IBM 시스템 - iSeries

통합 운영 환경
i5/OS PASE

버전 5 릴리스 4

IBM

IBM 시스템 - iSeries

통합 운영 환경
i5/OS PASE

버전 5 릴리스 4

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 69 페이지의 『주의사항』의 정보를 읽으십시오.

제 6 판(2006년 2월)

이 개정판은 새 개정판에 별도로 명시하지 않는 한, IBM i5/OS(제품 번호 5722-SS1) 버전 5, 릴리스 4, 수정 0 및 모든 후속 릴리스와 수정에 적용됩니다. 이 버전은 모든 축약 명령어 세트 컴퓨터(RISC) 모델 및 CISC 모델에서 실행되지 않습니다.

목차

i5/OS PASE

IBM® i5/OS™ PASE(i5/OS Portable Application Solutions Environment)는 IBM AIX® 어플리케이션을 최소한의 노력으로 IBM iSeries™ 서버에 이식할 수 있도록 합니다.

i5/OS PASE는 AIX 또는 Linux®와 같이 복잡한 오퍼레이팅 시스템 관리 없이 선택된 어플리케이션을 실행할 수 있는 통합 런타임 환경을 제공합니다. i5/OS 또한 PASE는 강력한 스크립팅 환경을 제공하는 산업 표준 및 사실상의 표준 쉘과 유틸리티도 제공합니다.

주: 코드 예제를 사용하여 66 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의할 수 있습니다.

V5R4의 새로운 사항

이 페이지에서는 이 릴리스의 i5/OS PASE에 대한 변경사항을 요점적으로 설명합니다.

- V5R4M0에 대한 i5/OS PASE는 AIX 5.3에서 추출됩니다(i5/OS PASE V5R3M0의 경우 AIX 5.2에서 추출).
- 다음 컴파일러 제품은 i5/OS PASE의 V4R4M0에서의 실행을 지원합니다.
  - AIX용 IBM XL C/C++ Enterprise Edition, V7.0
  - AIX용 IBM XL C Enterprise Edition, V7.0
  - AIX용 IBM XL Fortran Enterprise Edition, V9.1
- 다음 유틸리티가 새로 작성되거나 변경되었습니다.
  - apt(Java™ 주석 처리 툴인 QShell apt 명령 실행)
  - pack200(Java 아카이브 패킹 툴인 QShell pack200 명령 실행)
  - unpack200(Java 아카이브 패킹 해제 툴인 Qshell unpack200 명령 실행)
- 새로 작성되거나 변경된 i5/OS PASE 런타임 함수:
  - _OPEN_CCSID(i5/OS PASE에 대한 CCSID로 열기)
- 새 로케일이 추가되었습니다.
- 새로운 예가 추가되었습니다.

새로운 사항과 변경된 사항을 보는 방법

어디에서 기술적 변경이 이루어졌는지 쉽게 확인할 수 있도록 이 정보에서는 다음을 사용합니다.

- » 새 정보나 변경된 정보가 시작되는 위치를 표시하는 이미지
- « 새 정보나 변경된 정보가 끝나는 위치를 표시하는 이미지

이 릴리스에서 변경되거나 새로운 사항에 대한 기타 정보는 사용자에 대한 메모를 참조하십시오.

관련 개념

12 페이지의『i5/OS PASE에 AIX 컴파일러 설치』
이 주제의 단계를 수행하여 i5/OS PASE에 AIX 컴파일러를 설치할 수 있습니다.

관련 정보

i5/OS PASE shells and utilities

Runtime functions for use by i5/OS PASE programs

i5/OS PASE locales

인쇄 가능한 PDF

이 주제를 사용하여 이 정보의 PDF를 보고 인쇄할 수 있습니다.

이 문서의 PDF 버전을 보거나 다운로드하려면 i5/OS PASE(약 645KB)를 선택하십시오.

PDF 파일 저장

PDF를 보거나 인쇄를 위해 워크스테이션에 저장하려면 다음을 수행하십시오.

1. 브라우저에서 PDF를 마우스 오른쪽 단추로 클릭하십시오(위의 링크를 마우스 오른쪽 단추로 클릭).
2. PDF를 로컬로 저장하는 옵션을 클릭하십시오.
3. PDF를 저장하려는 디렉토리를 탐색하십시오.
4. **저장**을 클릭하십시오.

Adobe Reader 다운로드

해당 PDF를 보거나 인쇄하려면 시스템에 Adobe Reader가 설치되어 있어야 합니다. Adobe 웹 사이트 (www.adobe.com/products/acrobat/readstep.html) 에서 무료로 다운로드할 수 있습니다.

---

i5/OS PASE 시작하기

i5/OS PASE(i5/OS Portable Application Solutions Environment)를 사용하면 대부분의 AIX 어플리케이션 2진을 거의 변경하지 않고 i5/OS에서 실행할 수 있으며 플랫폼 솔루션 포트폴리오를 효과적으로 확장시킬 수 있습니다.

교차 플랫폼 어플리케이션의 개발과 전개는 효율적 비즈니스 컴퓨팅 환경의 중요 구성요소입니다. 시스템이 제공하는 함수를 쉽게 사용하고 통합하는 것도 역시 중요합니다. 이는 iSeries의 특징입니다. 비즈니스가 점차 개방형 컴퓨팅 환경으로 이동해감에 따라 서로 상이한 목표를 달성하는 것이 어렵고 시간과 비용이 많이 든다는 것을 알게 될 것입니다. 예를 들어, AIX 오퍼레이팅 시스템에서 실행되고 해당 기능을 사용하는 익숙한 어플리케이션에서 이익을 얻기 원하면서도 AIX 및 i5/OS 오퍼레이팅 시스템의 추가된 관리 부담은 원하지 않을 수 있습니다.

이러한 경우에는 i5/OS PASE(i5/OS Portable Application Solutions Environment)가 도움이 됩니다. i5/OS PASE를 사용하면 대부분의 AIX 어플리케이션 2진을 거의 변경하지 않고 i5/OS에서 실행할 수 있으며 플랫폼 솔루션 포트폴리오를 효과적으로 확장할 수 있습니다.

i5/OS PASE 개념

i5/OS PASE는 i5/OS에서 실행하는 AIX 어플리케이션에 대한 통합 런타임 환경입니다.

AIX의 ABI(Application Binary Interface)를 지원하고 AIX 공유 라이브러리, 쉘 및 유틸리티가 제공하는 지원의 광범위한 서브세트를 제공합니다. i5/OS PASE는 IBM PowerPC® 기계 명령어의 직접 처리를 지원하므로 기계 명령어만 에뮬레이트하는 환경이 갖는 단점은 없습니다.

i5/OS PASE 어플리케이션:

- C, C++, Fortran 또는 PowerPC 어셈블러에 기록할 수 있음
- AIX PowerPC 어플리케이션과 동일한 2진 실행 형식 사용
- i5/OS 작업에서 실행
- 파일 시스템, 보안 및 소켓과 같은 i5/OS 시스템 기능 사용

i5/OS PASE는 i5/OS의 UNIX® 오퍼레이팅 시스템이 아닌 점에 유념하십시오. i5/OS PASE는 AIX 프로그램을 거의 변경하지 않고 i5/OS에서 실행하도록 설계되었습니다. AIX 또는 Linux와 같은 다른 환경의 프로그램이 i5/OS PASE에서 실행되려면 먼저 AIX에서 컴파일될 수 있도록 작성되어야 합니다.

i5/OS PASE 통합 런타임은 iSeries 서버의 라이센스 내부 코드 커널에서 실행됩니다. 이 시스템은 i5/OS PASE 및 기타 런타임 환경(ILE 및 Java 포함)에서 여러 가지 공통 i5/OS 기능의 통합을 제공합니다. i5/OS PASE는 AIX 시스템 호출의 광범위한 서브세트를 구현합니다. i5/OS PASE에 대한 시스템 지원은 i5/OS PASE 프로그램이 액세스할 수 있는 메모리를 제어하고 권한이 없는 기계 명령어만 사용하도록 제한함으로써 시스템 보안 및 무결성을 강제 수행합니다.

최소의 노력으로 신속한 어플리케이션 전개

대부분의 경우 AIX 프로그램은 거의 변경없이 i5/OS PASE에서 실행될 수 있습니다. 필요한 AIX 프로그래밍 기술의 레벨은 AIX 프로그램의 설계에 따라 다릅니다. 또한 프로그램 설계에 추가 i5/OS 어플리케이션 통합(예: CL 명령으로)을 제공하여 어플리케이션 사용자의 구성 문제점을 최소화할 수 있습니다.

i5/OS PASE는 i5/OS 시장의 성공을 공유할 솔루션 개발자를 위해 또다른 이식 옵션을 추가합니다. 이식 시간을 크게 단축시키는 수단을 제공함으로써 i5/OS PASE는 시장 출시 시간과 솔루션 개발자의 투자 이익을 향상시킬 수 있습니다.

i5/OS에서 AIX 기술의 광범위한 서브세트

i5/OS PASE는 다음을 포함하여 AIX 기술의 광범위한 서브세트를 기반으로 하는 어플리케이션 런타임을 구현합니다.

- 표준 C 및 C++ 런타임(스레드세이프 및 비스레드세이프)

- Fortran 런타임(스레드세이프 및 비스레드세이프)
- pthreads 스레드 패키지
- 자료 변환을 위한 iconv 서비스
- BSD(Berkeley Software Distributions)에 상당하는 지원
- Motif widget 세트와 함께 X Window System 클라이언트 지원
- 유사 단말기(PTY) 지원

어플리케이션은 i5/OS PASE가 지원하는 AIX의 레벨을 실행하는 AIX 워크스테이션에서 개발되고 컴파일된 다음 i5/OS에서 실행됩니다.

또는 지원되는 컴파일러 제품 중 하나를 i5/OS PASE 환경에 설치하여 i5/OS PASE에서 어플리케이션을 완전히 개발, 컴파일, 빌드 및 실행할 수 있습니다.

i5/OS PASE는 또한 강력한 스크립팅 환경을 제공하는 Korn, Bourne 및 C 쉘과 거의 200개의 유틸리티도 포함합니다.

i5/OS PASE는 AIX 및 i5/OS 오퍼레이팅 시스템에 대해 IBM의 일반 프로세서 기술을 사용합니다. PowerPC 프로세서는 i5/OS PASE 런타임에서 어플리케이션을 실행하기 위해 i5/OS 모드에서 AIX 모드로 전환합니다.

i5/OS PASE에서 실행하는 어플리케이션은 i5/OS 통합 파일 시스템 및 iSeries용 DB2 Universal Database™과 통합됩니다. 어플리케이션은 Java 및 ILE(Integrated Language Environment®) 어플리케이션을 호출할 수 있습니다(이들 어플리케이션에 의해 호출될 수도 있음). 일반적으로, 보안, 메세지 처리, 통신, 백업 및 회복과 같은 i5/OS 조작 환경의 모든 측면을 활용할 수 있습니다. 동시에 AIX 인터페이스에서 파생된 어플리케이션 인터페이스도 활용할 수 있습니다.

관련 참조

10 페이지의『AIX 소스 컴파일』

i5/OS PASE에서의 설치를 지원하는 AIX 컴파일러 제품 중 하나를 설치하여 i5/OS PASE 환경에서 사용자 프로그램을 컴파일할 수도 있습니다.

관련 정보

i5/OS PASE shells and utilities

어플리케이션 개발에서 i5/OS PASE가 유용한 경우

API 분석을 사용하여 어플리케이션이 i5/OS PASE에 적합한지 여부를 판별할 수 있습니다. i5/OS PASE가 모든 상황에서 최상의 솔루션이 되지는 않습니다.

i5/OS PASE는 AIX 어플리케이션을 iSeries 서버에 이식하는 방법을 결정할 때 상당한 유연성을 제공합니다. 물론 i5/OS PASE는 사용자가 선택할 수 있는 여러 가지 옵션 중 단 하나의 옵션입니다.

