権限を確認するために、システムによって検証されます。所有者や適切な権限のあるユーザー・プロファイルは、他のユーザー・プロファイルに、オブジェクトを操作するさまざまなタイプの権限を委任することができます。権限検査は、すべてのタイプのオブジェクトに対して一様に行われます。

オブジェクトの権限のメカニズムによって、さまざまなレベルの制御が備えられます。ユーザーの権限は、必要なものだけに制限できます。 QOpenSys ファイル・システムに保管されるファイルには、 UNIX ファイルと同じ方法で権限を付与できます。次の表は、UNIX の許可と、i5/OS データベース・ファイルで使用されるセキュリティー値の関係を示すものです。 i5/OS において、*OBJOPR は *オブジェクトの使用*権限を表し、*EXCLUDE は*権限なし* を表します。 *READ、*ADD、*UPD、*DLT、および *EXECUTE はデータ権限です。ファイルを i5/OS PASE プログラムとして実行するには、そのファイルに対する *EXECUTE 権限 (および場合によっては *READ 権限) が必要です。

UNIX の許可	*OBJOPR	*READ	*ADD	*UPD	*DLT	*EXECUTE
r (読み取り)	X	X	-	-	-	-
w (書き込み)	X	-	X	X	X	-
x (実行)	X	-	-	-	-	X
権限なし	-	-	-	-	-	-

i5/OS PASE のユーザー・プロファイル

i5/OS では、認証情報は、/etc/passwd のようなファイルにではなく、個々の*プロファイル*に保管されます。ユーザーやグループにはプロファイルがあります。これらのプロファイルはすべて 1 つのネーム・スペースを共用しており、それぞれのプロファイルには、大文字のみを使用した固有な名前がなければなりません。 getpwnam() や getgrnam() といった API に小文字の名前が渡された場合、システムは名前のストリングを大文字に変換します。

getpwuid() や getgrgid() を呼び出してプロファイル名を戻させるときは、 i5/OS PASE 環境変数が結果を大文字で戻す PASE_USRGRP_LOWERCASE=N に設定されていないと、結果が小文字になります。

すべてのユーザーには、ユーザー ID (UID) があります。また、すべてのグループには、グループ ID (GID) があります。これらは Portable Operation System Interface X (POSIX) 1003.1 規格に従って定義されます。この 2 つの数値スペースは分かれているので、ユーザーの UID を 104 として、グループの GID を 104 としても、それらの ID はそれぞれ区別されます。

i5/OS には、機密保護担当者 QSECOFR 用のユーザー・プロファイルがあります。このユーザーの UID は 0 です。他のどのプロファイルも UID を 0 にすることはできません。 QSECOFR は、システム上で最高の特権を持つプロファイルであり、その意味では root ユーザーとして動作します。ただし、i5/OS には、システム管理者によって個々のユーザーに割り当てることのできる、一連の特殊特権があります。このような特権の 1 つは *ALLOBJ で、この特権は、ファイル・アクセスに対する任意アクセス制御をオーバーライドします (たとえば、AIX や Linux などのオペレーティング・システムでは、これを root 特権として使用するのが一般的です)。

root アクセスを使用する移植アプリケーションでは、 *ALLOBJ 権限を付与するアプリケーション・ユーザー 用に特定のユーザー・プロファイルを作成して、QSECOFR を使用させない方が、セキュリティーの

点では有利でしょう。 QSECOFR には、単一のアプリケーションで必要とされる以上の特権があるからです。 AIX または Linux などのオペレーション・システムとは異なり、i5/OS では、ユーザーのグループ・メンバーシップは必要ありません。 i5/OS では、ユーザー・プロファイルの GID が 0 であることは、より多くの特権があるグループを表すのではなく、*グループの割り当てがない* ことを意味します。

i5/OS のセキュリティーは、システム内に構築される統合セキュリティーに依存しています。オブジェクトへのすべてのアクセスは、セキュリティー検査に通る必要があります。セキュリティー検査は、アクセス時にプロセスを実行するユーザー・プロファイルに関して実行されます。

i5/OS PASE は、保全性とセキュリティーの保守を、各プロセスに別個のアドレス・スペースを与えることに依存しています。 i5/OS PASE アドレス・スペースでリソースが使用できない場合は、そのリソースにはアクセスできません。ファイル・システムのセキュリティーは、適切な権限なしにユーザーがそのアドレス・スペースにリソースをロードするのを防ぎます。リソースは、アドレス・スペースに入ると、プロセスが実行される ID であるかどうかに関係なく処理できるようになります。

i5/OS PASE プログラムは、システム関数の要求にシステム呼び出しを使用します。 i5/OS PASE プログラムに対するシステム呼び出しは、i5/OS によって処理されます。このインターフェースでは、i5/OS PASE プログラムは、間接的 (かつ安全) な方法でしかシステム内部にアクセスできません。