API 분석

어플리케이션이 i5/OS PASE에 적합한지 여부를 판별하는 출발점은 어플리케이션의 분석입니다. 즉, 어플리케이션이 사용하는 API, 라이브러리 및 유틸리티를 분석하고 i5/OS에서 어플리케이션이 얼마나 효과적으로 실행할지를 분석하는 것입니다. IBM Virtual Innovation Center for Hardware는 API 분석 툴을 통해 이러한 영역에서 도움을 줍니다. API 분석 툴은 어플리케이션을 분석하고 잠재적 장애물을 설명하는 무료 이식 평가 툴입니다. i5/OS PASE에 어플리케이션을 이식하는 프로시듀어에 이 분석 툴을 적용하는 방법에 대한 자세한 정보는 8 페이지의『i5/OS PASE에서 실행할 프로그램 준비』주제를 참조하십시오.

잠재적 i5/OS PASE 어플리케이션의 특성

다음은 i5/OS PASE의 사용 여부를 결정할 때 고려할 수 있는 몇 가지 유용한 지침입니다.

- **AIX 어플리케이션이 계산 집약적입니까?**

  i5/OS PASE는 최적화된 연산 라이브러리를 제공하여 iSeries 서버에서 계산이 많은 어플리케이션을 실행하기에 좋은 환경을 제공합니다.

- **어플리케이션이 fork(), X Window System 또는 유사 단말기(PTY) 지원과 같이 i5/OS PASE에서만 지원되는(또는 ILE에서 부분적으로만 지원됨) 기능에 의존하는 비중이 높습니까?**

  i5/OS PASE는 fork() 및 exec()를 지원하지만 이들 함수는 현재 i5/OS 시스템에 없습니다(form() 함수를 exec() 함수와 통합시키는 spawn() 함수 제외).

- **어플리케이션이 복잡한 AIX 시스템 기반 빌드 프로세스 또는 테스트 환경을 사용합니까?**

  i5/OS PASE를 사용하면 새 오퍼레이팅 시스템으로 쉽게 전송되지 않는 기존의 복잡한 프로세스가 있을 때 특히 유용한 AIX 시스템 기반 빌드 프로세스를 사용할 수 있습니다.

- **어플리케이션이 ASCII 문자 세트에 대한 종속성을 가지고 있습니까?**

  i5/OS PASE는 이러한 요구사항이 있는 어플리케이션에 대해 좋은 지원을 제공합니다.

- **어플리케이션이 많은 포인터 조작을 수행하거나 정수를 포인터로 변환(캐스트)합니까?**

  i5/OS PASE는 저렴한 수행 비용과 정수를 포인터로 변환할 수 있는 기능으로 32비트 및 64비트의 AIX 주소지정 모델을 둘 다 지원합니다.

i5/OS PASE가 최적 솔루션이 될 수 없는 경우

i5/OS PASE는 일반적으로 ILE에서 호출되어야 하는 여러 개의 호출 가능 인터페이스를 제공하고 다음 특성을 갖는 코드에는 적합하지 않습니다.

- 호출할 때마다 i5/OS PASE를 시작 또는 종료하거나 이미 사용 중인 i5/OS PASE 프로그램에서 Qp2CallPase API를 통해 i5/OS PASE 프로시듀어를 호출하여 제공된 것보다 더 높은 성능 호출 및 리턴을 필요로 하는 코드.

- ILE 호출자와 라이브러리 코드 사이에서 메모리나 이름공간을 공유해야 하는 코드. i5/OS PASE 프로그램은 프로그램을 호출한 ILE 코드와 메모리 또는 이름공간을 내재적으로 공유하지 않습니다. (그러나 i5/OS PASE에서 호출된 ILE 코드는 i5/OS PASE 메모리를 공유하거나 사용할 수 있습니다.)

관련 정보

API 분석 툴

하드웨어에 대한 IBM Virtual Innovation Center

i5/OS PASE 설치

이 주제의 지침을 수행하여 i5/OS PASE를 서버에 설치할 수 있습니다.

i5/OS PASE는 모든 iSeries 서버에서 무료로 사용할 수 있습니다. i5/OS PASE를 설치할 것을 권장합니다. 향상된 DNS(Domain Name System) 서버 및 ILE C++ 컴파일러와 같은 일부 시스템 소프트웨어에 i5/OS PASE 지원이 필요합니다.

서버에 i5/OS PASE를 설치하려면 다음 단계를 완료하십시오.

1. i5/OS 명령행에서 `GO LICPGM`을 입력하십시오.
2. 11(라이센스 프로그램 설치)을 선택하십시오.
3. 옵션 33(5722SS1 - Portable Application Solutions Environment)을 선택하십시오.
4. **옵션:** 추가 로케일을 설치하십시오.

   i5/OS PASE 제품은 i5/OS에 설치한 언어 피처와 연관된 로케일 오브젝트만 설치합니다. 서버에 있는 언어 피처에 포함되지 않은 로케일이 필요하면 추가 i5/OS 언어 코드를 주문하여 설치해야 합니다. 자세한 정보는 i5/OS PASE 국제화 및 i5/OS PASE 로케일을 참조하십시오.

i5/OS PASE에 어플리케이션을 이식하는 소프트웨어 개발자를 위한 라이센스 부여 정보:

i5/OS PASE는 i5/OS 시스템에 AIX 런타임 라이브러리의 서브세트를 제공합니다. i5/OS 라이센스는 i5/OS와 함께 제공된 모든 라이브러리 코드를 사용할 수 있는 권한을 부여합니다. 이 라이센스는 i5/OS PASE와 함께 제공되지 않은 AIX 라이브러리에 대한 라이센스는 포함하지 않습니다. 모든 AIX 제품은 IBM에서 별도로 라이센스를 부여합니다.

사용자 고유 어플리케이션을 i5/OS PASE에 이식하면 어플리케이션이 i5/OS PASE와 함께 제공되지 않은 AIX 라이브러리에 대해 종속성을 갖는 것을 알게 될 수 있습니다. 이들 라이브러리를 i5/OS 시스템에 이식하려면 먼저 해당 라이브러리를 제공한 소프트웨어 제품을 판별한 후 이 소프트웨어 제품에 대한 라이센스 계약의 조건을 살펴보아야 합니다. i5/OS 시스템에 이식하려면 IBM 또는 써드파티와의 협력이 필요할 수 있습니다. 이식을 시작하기 전에 이식 중인 코드와 관련한 모든 라이센스 계약을 면밀히 검토해야 합니다. IBM에 속한 라이브러리에 대한 라이센스 계약을 찾으려면 IBM 영업대표, IBM Porting Center 중 하나, Rochester의 고객 기술 센터 또는 개발자용 PartnerWorld®에 문의하십시오.

관련 개념

57 페이지의『국제화』

i5/OS PASE 런타임이 AIX 런타임에 기반하므로 i5/OS PASE 프로그램에서는 AIX에 지원되는 로케일, 문자 스트링 조작, 날짜 및 시간 서비스, 메세지 카탈로그 및 문자 코드화 변환 등에 동일한 여러 프로그래밍 인터페이스 세트를 사용할 수 있습니다.

관련 정보

i5/OS PASE locales

---

i5/OS PASE 계획

i5/OS PASE에 대한 작업을 시작할 때 이 주제에 나열된 사항이 도움이 될 수 있습니다.

i5/OS PASE는 최소의 노력으로 AIX 어플리케이션을 iSeries 서버로 이식할 수 있는 i5/OS에 AIX 런타임 환경을 제공합니다. 실제로 많은 AIX 프로그램이 변경 없이 i5/OS PASE에서 실행됩니다. 그 이유는 i5/OS PASE가 AIX에서 사용할 수 있는 여러 동일한 공유 라이브러리를 제공하고 pSeries® AIX PowerPC 프로세서에서 실행하는 것과 동일한 방법으로 iSeries PowerPC 프로세서에서 직접 실행되는 AIX 유틸리티의 광범위한 서브세트를 제공하기 때문입니다.

i5/OS PASE에 대한 작업을 시작할 때 다음과 같은 사항에 유의하십시오.

- **AIX 2진의 목표 릴리스와 2진이 실행될 i5/OS PASE의 릴리스 사이에 상관이 있습니다.**

  AIX에서 i5/OS PASE 어플리케이션을 컴파일하는 경우 AIX에서 작성된 어플리케이션 2진은 어플리케이션을 실행할 i5/OS PASE의 버전과 호환되어야 합니다. 다음 표는 i5/OS PASE의 여러 버전과 호환되는 AIX 2진 버전을 보여줍니다. 에를 들어 AIX 릴리스 5.1용으로 작성된 32비트 어플리케이션 i5/OS PASE V5R4, V5R3 또는 OS/400® PASE V5R2에서 실행되지만 OS/400 PASE V5R1에서는 실행되지 않습니다. 마찬가지로 AIX 릴리스 4.3용으로 작성된 64비트 어플리케이션은 OS/400 PASE V5R1에서는 실행되지만 i5/OS PASE V5R4, V5R3 또는 OS/400 PASE V5R2에서는 실행되지 않습니다.

AIX 릴리스	OS/400 V5R1	OS/400 V5R2	i5/OS V5R3	i5/OS V5R4
4.3(32비트)	X	X	X	X
4.3(64비트)	X	-	-	-
5.1(32비트 또는 64비트)	-	X	X	X
5.2(32비트 또는 64비트)	-	-	X	X
5.3(32비트 또는 64비트)	-	-	-	X

- **i5/OS PASE는 i5/OS에서 AIX 커널을 제공하지 않습니다.**

  대신, 공유 라이브러리에 필요한 하위 레벨 시스템 함수는 i5/OS 커널 또는 통합 i5/OS 함수로 라우트됩니다. 이러한 점에서 i5/OS PASE는 AIX과 i5/OS 플랫폼 사이의 간격을 메워줍니다. 코드는 공유 라이브러리의 API에 대해 AIX에서와 같은 구문을 사용하지만 i5/OS PASE 프로그램은 i5/OS 작업에서 실행되고 다른 i5/OS 작업처럼 i5/OS에 의해 관리됩니다.
- **대부분의 경우 i5/OS PASE에서 호출하는 API는 AIX에서와 동일한 방식으로 작동합니다.**

그러나 일부 API는 i5/OS PASE에서 다르게 작동하거나 i5/OS PASE에서 지원되지 않을 수도 있습니다. 이 때문에 i5/OS PASE 프로그램을 준비하기 위한 계획은 API 분석 툴을 사용한 철저한 코드 분석으로 시작되어야 합니다. 이 툴은 AIX 어플리케이션을 i5/OS PASE로 이식할 때 고려해야 할 프로그램 수정 유형에 대한 포괄적인 요약을 제공합니다.

- **AIX와 i5/OS 플랫폼 간의 몇 가지 차이점을 고려하십시오.**
  - AIX는 일반적으로 대소문자를 구분하지만 특정 i5/OS 파일 시스템은 대소문자를 구분하지 않습니다.
  - AIX는 자료 코드화에 일반적으로 ASCII를 사용하지만 i5/OS는 일반적으로 EBCDIC(Extended Binary Coded Decimal Interchange Code)를 사용합니다. i5/OS PASE 프로그램에서 ILE 코드 호출에 대한 세부사항을 관리할 경우 이 점을 고려해야 합니다. 예를 들어, i5/OS PASE에서 임의의 ILE 프로시듀어에 대해 호출할 때 스트링의 문자 코드화 변환을 처리하도록 i5/OS PASE 프로그램을 명시적으로 코딩해야 합니다. i5/OS PASE 런타임 지원은 문자 코드화 변환에 대해 iconv_open(), iconv() 및 iconv_close() 함수를 포함합니다.

    주: i5/OS PASE 및 ILE는 각각 자체 변환표를 사용하여 iconv() 인터페이스를 독립적으로 구현합니다. i5/OS PASE iconv() 지원이 지원하는 변환은 통합 파일 시스템에 바이트 스트림 파일로 저장되므로 사용자가 이를 수정하고 확장할 수 있습니다.
  - AIX 어플리케이션에서는 행(예: 파일 및 쉘 스크립트의 행)이 라인 피드(LF)로 끝나지만 퍼스널 컴퓨터(PC) 소프트웨어 및 i5/OS 소프트웨어에서는 행이 일반적으로 캐리지 리턴 및 라인 피드(CRLF)로 끝납니다.
  - AIX에서 사용하는 일부 스크립트 및 프로그램은 표준 유틸리티에 대한 하드 코딩된 경로를 사용할 수 있으므로 i5/OS PASE에서 사용할 경로를 나타내도록 경로를 수정할 필요가 있을 수 있습니다. 자세한 정보는 i5/OS PASE와 프로그램의 호환성 분석을 참조하십시오.

i5/OS PASE는 이러한 문제의 일부를 자동으로 처리합니다. 예를 들어, 시스템이 제공하는 i5/OS PASE 런타임 서비스(i5/OS 옵션 33과 함께 제공되는 공유 라이브러리의 시스템 호출 또는 런타임 함수 포함)를 사용할 경우, 파일 설명자(바이트 스트림 파일 또는 소켓)에 쓰거나 읽은 쓴 자료에 대해서는 일반적으로 변환이 수행되지 않더라도 i5/OS PASE는 필요한 경우 ASCII에서 EBCDIC으로 변환을 수행합니다.

다른 하위 레벨 함수(예: _ILECALL)를 사용하여 ILE 함수 및 API에 대한 호출로 i5/OS PASE 프로그램의 기능을 확장할 수 있지만 위에 언급한 것처럼 자료 변환 처리가 필요할 수 있습니다. 또한, 이러한 확장을 프로그램에 코딩하려면 추가 헤더 및 내보내기 파일을 사용해야 합니다.

관련 개념

9 페이지의『프로그램과 i5/OS PASE의 호환성 분석』

iSeries 서버에 대한 C 어플리케이션의 이식성을 평가하는 첫 번째 단계는 어플리케이션에 사용된 인터페이스를 분석하는 것입니다.

i5/OS PASE에서 실행할 프로그램 준비

i5/OS에서 효과적으로 실행되는 AIX 프로그램을 준비하기 위해 수행해야 하는 단계는 프로그램의 특성 및 i5/OS 시스템 고유 인터페이스와 함수에 대한 실제적인 필요에 따라 다릅니다.

i5/OS PASE에 어플리케이션을 이식하는 경우 먼저 어플리케이션이 AIX 컴파일러를 사용하여 컴파일하는지 확인해야 합니다. 이 요구사항에 맞게 프로그램을 수행해야 하는 경우도 있습니다.

프로그램과 i5/OS PASE의 호환성 분석

iSeries 서버에 대한 C 어플리케이션의 이식성을 평가하는 첫 번째 단계는 어플리케이션에 사용된 인터페이스를 분석하는 것입니다.

이 API 분석은 산업 표준이 아니며 i5/OS에서 지원되지 않는 어플리케이션에서 사용되는 해당 인터페이스를 식별합니다. 또한, AIX 또는 Linux 시스템과 i5/OS의 구조가 다르기 때문에 산업 표준과는 호환되지만 서로 다르게 지원되는 인터페이스를 식별합니다.

API 분석 툴은 프론트엔드 프로세스 및 백엔드 프로세스로 구성됩니다. 프런트엔드 프로세스는 컴파일된 어플리케이션을 스캔하여 어플리케이션에서 사용하는 인터페이스(외부 기능 및 자료)를 추출하고 이들 모든 인터페이스의 리스트를 생성합니다. 백엔드 프로세스는 이 인터페이스 리스트를 입력하여 일반 시스템 API 및 이의 지원 데이터베이스와 이 인터페이스를 비교합니다.

API 분석 툴의 프론트엔드 프로세스는 쉘 스크립트입니다. `nm` 또는 `dump` 명령을 사용하여 어플리케이션의 외부 기호 표에서 기호 정보를 찾습니다.

기호가 제거된 2진에는 툴이 분석을 수행하기에 충분한 동적 바인딩 정보를 포함할 수 있습니다. 정적으로 바인드시킨 2진은 라이브러리 인터페이스를 분석에서 제거하지만 분석을 위해 시스템 호출 종속성은 보여줍니다.

컴파일하기 전에 수행할 추가 분석

API 분석 툴을 통해 수집한 정보 외에 다음 정보도 수집해야 합니다.

- 어플리케이션에서 사용한 라이브러리 리스트 가져오기

  분석 툴은 어플리케이션에서 사용하는 일부 표준 API에 대한 피드백을 제공하지만 많은 공통 API 세트를 찾지는 못합니다. 라이브러리 분석은 사용자 어플리케이션이 사용하는 일부 미들웨어 API를 식별하는 데 도움이 됩니다. 각각의 사용자 명령과 공유 오브젝트에 대해 다음 명령을 실행하여 어플리케이션에 필요한 라이브러리 리스트를 얻을 수 있습니다.

  ```
  dump -H binary_name
  ```

- 코드에서 하드 코드 경로명 검사

  증명서를 변경하는 프로그램을 실행하거나 i5/OS PASE 환경 변수 PASE_EXEC_QOPENSYS=N인 경우에도 프로그램 또는 스크립트를 실행하려면 하드 코드된 경로명을 변경해야 합니다.

  /usr/bin/ksh가 절대 경로(루트에서 시작함)이므로 이 경로가 없거나 바이트 스트림 파일이 아니면 i5/OS PASE는 /QOpenSys 파일 시스템에서 searches /QOpenSys/usr/bin/ksh 경로명을 탐색합니다. QShell 유틸리티 프로그램은 바이트 스트림 파일이 아니므로 i5/OS PASE는 원래(절대) 경로가 QShell 유틸리티 프로그램에 대한 기호 링크(예: /usr/bin/sh)인 경우에도 /QOpenSys 파일 시스템을 탐색합니다.

  관련 개념

7 페이지의『i5/OS PASE 계획』

i5/OS PASE에 대한 작업을 시작할 때 이 주제에 나열된 사항이 도움이 될 수 있습니다.

관련 정보

API 분석 툴

AIX 소스 컴파일

i5/OS PASE에서의 설치를 지원하는 AIX 컴파일러 제품 중 하나를 설치하여 i5/OS PASE 환경에서 사용자 프로그램을 컴파일할 수도 있습니다.

프로그램이 AIX 인터페이스만 사용할 경우, 필수 AIX 헤더를 사용하여 컴파일하고 AIX 라이브러리와 링크하여 i5/OS PASE에 대한 2진을 준비하십시오. i5/OS PASE는 AIX 시스템 제공 공유 라이브러리와 정적으로 바인드된 어플리케이션을 지원하지 않는 점을 유의하십시오.

i5/OS PASE 프로그램은 PowerPC용 AIX 프로그램과 구조적으로 동일합니다.

i5/OS PASE(오퍼레이팅 시스템의 옵션 33)는 컴파일러를 포함하지 않습니다. AIX 시스템을 사용하여 i5/OS PASE 프로그램을 컴파일하거나 i5/OS PASE에서의 설치를 지원하는 AIX 컴파일러 제품 중 하나를 사용하여 i5/OS PASE 환경에서 사용자 프로그램을 컴파일할 수도 있습니다.

pSeries 서버에서 AIX 컴파일러 사용

PowerPC용 AIX ABI와 호환 가능한 출력을 생성하는 AIX 컴파일러 및 링크 프로그램을 사용하여 i5/OS PASE 프로그램을 빌드할 수 있습니다. i5/OS PASE는 PowerPC에 존재하지 않는 POWER™ 구조 명령어(캐시 관리에 대한 IBM POWER 명령어 제외)를 사용하는 2진에 대한 명령어 에뮬레이션 지원을 제공합니다.

i5/OS PASE에서 AIX 컴파일러 사용

i5/OS PASE는 i5/OS PASE 환경에서 다음과 같이 독립적으로 사용할 수 있는 AIX 컴파일러의 설치를 지원합니다.

- AIX용 IBM XL C/C++ Enterprise Edition, V7.0(5724-I11)
- AIX용 IBM XL C Enterprise Edition, V7.0(5724-I10)
- AIX용 IBM XL Fortran Enterprise Edition, V9.1(5724-I08)

이들 제품을 사용하여 iSeries의 i5/OS PASE 환경에서 AIX 어플리케이션을 개발, 컴파일, 빌드 및 실행할 수 있습니다.

개발 툴

AIX에서 사용하는 여러 개발 툴(예: `ld`, `ar`, `make`, `yacc`)이 i5/OS PASE와 함께 포함됩니다. 다른 소스의 여러 AIX 툴(예: 오픈 소스 툴 `gcc`)도 i5/OS PASE에서 사용할 수 있습니다.

iSeries Tools for Developers PRPQ(5799-PTL)에도 iSeries 어플리케이션의 개발, 빌드 및 이식을 지원하는 다양한 툴 배열이 있습니다. 이 PRPQ에 대한 자세한 정보는 IBM Virtual Innovation Center for Hardware 웹 사이트를 참조하십시오.

포인터 처리에 대한 컴파일러 참고사항

- xlc 컴파일러는 -qlngdbl128 및 -qalign=natural의 조합을 사용하여 16바이트 정렬에 대해 제한된 지원을 제공합니다(유형 long double의 경우). *ILEpointer* 유형을 사용하려면 MI(Machine Interface) 포인터가 구조 내에서 16바이트로 정렬되게 하기 위해 이러한 컴파일러 옵션이 필요합니다. -qldbl128 옵션을 사용하면 유형 long double은 long double 필드에 대해 printf와 같은 조작을 처리할 수 있도록 libc128.a의 사용을 필요로 하는 128비트 유형으로 만들어집니다.

  -qlngdbl128 옵션을 가져와서 libc128.a로 링크하는 쉬운 방법은 xlc 명령 대신 xlc128 명령을 사용하는 것입니다.
- xlc/xlC 컴파일러는 현재 정적 변수나 자동 변수에 대해 16바이트 정렬을 강제하는 수단을 제공하지 않습니다. 단순히 이 컴파일러는 구조 내에서 128비트 long double 필드에 대한 상대적인 정렬을 보장합니다. i5/OS PASE 버전의 malloc는 16바이트 정렬 기억장치를 항상 제공하므로 스택 기억장치의 16바이트 정렬을 배열할 수 있습니다.
- 헤더 파일 as400_types.h는 유형 long long에서도 64비트 정수가 됩니다. xlc 컴파일러의 -qlonglong 옵션이 이 구조를 보장합니다(xlc 컴파일러를 실행하는 모든 명령의 디폴트가 아님).

예

다음 예는 AIX 시스템에서 i5/OS PASE 프로그램을 컴파일할 때 사용할 수 있습니다. i5/OS PASE에 설치된 컴파일러를 사용하여 프로그램을 컴파일하는 경우 i5/OS 시스템 고유 헤더 파일 또는 i5/OS 시스템 고유 내보내기 파일의 위치에 대해 컴파일러 옵션을 지정하지 않아도 됩니다. 이는 이들 파일이 i5/OS 시스템의 디폴트 경로 위치인 /usr/include/ 및 /usr/lib/에 있기 때문입니다.

예 1

AIX 시스템에서 다음 명령은 libc.a에서 내보낸 i5/OS 시스템 고유 인터페이스를 사용할 수 있는 i5/OS PASE 프로그램 testpgm을 작성합니다.

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

이 예는 i5/OS 시스템 고유 헤더 파일을 AIX 디렉토리 /usr/include로 복사하고 i5/OS 시스템 고유 내보내기 파일을 AIX 디렉토리 /mydir로 복사한 것으로 가정합니다.

예 2

다음 예는 i5/OS 시스템 고유 헤더 및 내보내기 파일이 /pase/lib에 있는 것으로 가정합니다.