関連情報

セキュリティー

実行管理機能

i5/OS は、システム上の他のジョブを処理するのと同じ方法で i5/OS PASE プログラムを処理します。

関連情報

実行管理機能

---

i5/OS PASE プログラムのデバッグ

i5/OS PASE ランタイム環境は、syslog() ランタイム機能、および (より複雑なメッセージ・ルーティングのための) `syslogd` バイナリーのライブラリー・サポートを提供しています。加えて、i5/OS の既存の機能、たとえば診断メッセージ用のジョブ・ログや、 i5/OS システム・オペレーター・メッセージ・キュー QSYSOPR への重大メッセージの送信などを使用することができます。

アプリケーションによっては、i5/OS PASE アプリケーションをデバッグするストラテジーで、異なるパスを使用することができます。

- アプリケーションが i5/OS の統合 (たとえば、 DB2 UDB for iSeries または ILE 機能との統合) を必要としない場合、まず AIX 上でアプリケーションをデバッグする必要があります。
- その後、i5/OS PASE `dbx` と i5/OS デバッグ機能 (ジョブ・ログなど) を組み合わせて使用して、 i5/OS 上でアプリケーションをデバッグします。

データベースまたは ILE 関数を使用するようにコーディングしたアプリケーションを AIX 上で完全にテストすることはできませんが、 AIX 上でアプリケーションの残りの部分をデバッグすることにより、構造と設計が適切なものとなるようにすることができます。

i5/OS PASE での dbx の使用

i5/OS PASE は、AIX `dbx` デバッガー・ユーティリティーをサポートします。このユーティリティーを使用すると、親プロセスや子プロセスなどの関連プロセスを、ソース・コード・レベルでデバッグすることが

できます (そのようにコンパイルされた場合)。 i5/OS PASE で実行されるデバッガーに AIX ソースが見えるようにするために、ネットワーク・ファイル・システム (NFS) を使用することができます。

xterm および aixterm 用の i5/OS PASE サポートにより、 dbx を使用して親プロセスと子プロセスの両方をデバッグすることができます。 dbx は、dbx を 2 番目のプロセスに付加した別の xterm ウィンドウを立ち上げます。

dbx の詳細は、AIX 文書 Web サイト (英語) を参照してください。 dbx コマンド行で help と入力することもできます。

i5/OS デバッグ・ツールの使用

i5/OS PASE アプリケーションをデバッグするために、 i5/OS では以下のツールを使用することができます。

- iSeries System Debugger は、i5/OS PASE アプリケーションのデバッグ用の特定のサポートを提供します。
- ILE C ソース・デバッガーは、コードにおける問題を判別する上で効果的なツールです。

  関連情報

  iSeries システム・デバッガー

  WebSphere Development Studio ILE C/C++ Programmer's Guide PDF

パフォーマンスの最適化

最適なパフォーマンスを得るため、アプリケーション・バイナリーは、ローカル・ストリーム・ファイル・システムに保管するようにしてください。

バイナリー (基本プログラムとライブラリー) がローカル・ストリーム・ファイル・システムの外部にあると、ファイル・マッピングが行えないため、 i5/OS PASE プログラムの開始がかなり遅くなります。

多くの fork() 操作を実行する i5/OS PASE でアプリケーションを実行すると、 AIX で実行するときに比べて速度が落ちます。これは、それぞれの i5/OS PASE fork() 操作ごとに新しい i5/OS ジョブが開始されることで、パフォーマンスがかなりの影響を受ける可能性があるためです。

パフォーマンス・データの収集と分析については、「システム管理」のカテゴリーにある『パフォーマンス』のトピックを参照してください。

例

以下の例が i5/OS PASE 情報として提供されています。

注: コード例を使用することにより、 62 ページの『コードに関する特記事項』 の条件に同意することになります。

ILE プログラムからのi5/OS PASE プログラムおよびプロシージャーの実行

- ILE プログラムからの i5/OS PASE プログラムの実行
- ILE プログラムからの i5/OS PASE プロシージャーの呼び出し

i5/OS PASE プログラムからの i5/OS プログラムの呼び出し

- i5/OS PASE プログラムからの ILE プロシージャーの呼び出し
- i5/OS PASE からの i5/OS プログラムの呼び出し
- i5/OS PASE からの CL コマンドの実行

i5/OS PASE プログラムでの DB2 UDB for iSeries 関数の使用

- i5/OS PASE プログラムでの DB2 UDB for iSeries コール・レベル・インターフェースの呼び出し

---

i5/OS PASE の関連情報

i5/OS PASE のトピックに関連した Information Center のトピックおよび Web サイトについては、このトピックをお読みください。

IBM Redbooks™ および Redpapers

iSeries サーバーへの PHP の導入 (英語) この Redpaper で説明されているステップバイステップのインプリメンテーションには、 i5/OS Portable Application Solutions Environment (i5/OS PASE) で実行できる Hypertext Preprocessor (PHP) の CGI 版が含まれています。