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

예 3

다음 예는 예 2와 동일한 프로그램을 같은 옵션을 사용하여 빌드합니다. 그러나 컴파일된 어플리케이션이 스레드세이프 런타임 라이브러리와 링크되도록 하기 위해 멀티스레드 프로그램에 `xlc_r` 명령을 사용합니다.

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

이들 예에서 iSeries용 IBM DB2 Universal Database(UDB) CLI(Call Level Interface)에 i5/OS PASE 지원을 사용하는 경우 빌드 명령에 -bI:/pase/include/libdb400.exp도 지정해야 합니다.

-bI 지시문은 `ld` 명령에 매개변수를 전달하도록 컴파일러에 지시합니다. 지시문은 라이브러리에서 내보낸 기호가 들어 있는 내보내기 파일을 어플리케이션이 가져오도록 지정합니다.

관련 개념

3 페이지의『i5/OS PASE 개념』

i5/OS PASE는 i5/OS에서 실행하는 AIX 어플리케이션에 대한 통합 런타임 환경입니다.

관련 정보

i5/OS PASE shells and utilities

i5/OS PASE에 AIX 컴파일러 설치

이 주제의 단계를 수행하여 i5/OS PASE에 AIX 컴파일러를 설치할 수 있습니다.

별도로 사용할 수 있는 다음 AIX 컴파일러 중 하나를 i5/OS PASE 환경에 설치할 수 있습니다.

- AIX용 IBM XL C/C++ Enterprise Edition, V7.0(5724-I11)
- AIX용 IBM XL C Enterprise Edition, V7.0(5724-I10)
- AIX용 IBM XL Fortran Enterprise Edition, V9.1(5724-I08)

이들 제품을 사용하면 iSeries 서버의 i5/OS PASE 환경에서 AIX 어플리케이션을 전체적으로 개발, 컴파일, 빌드 및 실행할 수 있습니다.

관련 개념

1 페이지의『V5R4의 새로운 사항』

이 페이지에서는 이 릴리스의 i5/OS PASE에 대한 변경사항을 요점적으로 설명합니다.

관련 정보

AIX용 XL C/C++ Enterprise Edition

AIX용 XL C Enterprise Edition

AIX 컴파일러 설치:

i5/OS PASE는 일반적으로 AIX 시스템에서 어플리케이션을 설치하는 데 사용되는 AIX `smit` 또는 `installp` 유틸리티를 지원하지 않습니다. AIX용 XL C/C++ Enterprise Edition, V7.0 또는 AIX용 XL C Enterprise Edition 제품의 설치는 개별 컴파일러의 설치 매체에 포함된 ″비 디폴트 설치″ 스크립트를 통해 수행됩니다.

다음 단계는 AIX용 XL C/C++ Enterprise Edition, 또는 AIX용 XL C Enterprise Edition 제품을 iSeries i5/OS PASE에 설치합니다.

1. 필요한 전제조건이 있는지 확인하십시오. 컴파일러 설치 매체 외에도 이 컴파일러를 성공적으로 설치하고 사용하려면 iSeries 서버에 다음 프로그램도 설치해야 합니다.
   - 5722SS1 옵션 33 - i5/OS PASE 자체
   - 5722SS1 옵션 13 - 시스템 개방성 포함, /usr/include 통합 파일 시스템 디렉토리의 컴파일러 헤더 파일 포함
   - Perl. 컴파일러 설치 스크립트에는 Perl이 필요합니다. 다음은 Perl을 설치하는 두 가지 방법입니다.
     – 5799PTL - iSeries Tools for Developers PRPQ. Perl(여러 가지 다른 개발 툴과 함께)은 별도로 사용할 수 있는 iSeries Tools For Developers PRPQ에 포함됩니다.
     – http://www.cpan.org/ports/#os400 - i5/OS PASE용 Perl 포트 2진 분배
2. 컴파일러 제품 설치 CD를 iSeries CDROM 장치에 삽입하십시오.
3. *ALLOBJ 권한이 있는 사용자 프로파일로 i5/OS에 사인 온하십시오. 이 사용자 프로파일에서 컴파일러 제품 파일을 소유합니다.
4. 다음 CL 명령을 입력하여 대화식 i5/OS PASE 단말기 세션을 시작하십시오. `call qp2term`
5. 다음 명령을 입력하여 해당 컴파일러 설치 스크립트를 복원하십시오.

컴파일러	명령
AIX용 XL C/C++ Enterprise Edition의 경우	`cd / restore -qf /QOPT/CDROM/USR/SYS/INST.IMA/VACPP.NDI ./usr/vacpp/bin/vacppndi`
AIX용 XL C Enterprise Edition의 경우:	`cd / restore -qf /QOPT/CDROM/USR/SYS/INST.IMA/VAC.NDI ./usr/vac/bin/vacndi`
AIX용 XL Fortran Enterprise Edition의 경우:	`cd /` `restore -qf /QOPT/CDROM/USR/SYS/INST.IMA/XLF.NDI ./usr/lpp/xlf/bin/xlfndi`

6. 설치 스크립트를 실행하여 컴파일러를 설치하십시오. 컴파일러의 대상 디렉토리는 해당 명령의 -b 옵션에서 지정합니다. 다음 표의 명령에는 컴파일러에 권장되는 디렉토리명을 사용합니다. 다른 디렉토리를 선택할 경우 디렉토리가 /QOpenSys 트리에 있어야 합니다(파일명에 대소문자 구분 허용).

컴파일러	명령
AIX용 XL C/C++ Enterprise Edition의 경우	`/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/vacpp/bin/vacppndi -i -d /QOPT/CDROM/USR/SYS/INST.IMA -b /QOpenSys/xlc70`
AIX용 XL C Enterprise Edition의 경우:	`/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/vac/bin/vacndi -i -d /QOPT/CDROM/USR/SYS/INST.IMA -b /QOpenSys/xlc70`
AIX용 XL Fortran Enterprise Edition의 경우:	`/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/lpp/xlf/bin/xlfndi -i -d /QOPT/CDROM/USR/SYS/INST.IMA -b /QOpenSys/xlf91`
주: 명령을 하나의 긴 명령으로 입력하십시오.	

이제 i5/OS PASE에서 사용할 수 있도록 컴파일러가 설치되었습니다.

AIX용 XL C/C++ Enterprise Edition 컴파일러 명령(예: xlC)은 /QOpenSys/xlc70/usr/vacpp/bin/ 디렉토리에 있습니다. 이 디렉토리를 $PATH 환경 변수에 추가할 수 있습니다.

AIX용 XL C/C++ Enterprise Edition 컴파일러 문서는 /QOpenSys/xlc70/usr/vacpp/pdf/en_US/ 디렉토리에 있으며 Adobe Acrobat 형식으로 되어 있습니다.

AIX용 XL C Enterprise Edition 컴파일러 명령(예: xlc 및 cc)은 /QOpenSys/xlc70/usr/vac/bin/ 디렉토리에 있습니다. 이 디렉토리를 $PATH 환경 변수에 추가할 수 있습니다.

AIX용 XL C Enterprise Edition 컴파일러 문서는 /QOpenSys/xlc70/usr/vac/pdf/en_US/ 디렉토리에 있으며 Adobe Acrobat 형식으로 되어 있습니다.

AIX용 XL Fortran 컴파일러 명령(예: xlf)은 /QOpenSys/xlf91/usr/bin/ 디렉토리에 있습니다. 이 디렉토리를 $PATH 환경 변수에 추가할 수 있습니다.

AIX용 XL Fortran 컴파일러 문서는 /QOpenSys/xlf91/usr/share/man/info/en_US/xlf/pdf/ 디렉토리에 있으며 Adobe Acrobat 형식으로 되어 있습니다.

PTF 갱신 지침:

AIX용 XL C/C++ Enterprise Edition, V7.0 및 AIX용 XL C Enterprise Edition 제품에 대한 프로그램 임시 수정(PTF)의 설치는 초기 컴파일러 설치에 사용되는 스크립트와 동일한 ″비 디폴트 설치″ 스크립트를 사용하여 수행됩니다.

PTF를 설치하기 전에 먼저 이 주제에 있는 이전 단계를 사용하여 컴파일러를 설치해야 합니다.

다음 단계는 AIX용 XL C/C++ Enterprise Edition, V7.0 또는 AIX용 XL C Enterprise Edition 제품에 대한 PTF를 iSeries i5/OS PASE에 설치합니다.

1. 설치할 PTF 패키지 파일을 받으십시오. XL C/C++ Enterprise Edition 웹 사이트의 지원 다운로드 섹션에서 컴파일러 PTF 패키지의 압축된 TAR 이미지를 다운로드할 수 있습니다.
2. PTF 패키지 파일의 압축을 푸십시오. 압축된 TAR 이미지를 /QOpenSys/vacptf/ 디렉토리로 다운로드했으면 QP2TERM 명령행에서 다음 명령을 사용하여 이를 수행할 수 있습니다.

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

3. 설치할 PTF 패키지의 리스트를 포함하는 파일을 작성하십시오. QP2TERM 명령행에서 다음 명령을 사용하여 이를 수행하십시오.

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

4. 설치 스크립트를 실행하여 PTF를 설치하십시오. 갱신 중인 컴파일러에 따라 QP2TERM 명령행에서 다음 명령 중 하나를 입력하십시오.

컴파일러	명령
AIX용 XL C/C++ Enterprise Edition의 경우	/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/vacpp/bin/vacppndi -d /QOpenSys/ptf -b /QOpenSys/xlc70 -u /QOpenSys/ptf/ptflist.txt
AIX용 XL C Enterprise Edition의 경우:	/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/vac/bin/vacndi -d /QOpenSys/ptf -b /QOpenSys/xlc70 -u /QOpenSys/ptf/ptflist.txt
AIX용 XL Fortran Enterprise Edition의 경우:	/QIBM/ProdData/DeveloperTools/pase/bin/perl /usr/lpp/xlf/bin/xlfndi -d /QOpenSys/ptf -b /QOpenSys/xlf91 -u /QOpenSys/ptf/ptflist.txt
주: 명령을 하나의 긴 명령으로 입력하십시오.	

설치 스크립트는 PTF 갱신 전에 있던 컴파일러 파일의 압축된 TAR 백업을 작성합니다. 이들 명령어에 표시된 디렉토리를 사용할 경우 이 파일의 이름은 /QOpenSys/xlc70.backup.tar.Z 또는 /QOpenSys/xlf91.backup.tar.Z으로 지정됩니다. PTF 갱신 설치 시 또는 PTF 갱신 자체에 문제점이 있으면 이 백업에서 복원하여 PTF 갱신을 설치 제거할 수 있습니다.

관련 정보

AIX용 XL C/C++ Enterprise Edition

i5/OS PASE 프로그램을 iSeries 서버에 복사

i5/OS PASE에서 실행하려는 AIX 2진을 통합 파일 시스템에 복사하십시오.

통합 파일 시스템에서 사용할 수 있는 모든 파일 시스템은 i5/OS PASE에서 사용할 수 있습니다.

오퍼레이팅 시스템 간에 파일을 이동할 때 어플리케이션이 대소문자를 구별하는지 및 AIX가 사용하는 행 종료 문자와 i5/OS에 사용하는 행 종료 문자가 차이가 있는지 알고 있어야 합니다. 이러한 차이점은 문제를 일으킬 수 있습니다.

파일 전송 프로토콜(FTP), 서버 메세지 블록(SMB) 또는 리모트 파일 시스템을 사용하여 i5/OS PASE 프로그램 및 관련 파일을 iSeries 서버와 주고 받을 수 있습니다.

관련 참조

19 페이지의『헤더 파일 복사』

이 주제의 지침을 수행하여 iSeries 서버에서 AIX 기계로 헤더 파일을 복사할 수 있습니다.

21 페이지의『내보내기 파일 복사』

이 주제의 지침을 수행하여 iSeries 서버에서 AIX 디렉토리로 내보내기 파일을 복사할 수 있습니다.