Web サイト

- ロードマップおよびリソースの使用可能化 (英語)
  (http://www.ibm.com/servers/enable/site/porting/index.html)

  この Web サイトでは、i5/OS PASE と、ご使用のアプリケーションを iSeries サーバーに移植するための他のソリューションとを比較しています。

- i5/OS PASE (英語) (http://www.ibm.com/servers/enable/site/porting/iseries/pase/index.html)

  この Web サイトは、i5/OS PASE を使用した、アプリケーションの iSeries サーバーへの移植に関する情報を提供しています。

- API 分析ツール (英語) (http://www.ibm.com/servers/enable/site/porting/iseries/overview/apitool.html)

  この分析ツールは、アプリケーションでの AIX コマンド、API、およびユーティリティーの使用が、i5/OS PASE によってどのようにサポートされるかについての詳細情報を提供しています。

- AIX 文書 (英語) (http://www.ibm.com/servers/aix/library/)

  この Web サイトは、AIX のコマンドおよびユーティリティーに関する情報を提供しています。

ニュース・グループ

i5/OS PASE ニュース・グループ (news://news.software.ibm.com/ibm.software.iseries.pase) では、 i5/OS PASE に関するユーザーからの質問と回答がやり取りされています。

その他の情報

- i5/OS PASE API

  i5/OS PASE API の以下のカテゴリーの詳細については、このトピックを参照してください。

– 呼び出し可能プログラム API
– ILE プロシージャー API
– i5/OS PASE プログラムが使用するためのランタイム機能

i5/OS PASE プログラムを実行するには、システム API を呼び出す必要があります。システムでは、i5/OS PASE プログラムを実行するために、呼び出し可能プログラム API と ILE プロシージャー API の両方を提供しています。呼び出し可能プログラム API の使用は簡単ですが、 ILE プロシージャー API で使用可能なすべての制御を提供するわけではありません。

- i5/OS PASE シェルおよびユーティリティー

  i5/OS PASE には 3 つのシェル (Korn、Bourne、および C Shell)、および i5/OS PASE プログラムとして実行する 200 個近くのユーティリティーがあります。i5/OS PASE シェルおよびユーティリティーでは拡張可能なスクリプト環境を提供しており、その環境には業界標準およびデファクト・スタンダードのコマンドが多数含まれています。

- i5/OS PASE コマンド

  このトピックで説明されている i5/OS PASE コマンドのほとんどは、AIX コマンドと同じオプションをサポートしており、同じ動作をします。 i5/OS PASE コマンドに加えて、各 i5/OS PASE シェルは多数の組み込みコマンド (cd、exec、if など) をサポートします。

- i5/OS PASE ランタイム・ライブラリー

  i5/OS PASE ランタイムは AIX ランタイムで提供される大規模なインターフェースのサブセットをサポートします。i5/OS PASE でサポートされるランタイム・インターフェースのほとんどは、AIX と同じオプションおよび動作を提供します。i5/OS PASE ランタイム・ライブラリーは /usr/lib に (シンボリック・リンクとして) インストールされます。

PDF ファイルの保存

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

1. ブラウザーで PDF を右マウス・ボタン・クリックします (上部のリンクを右マウス・ボタン・クリック)。
2. PDF をローカルに保存するオプションをクリックします。
3. PDF を保存したいディレクトリーに進みます。
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 がシステムにインストールされていることが必要です。このアプリケーションは、 Adobe Web サイト (www.adobe.com/products/acrobat/readstep.html) から無料でダウンロードできます。

コードに関する特記事項

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。

IBM、そのプログラム開発者、または供給者は、いかなる場合においてもその予見の有無を問わず、以下に対する責任を負いません。

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。

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

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

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

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

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

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

IBM Corporation
Software Interoperability Coordinator, Department 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。

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「i5/OS PASE」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

以下は、IBM Corporation の商標です。

| AIX
| e(ロゴ)server
| eServer
| i5/OS
| IBM
| IBM (ロゴ)
| iSeries
| pSeries
| AFS
| DFS
| Integrated Language Environment
| NetServer
| PartnerWorld
| POWER
| PowerPC
| DB2
| DB2 Universal Database
| OS/400

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

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。

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。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

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、第三者の権利の不侵害の保証、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
〒106-8711　東京都港区六本木3-2-12