관련 정보

Integrated file system

대소문자 구분

어플리케이션이 대소문자를 구분할 경우, /QOpenSys 파일 시스템으로 이동시키거나, 대소문자를 구분하도록 작성된 사용자 정의 파일 시스템으로 이동시키십시오.

AIX 및 Linux와 같은 오퍼레이팅 시스템의 인터페이스는 일반적으로 대문자와 소문자를 구분합니다. i5/OS에서는 항상 대소문자를 구분하지는 않습니다. 대소문자 구분이 기존 코드를 복잡하게 만드는 경우 몇 가지 상황에 특히 유의해야 합니다.

디렉토리 또는 파일에서 대소문자 구분은 i5/OS에서 사용하는 파일 시스템에 따라 다릅니다. /QOpenSys 파일 시스템은 대소문자를 구분하므로 대소문자가 구분되는 사용자 정의 파일 시스템(UDFS)을 작성할 수 있습니다.

예

다음 예는 대소문자 구분으로 인해 발생할 수 있는 문제점입니다.

예 1

이 예에서 쉘은 readdir()이 리턴하는 값에 대한 총칭명 접두부의 문자 비교를 수행합니다. 그러나 QSYS.LIB 파일 시스템은 대문자로 된 디렉토리 항목을 리턴하므로 어떤 항목도 소문자의 총칭명 접두부와 일치하지 않습니다.

```
$ 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
```

예 2

이 예는 첫 번째 예와 유사하지만, 이 예에서는 쉘 대신 find 유틸리티가 비교를 수행합니다.

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

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

예 3

ps 유틸리티는 사용자 이름의 대소문자를 구분하므로 -u 옵션에 대해 지정된 대문자 이름과 i5/OS PASE 런타임 함수 getpwuid()가 리턴한 소문자 이름 사이의 일치를 인식하지 않습니다.

```
$ 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
```

관련 정보

File system comparison

통합 파일 시스템 파일의 행 종료 문자

AIX 및 i5/OS은 텍스트 파일(예: 파일 및 쉘 스크립트)에서 서로 다른 행 종료 문자를 사용합니다.

i5/OS PASE 프로그램의 소스인 AIX 어플리케이션에서는 행(예; 파일 및 쉘 스크립트의 행)이 라인 피드(LF)로 끝나야 합니다. 그러나 PC 소프트웨어 및 일반 i5/OS 소프트웨어에서는 종종 캐리지 리턴 및 라인 피드(CRLF)로 끝납니다.

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

FTP에 사용된 CRLF

이러한 차이로 인해 문제가 발생할 수 있는 한 가지 예는 FTP를 사용하여 소스 파일 및 쉘 스크립트를 AIX에서 iSeries로 전송할 때입니다. 표준 FTP는 텍스트 모드에서 송신된 자료를 호출하여 행의 끝에서 캐리지 리턴 및 라인 피드(CRLF)를 사용합니다. AIX에서 FTP 유틸리티는 텍스트 모드에서 인바운드 파일을 처리할 때 캐리지 리턴(CR)을 제거합니다. i5/OS FTP는 자료 스트림에 나타난 사항을 항상 정확하게 기록하고 텍스트 모드에 대해 CRLF를 항상 보유하여 i5/OS PASE 런타임 및 유틸리티에 문제를 일으킵니다.

가능하면 AIX 오퍼레이팅 시스템에서 2진 모드 전송을 사용하여 이러한 문제를 방지하십시오. 대부분의 경우 퍼스널 컴퓨터에서 전송된 텍스트 파일은 CRLF 구분 행이 있습니다. 먼저 파일을 AIX로 전송하면 문제가 해결될 수 있습니다. 다음 명령을 사용하여 현재 디렉토리에 있는 파일에서 CR을 제거할 수 있습니다.

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

iSeries 및 PC 편집기에 사용된 CRLF

또한 iSeries 서버에서 편집기를 사용하거나 워크스테이션에서 편집기(예: Windows® 메모장 편집기)를 사용하여 파일 또는 쉘 스크립트를 편집할 경우에도 문제가 발생할 수 있습니다. 이들 편집기는 새 행 분리자로 CRLF를 사용하고 i5/OS PASE가 예상하는 LF를 사용하지 않습니다.

CRLF를 새 행 분리자로 사용하지 않는 여러 편집기(예: `ez` 편집기)를 사용할 수 있습니다.

관련 정보

하드웨어에 대한 IBM Virtual Innovation Center

파일 전송

이 주제에서 설명하는 세 가지 방법 중 하나로 i5/OS PASE 프로그램 및 관련 파일을 iSeries 서버와 송수신할 수 있습니다.

- 파일 전송 프로토콜(FTP)을 사용하여 프로그램 복사
- 서버 메세지 블록(SMB)을 사용하여 프로그램 복사
- 리모트 파일 시스템을 사용하여 프로그램 복사

파일 전송 프로토콜(FTP)을 사용하여 프로그램 복사

i5/OS 파일 전송 프로토콜(FTP) 디먼 및 클라이언트를 사용하여 파일을 i5/OS 통합 파일 시스템과 송수신할 수 있습니다. 2진 모드로 파일을 전송하십시오. FTP 부속 명령 `binary`를 사용하여 이 모드를 설정하십시오.

통합 파일 시스템에 파일을 넣을 때 명명 형식 1(i5/OS FTP 명령의 `NAMEFMT 1` 부속 명령)을 사용해야 합니다. 이 형식을 사용하면 경로명을 사용하고 파일을 스트림 파일로 전송할 수 있습니다. 명명 형식 1로 들어가려면 다음 중 하나를 수행하십시오.

- 경로명을 사용하여 디렉토리를 변경하십시오.

  그러면 세션이 자동으로 이름 형식 1에 놓입니다. 이 방법을 사용할 경우 첫 번째 디렉토리 앞에 슬래시(/)가 붙습니다. 예를 들면 다음과 같습니다.

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

- 리모트 클라이언트에 FTP 부속 명령 `quote site namefmt 1`을 사용하거나 `namefmt 1`을 로컬 클라이언트로 사용하십시오.

서버 메세지 블록(SMB)을 사용하여 프로그램 복사

i5/OS는 서버 메세지 블록(SMB) 클라이언트 및 서버 구성요소를 지원합니다. NetServer™가 구성되어 실행 중일 때 i5/OS PASE는 /QNTC 파일 시스템을 통해 네트워크에 있는 SMB 서버에 액세스할 수 있습니다. AIX 또는 Linux 플랫폼에서 동일한 서비스를 제공하려면 SAMBA 서버가 필요합니다. AIX와 같이 구성되고 작동 가능한 시스템을 설치하면 i5/OS PASE에서 디렉토리 및 파일을 사용할 수 있습니다.

리모트 파일 시스템을 사용하여 프로그램 복사

i5/OS에서는 네트워크 파일 시스템(NFS)의 파일 시스템을 통합 파일 시스템 파일 공간의 마운트 지점에 마운트할 수 있습니다. AIX는 분산 파일 시스템(DFS™) 및 Andrew 파일 시스템(AFS®)과 함께 NFS를 지원하므로(DFS에서 NFS로 및 AFS에서 NFS로 변환 프로그램을 사용하여) i5/OS가 이들 파일 시스템을 내보내고 마운트할 수 있도록 합니다. 그런 다음, i5/OS PASE 어플리케이션이 이들 파일 시스템을 사용할 수 있습니다. 액세스 중인 디렉토리 경로 또는 파일에 대한 보안 권한은 i5/OS 사용자 프로파일의 사용자 ID 번호와 그룹 ID 번호를 통해 검증됩니다. 여러 플랫폼에서 같은 사람으로 인식되는 사용자 프로파일은 모든 시스템에서 같은 사용자 ID를 가져야 합니다.

i5/OS는 NFS 서버로 사용하는 것이 가장 효과적입니다. 이 경우 AIX 시스템에서 i5/OS 통합 파일 시스템의 디렉토리로 마운트하고 AIX는 프로그램이 빌드되면 이를 바로 i5/OS에 기록해야 합니다.

주: i5/OS NFS는 현재 멀티스레드 어플리케이션에서 지원되지 않습니다.

관련 정보

FTP

i5/OS 함수를 사용하도록 i5/OS PASE 프로그램 사용자 정의

AIX 어플리케이션에서 시스템 제공 i5/OS PASE 공유 라이브러리가 직접 지원하지 않는 i5/OS 함수를 사용하려면 어플리케이션을 준비하기 위한 몇 가지 추가 단계를 수행해야 합니다.

준비를 수행하려면 다음 단계를 완료하십시오.

1. i5/OS 시스템 고유 함수에 대한 액세스를 조정하는 필수 i5/OS PASE 런타임 함수를 호출하는 AIX 어플리케이션을 코딩하십시오.
2. AIX 시스템에서 i5/OS PASE 프로그램을 컴파일하는 경우 사용자 정의된 어플리케이션을 컴파일하기 전에 다음 단계를 수행하십시오.
   a. 필수 i5/OS 시스템 고유 헤더 파일을 AIX 시스템으로 복사하십시오.
   b. 필수 i5/OS 시스템 고유 헤더 파일을 AIX 시스템으로 복사하십시오.

   관련 개념

   34 페이지의『i5/OS PASE 프로그램에서 i5/OS 프로그램과 프로시듀어 호출』
   i5/OS PASE는 i5/OS 기능에 대한 통합 액세스를 제공하는 ILE 프로시듀어, Java 프로그램, OPM 프로그램, i5/OS API 및 CL 명령을 호출하는 메소드를 제공합니다.

   47 페이지의『i5/OS PASE 프로그램과 i5/OS의 대화 방법』
   i5/OS 기능을 사용하도록 i5/OS PASE 프로그램을 사용자 정의할 때 프로그램이 이들과 대화하는 방법을 고려해야 합니다.

   관련 정보

   Runtime functions for use by i5/OS PASE programs

헤더 파일 복사

이 주제의 지침을 수행하여 iSeries 서버에서 AIX 기계로 헤더 파일을 복사할 수 있습니다.

i5/OS PASE는 i5/OS 시스템 고유 지원에 대한 헤더 파일로 표준 AIX 런타임을 증가시킵니다. 이들 헤더 파일은 i5/OS PASE 및 i5/OS 오퍼레이팅 시스템에서 제공됩니다.

iSeries 서버에서 헤더 파일 검색 경로에 있는 AIX 기계로 헤더 파일을 복사하십시오.

헤더 파일을 /usr/include AIX 디렉토리 또는 컴파일러에 대한 헤더 파일 검색 경로에 있는 다른 디렉토리로 복사할 수 있습니다.

/usr/include 외의 다른 디렉토리를 사용할 경우 AIX 컴파일러 명령에 -I 옵션을 사용하여 헤더 파일 검색 경로에 이 디렉토리를 추가할 수 있습니다.

i5/OS PASE 헤더 파일 복사

i5/OS PASE 헤더 파일은 다음 i5/OS 디렉토리에 있습니다.

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

i5/OS PASE는 다음 헤더 파일을 제공합니다.

헤더 파일	설명
as400_protos.h	이 헤더 파일은 기타 i5/OS PASE 시스템 고유 기능을 ILE에 제공합니다.
as400_types.h	이 헤더 파일은 ILE에 대한 호출에 대해 고유 i5/OS 매개변수 유형을 선언합니다. 이 헤더 파일은 16바이트 기계 인터페이스(MI) 포인터에 대해 128비트 필드가 되는 long double 유형을 사용하는 `ILEpointer` 유형을 선언합니다. as400_types.h에 선언된 기타 유형은 유형 long long에서 64비트 정수가 됩니다. as400_types.h에 적절한 크기 및 정렬 유형이 선언되었는지 확인하려면 AIX 컴파일러를 `-qlngdbl128`, `-qalign=natural` 및 `-qlonglong` 옵션으로 실행해야 합니다.
os400msg.h	이 헤더 파일은 i5/OS 메세지를 송수신하기 위한 함수를 선언합니다.

i5/OS 헤더 파일 복사

i5/OS PASE 어플리케이션의 다른 i5/OS 함수에 액세스하려면 사용 중인 i5/OS 함수의 헤더 파일을 사용자의 개발 기계에 복사하는 것이 좋을 수 있습니다. 일반적으로 i5/OS 프로그램 또는 프로시듀어는 i5/OS PASE 어플리케이션에서 직접 실행할 수 있다는 점을 참고하십시오. 자세한 정보는 i5/OS PASE 프로그램에서 i5/OS 프로그램 및 프로시듀어 호출을 참조하십시오.

i5/OS 시스템 제공 헤더 파일은 /QIBM/include 디렉토리에 있습니다.

어플리케이션에 i5/OS API 헤더 파일이 필요한 경우, 변환된 파일을 AIX 디렉토리에 복사하기 전에 먼저 이 헤더 파일을 EBCDIC에서 ASCII로 변환해야 합니다.

EBCDIC 텍스트 파일을 ASCII로 변환하는 한 가지 방법은 i5/OS PASE `Rfile` 유틸리티를 사용하는 것입니다.

다음 예에서는 i5/OS PASE `Rfile` 유틸리티를 사용하여 i5/OS 헤더 파일 /QIBM/include/qusec.h를 읽고 해당 자료를 i5/OS PASE CCSID(코드화 문자 세트 ID)로 변환한 후, 각 행에서 뒤에 있는 공백을 제거한 다음 그 결과를 바이트 스트림 파일 `ascii_qusec.h`에 씁니다.

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

관련 개념

48 페이지의『데이터베이스』

i5/OS PASE는 iSeries용 DB2® UDB CLI(Call Level Interface)를 지원합니다. AIX 및 i5/OS의 DB2 CLI는 서로에 대해 적절한 서브세트가 아니므로 일부 인터페이스에 약간의 차이점이 있고 구현된 일부 API가 다른 시스템에는 없을 수 있습니다.

34 페이지의『i5/OS PASE 프로그램에서 i5/OS 프로그램과 프로시듀어 호출』

i5/OS PASE는 i5/OS 기능에 대한 통합 액세스를 제공하는 ILE 프로시듀어, Java 프로그램, OPM 프로그램, i5/OS API 및 CL 명령을 호출하는 메소드를 제공합니다.

관련 태스크

35 페이지의『ILE 프로시듀어 호출』
이 주제의 지침을 수행하여 i5/OS PASE 프로그램에서 ILE 프로시듀어를 준비하고 호출할 수 있습니다.

관련 참조

15 페이지의『i5/OS PASE 프로그램을 iSeries 서버에 복사』
i5/OS PASE에서 실행하려는 AIX 2진을 통합 파일 시스템에 복사하십시오.

내보내기 파일 복사

이 주제의 지침을 수행하여 iSeries 서버에서 AIX 디렉토리로 내보내기 파일을 복사할 수 있습니다.

i5/OS 시스템 고유 함수에 액세스할 필요가 있는 어플리케이션을 빌드하려면 다음 i5/OS 디렉토리에 있는 내보내기 파일을 사용하는 것이 좋습니다.

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

이러한 파일을 AIX 디렉토리에 복사할 수 있습니다. AIX `ld` 명령(또는 컴파일러 명령)에서 `-bI:` 옵션을 사용하여 AIX 시스템의 공유 라이브러리에 없는 기호를 정의하십시오.

i5/OS PASE는 다음 내보내기 파일을 제공합니다.

내보내기 파일	함수
as400_libc.exp	이 파일은 libc.a에 있는 i5/OS 시스템 고유 함수에 대한 내보내기 파일입니다. as400_libc.exp 파일은 해당 라이브러리의 AIX 버전에서 내보내지 않은 libc.a의 i5/OS PASE 버전으로부터의 모든 내보내기를 정의합니다.
libdb400.exp	이 파일은 i5/OS 데이터베이스 함수에 대한 내보내기 파일입니다. libdb400.exp 파일은 i5/OS PASE libdb400.a 라이브러리로부터의 내보내기를 정의합니다(iSeries용 DB2 UDB 호출 레벨 인터페이스(CLI) 지원).

관련 개념

48 페이지의『데이터베이스』
i5/OS PASE는 iSeries용 DB2 UDB CLI(Call Level Interface)를 지원합니다. AIX 및 i5/OS의 DB2 CLI는 서로에 대해 적절한 서브세트가 아니므로 일부 인터페이스에 약간의 차이점이 있고 구현된 일부 API가 다른 시스템에는 없을 수 있습니다.

관련 참조

15 페이지의『i5/OS PASE 프로그램을 iSeries 서버에 복사』
i5/OS PASE에서 실행하려는 AIX 2진을 통합 파일 시스템에 복사하십시오.

i5/OS 기능에 액세스하기 위한 i5/OS PASE API

i5/OS PASE는 ILE 코드 및 기타 i5/OS 기능에 액세스하기 위한 다양한 API를 제공합니다. 컴파일러가 수행하는 것과 비교하여 사용자 스스로의 준비와 구조 빌드를 수행하려는 정도에 따라 사용할 API를 선택할 수 있습니다.

관련 정보

i5/OS 환경에서 i5/OS PASE 프로그램 사용

i5/OS PASE 프로그램은 작업에서 실행하는 다른 i5/OS 프로그램을 호출할 수 있으며, 다른 i5/OS 프로그램은 i5/OS PASE 프로그램의 프로시듀어를 호출할 수 있습니다.

i5/OS PASE 프로그램 및 프로시듀어 실행

이 주제에서는 작업에서 i5/OS PASE 프로그램을 시작하고 ILE 프로그램에서 i5/OS PASE 프로시듀어 호출에 대한 정보 및 예를 읽을 수 있습니다.

i5/OS PASE 프로그램은 여러 가지 방법으로 실행할 수 있습니다.

- i5/OS 작업에서
- i5/OS PASE 대화식 쉘 환경에서
- ILE 프로시듀어에서 호출된 프로그램으로서

주: i5/OS에서 i5/OS PASE 프로그램을 실행할 때 i5/OS PASE 환경 변수가 ILE 환경 변수와 독립적인 점을 유의하십시오. 한 환경에서 변수를 설정해도 다른 환경에 영향이 미치지 않습니다.

i5/OS PASE 프로그램에 대해 작업하는 데 사용할 수 있는 ILE 프로시듀어

i5/OS PASE는 많은 ILE 프로시듀어 API를 제공하며, ILE 코드는 이를 통해 i5/OS PASE 서비스에 액세스할 수 있습니다(i5/OS PASE 프로그램의 특수 프로그래밍을 사용하지 않고).

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

ILE 스레드에 접속

i5/OS PASE로 작성하지 않은 스레드(예: Java 스레드 또는 ILE pthread_create로 작성한 스레드)에서 실행되는 ILE 코드에서 i5/OS PASE 프로그램에 있는 프로시듀어를 호출할 수 있습니다. Qp2CallPase는 ILE 스레드를 i5/OS PASE에 자동으로 접속시키지만(해당 i5/OS PASE pthread 구조 작성), i5/OS PASE 프로그램이 시작될 때 i5/OS PASE 환경 변수 PASE_THREAD_ATTACH가 Y로 설정된 경우에만 가능합니다.

i5/OS PASE to i5/OS 프로그램에서 결과 리턴

i5/OS _RETURN() 함수를 사용하여 i5/OS PASE 프로그램을 호출하고 i5/OS PASE 환경을 종료하지 않고 결과를 리턴할 수 있습니다. 이를 통해 i5/OS PASE 프로그램을 시작한 다음, QP2SHELL2(QP2SHELL이 아님) 또는 Qp2RunPase API가 리턴된 후 Qp2CallPase를 사용하여 이 프로그램에 있는 프로시듀어를 호출할 수 있습니다.

관련 개념

33 페이지의『환경 변수에 대한 작업』

i5/OS PASE 환경 변수는 ILE 환경 변수와 독립적입니다. 한 환경에서 변수를 설정하여도 다른 환경에 영향이 미치지 않습니다.

관련 정보

i5/OS PASE ILE Procedure APIs

_RETURN()--Return Without Exiting i5/OS PASE

QP2SHELL()을 사용하여 i5/OS PASE 프로그램 실행

QP2SHELL 또는 QP2SHELL2 프로그램을 사용하여 i5/OS 명령행과 고급 레벨 언어 프로그램, 일괄처리 작업 또는 대화식 작업에서 i5/OS PASE 프로그램을 실행할 수 있습니다.

이들 프로그램은 i5/OS PASE 프로그램을 이를 호출하는 작업에서 실행합니다. i5/OS PASE 프로그램의 이름은 프로그램의 한 매개변수로서 전달됩니다.

QP2SHELL() 프로그램은 새 활성 그룹에서 i5/OS PASE 프로그램을 실행합니다. QP2SHELL2() 프로그램은 호출자의 활성 그룹에서 실행됩니다.

다음 예에서는 i5/OS 명령행에서 `ls` 명령을 실행합니다.

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

CL 변수를 사용하여 QP2SHELL()에 값을 전달하는 경우 반드시 변수를 널(null)로 종료해야 합니다. 예를 들어 위의 예는 다음과 같은 방법으로 코딩해야 합니다.

```
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
```

관련 정보

QP2SHELL() and QP2SHELL2()--Run an i5/OS PASE Shell Program

QP2TERM()을 사용하여 i5/OS PASE 프로그램 실행

이 i5/OS 프로그램을 사용하여 i5/OS PASE 프로그램을 대화식 쉘 환경을 실행합니다.

QP2TERM() 프로그램에서 i5/OS PASE 대화식 단말기 세션을 시작하십시오.

다음 명령은 디폴트 Korn 쉘 프롬프트 (/QOpenSys/usr/bin/sh)를 화면에 기록합니다.

```
call qp2term
```

이 프롬프트에서는 i5/OS PASE 프로그램을 별도의 일괄처리 작업으로 실행합니다. QP2TERM()은 대화식 작업을 사용하여 출력을 표시하고 파일 stdin, stdout 및 stderr에 대한 입력을 일괄처리 작업으로 승인합니다.

Korn 쉘이 디폴트이지만 프로그램으로 전달할 인수 스트링 뿐만 아니라 실행할 i5/OS PASE 프로그램의 경로명도 선택적으로 지정할 수 있습니다.

QP2TERM()으로 시작하는 대화식 세션에서 i5/OS PASE 프로그램 및 유틸리티를 실행할 수 있으며 stdout 및 stderr은 단말기 화면에 쓰여지고 화면이동됩니다.

관련 정보

QP2TERM()--Run an i5/OS PASE Terminal Session

i5/OS PASE shells and utilities

i5/OS 프로그램에서 i5/OS PASE 프로그램 실행

이 주제의 단계를 수행하여 다른 ILE 프로시듀어에서 Qp2CallPase() 및 Qp2CallPase2() ILE 프로시듀어를 호출하여 i5/OS PASE 프로그램을 시작하고 실행할 수 있습니다. 예는 다음과 같습니다.

Qp2RunPase() API를 사용하여 i5/OS PASE 프로그램을 실행하십시오. 프로그램명, 인수 스트링 및 환경 변수를 지정하십시오.

Qp2RunPase() API가 i5/OS PASE 프로그램을 호출한 작업에서 이 프로그램을 실행합니다. i5/OS PASE 프로그램(필요한 공유 라이브러리 포함)을 로드한 후 이 프로그램으로 제어를 전송합니다.

이 API는 QP2SHELL() 및 QP2TERM() 보다 i5/OS PASE가 실행되는 방법에 대해 더 많은 제어를 제공합니다.

관련 정보

Qp2RunPase()--Run an i5/OS PASE Program

예: i5/OS 프로그램에서 i5/OS PASE 프로그램을 실행하십시오.:

이 주제에 설명된 예는 i5/OS PASE 프로그램을 호출하는 ILE 프로그램 및 ILE 프로그램이 호출하는i5/OS PASE 프로그램을 보여줍니다.

주: 코드 예제를 사용하여 66 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의할 수 있습니다.

예 1: i5/OS PASE 프로그램을 호출하는 ILE 프로그램

다음 ILE 프로그램은 i5/OS PASE 프로그램을 호출합니다. 이 예 다음에는 이 프로그램이 호출하는 i5/OS PASE 코드의 예가 나와 있습니다.

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

/* QP2RunPase()에 대한 포함 파일. */

#include <qp2user.h>

/*****************************************
    샘플:
  QP2RunPase()를 사용하고 하나의 스트링
  매개변수를 전달하여 i5/OS PASE 프로그램을
  호출하는 간단한 ILE C 프로그램.
    컴파일 예:
        CRTCMOD MODULE(MYLIB/SAMPLEILE) SRCFILE(MYLIB/QCSRC)
        CRTPGM PGM(MYLIB/SAMPLEILE)
*****************************************/

void main(int argc, char*argv[])
 {
    /* PASE 프로그램의 경로명 */
    char *PasePath = "/home/samplePASE";
    /* QP2RunPase()의 리턴 코드 */
    int rc;
    /* i5/400 PASE 프로그램에 전달되는
     매개변수 */
    char *PASE_parm = "My Parm";
  /* i5/OS PASE 프로그램에 대한 인수 리스트로
         포인터 리스트에 대한 포인터입니다. */
    char **arg_list;
    /* 인수 리스트를 할당하십시오. */
    arg_list =(char**)malloc(3 * sizeof(*arg_list));
    /* 프로그램명을 첫 번째 요소로 설정하십시오. 이것은 UNIX 규약입니다. */
    arg_list[0] = PasePath;
    /* 매개변수를 첫 번째 요소로 설정하십시오. */
    arg_list[1] = PASE_parm;
    /* 인수 리스트의 마지막 요소는 반드시 널(null)이어야 합니다. */
    arg_list[2] = 0;
  /* i5/OS PASE 프로그램 호출. */
      rc = Qp2RunPase(PasePath, /* 경로명 */
            NULL,           /* ILE를 호출하기 위한 기호. 이 샘플에서는 사용되지 않음 */
            NULL,           /* ILE 호출에 대한 기호 자료. 여기서는 사용되지 않음 */
            0,              /* ILE 호출에 대한 기호 자료. 여기서는 사용되지 않음 */
      819,            /* i5/OS PASE에 대한 ASCII CCSID */
      arg_list,       /* i5/OS PASE 프로그램에 대한 인수 */
            NULL);          /* 환경 변수 리스트. 이 샘플에서는 사용되지 않음 */
 }
```

예 2: ILE 프로그램에서 호출되는 i5/OS PASE 프로그램

다음 i5/OS PASE 프로그램은 위의 ILE 프로그램에 의해 호출됩니다.

```
#include <stdio.h>

/*****************************************
    샘플:
  QP2RunPase()를 사용하고 하나의 스트링
    매개변수를 채택하여 ILE에서 호출한
    간단한 i5/OS PASE 프로그램.
    ILE 샘플 프로그램은 이 프로그램이
    /home/samplePASE에 있는 것으로 압니다.
  AIX에서 컴파일한 후 i5/OS으로 ftp 전송하십시오.
    ftp 전송을 하려면 다음 명령을 사용하십시오.
  > binary
  > site namefmt 1
  > put samplePASE /home/samplePASE
*****************************************/

int main (int argc, char *argv[])
{
        /* 전달된 인사와 매개변수를 인쇄하십시오. argv[0]가 프로그램 이름이므로
              argv[1]은 매개변수입니다. */
     printf("Hello from i5/OS PASE program %s. Parameter value is ₩"%s₩".₩n", argv[0], argv[1]);

       return 0;
}
```

i5/OS 프로그램에서 i5/OS PASE 프로시듀어 호출

다른 ILE 프로시듀어에서 Qp2CallPase() 및 Qp2CallPase2() ILE 프로시듀어를 호출하여 i5/OS PASE 환경이 이미 실행 중인 작업에서 i5/OS PASE 프로그램을 실행할 수 있습니다.

Qp2RunPase() API가 먼저 시작되어 작업에서 i5/OS PASE 프로그램을 실행합니다. 이 작업에서 i5/OS PASE가 이미 사용 중이면 오류를 리턴합니다.

i5/OS PASE 프로그램을 이미 실행 중인 작업에서 i5/OS PASE 프로시듀어를 호출하려면 Qp2CallPase() 및 Qp2CallPase2() API를 사용하십시오.

관련 정보

Qp2CallPase()--Call an i5/OS PASE Procedure

예 1: i5/OS 프로그램에서 i5/OS PASE 프로시듀어 호출:

이 주제의 예는 i5/OS PASE 프로시듀어를 호출하는 ILE 프로그램을 보여줍니다.

주: 코드 예제를 사용하여 66 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의할 수 있습니다.

```
#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;

    /*
     * QP2SHELL2를 호출하여 i5/OS PASE 프로그램
     * /usr/lib/start32를 실행하십시오. 이는 i5/OS PASE를
          * 32비트 모드로 시작합니다. (리턴 시 활동 상태를 유지합니다.)
     */
        QP2SHELL2("/usr/lib/start32");

    /*
          * Qp2dlopen은 첫 번째 인수가 널(null) 포인터일
          * 경우 글로벌명 공간을 엽니다(새 공유 실행 파일을
          * 로드하지 않음). Qp2dlsym은 i5/OS PASE getpid
          * 서브루틴(공유 라이브러리 libc.a에서 내보냄)을
          * 찾습니다.
     */
        id = Qp2dlopen(NULL, QP2_RTLD_NOW, JOB_CCSID);
        getpid_pase = Qp2dlsym(id, "getpid", JOB_CCSID, NULL);

    /*
     * Qp2CallPase를 호출하여 i5/OS PASE getpid
     * 함수를 실행하고 결과를 인쇄하십시오. 함수 결과가
     * -1인 경우 Qp2errnop를 사용하여 i5/OS PASE errno를
     * 찾아 인쇄하십시오.
     */
        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());

    /*
          *  Qp2dlopen 인스턴스를 닫고 Qp2EndPase를
     *  호출하여 이 작업에서 i5/OS PASE를 종료하십시오.
     */
        Qp2dlclose(id);
        Qp2EndPase();
      return 0;
}
```

예 2: i5/OS PASE 프로시듀어에 대한 호출에서 포인터 인수를 사용하는 i5/OS ILE 프로그램:

이 예에서 i5/OS ILE 프로그램은 호출되는 i5/OS PASE 프로시듀어를 통해 메모리 기억장치를 할당 및 공유하는 데 두 가지 서로 다른 기법을 사용합니다.

주: 다음 코드 예제를 사용하여 66 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의할 수 있습니다.

```
/* 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 */
}
```

예 2의 ILE 부분에 사용되는 여러 가지 함수

- **startPASE() 함수**

  i5/OS PASE를 프로세스에 사용하려면 먼저 이를 시작해야 합니다. QP2SHELL, QP2TERM 또는 Qp2RunPase와 같은 API를 사용하여 i5/OS PASE 어플리케이션 기본 진입점을 호출하여 자동으로 시작할 수 있습니다.

  그러나 이 예는 공유 라이브러리에서 내보낸 i5/OS PASE 함수를 호출하므로(기본 진입점을 사용하지 않고) i5/OS PASE를 수동으로 시작해야 합니다. 이를 위해 두 개의 i5/OS PASE 시작 프로그램 유틸리티를 사용할 수 있습니다. i5/OS PASE의 32비트 버전을 시작할 경우 /usr/lib/start32, i5/OS PASE의 64비트 버전을 시작할 경우 /usr/lib/start64를 사용할 수 있습니다.

  각 i5/OS 프로세스에서는 i5/OS PASE의 단일 인스턴스만 실행할 수 있다는 점을 유의하십시오. Qp2ptrsize() API는 i5/OS PASE가 이미 실행 중인지 판별하는 데 사용할 수 있습니다.
  - i5/OS PASE가 프로세스에서 현재 활동하지 않을 경우 Qp2ptrsize()는 0을 리턴합니다.
  - i5/OS PASE가 32비트 모드로 활동할 경우 Qp2ptrsize()는 4를 리턴합니다.
  - i5/OS PASE가 64비트 모드로 활동할 경우 Qp2ptrsize()는 8을 리턴합니다.
- **openlib() 및 findsym() 함수**

  이들 함수는 i5/OS 공유 라이브러리를 열고 사용자가 Qp2dlopen() 및 Qp2dlsym()을 사용하여 호출하려는 함수에 대한 포인터를 확보합니다. 이들 함수는 여러 플랫폼에서의 dlopen() 및 dlsym() 루틴과 유사합니다.
- **Qp2CallPase2 호출에 대한 인수 설정**

  callPASE() 함수를 통해 Qp2CallPase2()를 호출하기 전에 main() 루틴은 ILE와 i5/OS PASE 함수 간의 인터페이스를 정의하는 다음 변수를 설정합니다.
  - 서명 배열 변수는 i5/OS PASE 함수에 대한 인수를 정의합니다. 이 배열의 요소는 일반적으로 qsysinc/h.qp2user 포함 파일에 있는 #define을 사용하여 설정됩니다.
  - paseFunctionArglist 구조는 i5/OS PASE 런타임이 함수 호출 시 i5/OS PASE 함수로 전달될 인수로 맵핑할 ILE 변수를 포함합니다. paseFunctionArglist의 멤버는 서명 배열에 선언된 i5/OS PASE 함수의 서명에 해당합니다.
  - inputAndResultBuffer 구조는 i5/OS PASE 런타임이 함수 호출 시 ILE와 i5/OS PASE 함수 사이에 일종의 공유 버퍼로 사용할 ILE 변수를 포함합니다.

이 구조의 첫 번째 멤버(이 예에서의 결과)는 i5/OS PASE 함수의 리턴 값을 포함합니다. 이 변수는 Qp2CallPase2 API 호출에서 네 번째 인수로서 제공되는 result-type 인수와 일치해야 합니다. 이 첫 번째 요소 다음부터는 함수 호출 시 i5/OS PASE 환경으로 복사될 기억장치를 나타냅니다.

이 예에서 inputAndResultBuffer 구조의 inputValue 요소는 i5/OS PASE 함수에 대한 첫 번째 인수가 지정하는 by-address 인수 데이터를 포함합니다.

– 이 예는 호출되는 i5/OS PASE 함수에 대한 포인터 인수를 두 가지 다른 방법으로 설정합니다.

- paseFunctionArglist.outputPasePtr 함수에 대한 두 번째 인수는 Qp2malloc() 함수를 호출하여 설정합니다. Qp2malloc()는 i5/OS PASE 런타임 힙(heap)으로부터 메모리를 할당한 후 이 할당된 기억장치로 ILE 포인터 및 i5/OS PASE 포인터를 모두 리턴합니다.
- 첫 번째 인수인 paseFunctionArglist.inputPasePtr은 OR에 의해 qp2user.h #define QP2_ARG_PTR_TOSTACK과 연결된 inputAndResultBuffer 구조의 inputValue 요소의 오프셋으로 설정됩니다.

  이는 i5/OS PASE 런타임에게 i5/OS PASE 메모리로 inputAndResultBuffer.inputValue가 복사된 주소로 i5/OS PASE 함수에 대한 호출에 제공된 실제 포인터 값을 수정하도록 지시합니다.

- **callPASE() 함수**

  이 함수는 Qp2CallPase2() API 및 main() 루틴에 설정된 인수를 사용하여 i5/OS PASE 함수를 호출합니다.

- **프로세스에서 i5/OS PASE 종료**

  i5/OS PASE 함수를 호출한 후, Qp2dlclose() API를 호출하여 i5/OS PASE 공유 라이브러리를 언로드하고 Qp2EndPase()를 호출하여 이 예를 시작할 때 호출한 start64 프로그램을 종료합니다.

Java에서 i5/OS PASE 원시 메소드 사용

Java 프로그램에서 i5/OS PASE 환경에서 실행되는 i5/OS PASE 원시 메소드를 사용할 수 있습니다.

i5/OS PASE 원시 메소드에 대한 지원은 i5/OS PASE 원시 메소드에서 원시 iSeries JNI(Java Native Interface)의 전체 사용과 원시 iSeries JVM에서 i5/OS PASE 원시 메소드를 호출할 수 있는 기능을 포함합니다.

관련 정보

IBM i5/OS PASE native methods for Java

환경 변수에 대한 작업

i5/OS PASE 환경 변수는 ILE 환경 변수와 독립적입니다. 한 환경에서 변수를 설정하여도 다른 환경에 영향이 미치지 않습니다.

그러나 i5/OS PASE 프로그램을 실행하는 데 사용하는 메소드에 따라 ILE에서 i5/OS PASE로 변수를 복사할 수 있습니다.

대화식 i5/OS PASE 세션의 환경 변수

ILE 환경 변수는 QP2SHELL() 및 QP2TERM()으로 시작될 때만 i5/OS PASE로 전달됩니다. WRKENVVAR(환경 변수에 대한 작업) 명령을 사용하여 i5/OS PASE를 시작하기 전에 필요에 따라 환경 변수를 변경, 추가 또는 삭제하십시오.

호출된 i5/OS PASE 세션의 환경 변수

i5/OS PASE가 프로그램 호출에서 시작될 때(Qp2RunPase() API를 사용하여) 환경 변수에 대한 완전한 제어를 할 수 있습니다. i5/OS PASE 프로그램을 호출한 ILE 환경과 아무 관계가 없는 환경 변수를 전달할 수 있습니다.

CL 명령을 실행하기 전에 환경 변수를 ILE로 복사

systemCL() 런타임 함수의 옵션을 사용하여 CL 명령을 실행하기 전에 i5/OS PASE 환경 변수를 ILE 환경으로 복사할 수 있습니다. 이는 또한 i5/OS PASE `system` 유틸리티의 디폴트 작동이기도 합니다.

관련 참조

22 페이지의『i5/OS PASE 프로그램 및 프로시듀어 실행』
이 주제에서는 작업에서 i5/OS PASE 프로그램을 시작하고 ILE 프로그램에서 i5/OS PASE 프로시듀어 호출에 대한 정보 및 예를 읽을 수 있습니다.

관련 정보

QP2SHELL() and QP2SHELL2()--Run an i5/OS PASE Shell Program

QP2TERM()--Run an i5/OS PASE Terminal Session

systemCL()--Run a CL Command for i5/OS PASE

i5/OS PASE environment variables

i5/OS PASE 프로그램에서 i5/OS 프로그램과 프로시듀어 호출

i5/OS PASE는 i5/OS 기능에 대한 통합 액세스를 제공하는 ILE 프로시듀어, Java 프로그램, OPM 프로그램, i5/OS API 및 CL 명령을 호출하는 메소드를 제공합니다.

i5/OS 프로그램 및 프로시듀어에 대한 일반 구성 요구사항

i5/OS PASE 프로그램 환경에서 i5/OS 환경으로 호출할 때, 다음과 같은 이유 때문에 일반적으로 i5/OS 프로그램이 활성 그룹에 대해 *CALLER로 컴파일되었는지 확인해야 합니다.

- i5/OS PASE(Qp2RunPase API에 의해 호출됨)를 시작한 활성 그룹에서 실행하는 코드만 Qp2CallPase와 같은 ILE API를 사용하여 i5/OS PASE 프로그램과 대화할 수 있습니다.
- 멀티스레드 작업에서 활성 그룹을 제거해야 하는 경우(i5/OS PASE 포크(fork)에 의해 작성된 모든 작업에서 멀티스레드가 가능한 경우) ILE 런타임이 전체 작업을 종료할 수 있습니다(i5/OS PASE도 종료됨). ACTGRP(*CALLER)를 사용하면 종료하려는 시점까지 작업 종료를 지연할 수 있습니다.

systemCL() 런타임 함수를 사용하여 멀티스레드가 가능하지 않은 별도의 작업에서 CL 명령(CALL 명령 포함)을 실행함으로써 멀티스레드가 가능한 작업에서 실행되는 문제점을 방지할 수 있습니다.

관련 태스크

19 페이지의『i5/OS 함수를 사용하도록 i5/OS PASE 프로그램 사용자 정의』
AIX 어플리케이션에서 시스템 제공 i5/OS PASE 공유 라이브러리가 직접 지원하지 않는 i5/OS 함수를 사용하려면 어플리케이션을 준비하기 위한 몇 가지 추가 단계를 수행해야 합니다.

ILE 프로시듀어 호출

이 주제의 지침을 수행하여 i5/OS PASE 프로그램에서 ILE 프로시듀어를 준비하고 호출할 수 있습니다.

i5/OS PASE 프로그램에서 ILE 프로시듀어를 호출할 경우, 텍스트를 적절한 CCSID로 변환하고 변수 및 구조를 설정하여 먼저 teraspace에서 프로시듀어를 사용할 수 있도록 준비해야 합니다.

- **teraspace에서 ILE 프로시듀어 사용 가능**

  i5/OS PASE에서 호출하는 모든 ILE 모듈은 teraspace 옵션을 *YES로 설정하여 컴파일해야 합니다. ILE 모듈을 이 방법으로 컴파일하지 않으면, i5/OS PASE 어플리케이션에 대한 작업 기록부에 MCH4433 오류 메세지(목표 프로그램 &2에 대한 기억장치 모델이 유효하지 않음)가 수신됩니다.

- **적절한 CCSID로 텍스트 변환**

  ILE와 i5/OS PASE 간에 전달되는 텍스트는 전달되기 전에 적절한 CCSID로 변환할 필요가 있을 수 있습니다. 이러한 변환을 수행하지 않으면 문자 변수가 해독 불가능한 값을 포함할 수 있습니다.

- **변수 및 구조 설정**

  i5/OS PASE 프로그램에서 ILE를 호출하려면 변수 및 구조를 설정해야 합니다. 필수 헤더 파일을 AIX 시스템으로 복사했는지 확인하고 서명, 결과 유형 및 인수 리스트 변수를 설정해야 합니다.

  – **헤더 파일:** i5/OS PASE 프로그램에는 ILE 호출에 필요한 헤더 파일 as400_types.h 및 as400_protos.h가 있어야 합니다. as400_type.h 헤더 파일에는 i5/OS 시스템 고유 인터페이스에 사용되는 유형의 정의가 들어 있습니다.

  – **서명:** 서명 구조에는 i5/OS PASE와 ILE 간에 전달되는 인수의 유형 및 그 전달 순서에 대한 설명이 들어 있습니다. 호출 중인 ILE 프로시듀어가 요구하는 유형에 대한 코드화는 as400_types.h 헤더 파일에서 찾을 수 있습니다. 서명에 4바이트 미만의 고정 소수점 인수 또는 8바이트 미만의 부동 소수점 인수가 들어 있는 경우 다음의 pragma 인수를 사용하여 ILE C 코드를 컴파일해야 합니다.

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

    이 pragma가 없으면 ILE에 대한 표준 C 링크에서 1바이트 및 2바이트 정수 인수를 4바이트로 확장하고 4바이트 부동 소수점 인수를 8바이트로 확장해야 합니다.

  – **결과 유형:** 결과 유형은 간단하며 리턴 유형 C와 유사하게 작동합니다.

  – **인수 리스트:** 인수 리스트는 서명 배열의 항목에 지정된 유형 필드가 올바른 순서로 구성된 구조여야 합니다. size_ILEarglist() 및 build_ILEarglist() API를 사용하여 서명에 기초한 인수 리스트를 동적으로 빌드할 수 있습니다.

i5/OS PASE 프로그램에서 ILE 프로시듀어를 호출하려면 코드에서 다음 API 호출을 작성하십시오.

1. i5/OS PASE를 시작한 프로시듀어와 연관된 ILE 활성 그룹으로 바인드 프로그램을 로드하십시오. _ILELOAD() API를 사용하여 이를 수행할 수 있습니다.

   바인드 프로그램이 i5/OS PASE를 시작한 활성 그룹에서 이미 활성인 경우 이 단계를 수행하지 않아도 됩니다. 이런 경우 활성화 마크 매개변수에 지정된 0 값을 사용하여 현재 활성 그룹의 모든 활성 바인드 프로그램에서 모든 기호를 탐색하면 _ILESYM 단계로 진행할 수 있습니다.
2. ILE 바인드 프로그램의 활성화에서 내보낸 기호를 찾아 16바이트 태그 포인터를 기호에 대한 데이터나 프로시듀어로 리턴하십시오. _ILESYM() API를 사용하여 이를 수행할 수 있습니다.
3. i5/OS PASE 프로그램에서 ILE 프로시듀어로 제어를 전송하려면 해당 ILE 프로시듀어를 호출하십시오. _ILECALL() 또는 _ILECALLX() API를 사용하여 이를 수행할 수 있습니다.

   관련 참조

   19 페이지의『헤더 파일 복사』
   이 주제의 지침을 수행하여 iSeries 서버에서 AIX 기계로 헤더 파일을 복사할 수 있습니다.

   관련 정보

   size_ILEarglist()--Compute ILE Argument List Size for i5/OS PASE()

   build_ILEarglist()--Build an ILE Argument List for i5/OS PASE

   _ILELOADX()--Load an ILE Bound Program for i5/OS PASE

   _ILESYMX()--Find an Exported ILE Symbol for i5/OS PASE

   _ILECALLX()--Call an ILE Procedure for i5/OS PASE

   ILE 개념 PDF

예: ILE 프로시듀어 호출:

이 주제의 코드 예제는 서비스 프로그램의 일부인 ILE 프로시듀어 및 프로그램을 작성하는 컴파일러 명령을 호출하는 i5/OS PASE 코드를 보여줍니다.

서비스 프로그램의 일부인 ILE 프로시듀어 및 프로그램을 작성하는 컴파일러 명령을 호출하는 i5/OS PASE 코드를 보여주는 다음 코드 예제에는 두 개의 프로시듀어가 있습니다. 각 프로시듀어는 ILE 프로시듀어에 대한 여러 가지 작업 방식을 보여주지만 두 프로시듀어 모두 동일한 ILE 프로시듀어를 호출합니다. 첫 번째 프로시듀어는 i5/OS PASE 시스템 제공 메소드를 사용하여 _ILECALL API에 대한 자료 구조를 빌드하는 방법을 예시합니다. 그런 후 두 번째 프로시듀어는 인수 리스트를 수동으로 빌드합니다.

주: 코드 예제를 사용하여 66 페이지의『코드 라이센스 및 면책사항 정보』의 조건에 동의할 수 있습니다.

예 1: i5/OS PASE C 코드

다음의 코드 예제에는 코드를 설명하는 주석이 여러 곳에 나와 있습니다. 예를 입력하거나 검토할 때 이 주석을 반드시 읽어주십시오.

```
/* 이름: PASEtoILE.c
 *
  * 컴파일러 옵션 -qalign=natural 및 -qldbl128을 사용하여
  * 상대 16바이트 정렬 유형을 long double(유형 ILEpointer
  * 내부에서 사용됨)로 만들어야 합니다.
 *
 */
#include <stdlib.h>
#include <malloc.h>
#include <sys/types.h>
#include <stdio.h>
#include "as400_types.h"
#include "as400_protos.h"

/*
  * init_pid는 ILEtarget에서 주소 지정한 ILEpointer를
  * 추출한 프로세스의 프로세스 ID(PID)를 저장합니다.
  * init_pid는 이 프로그램의 exec() 이후에 나오는
  * 첫 번째 참조에서 초기화를 강제 수행하기 위한
  * 유효한 PID가 아닌 값으로 초기화됩니다.
 *
  * 사용자 코드가 pthread 인터페이스를 사용하는 경우
  * pthread_atfork()를 사용하여 등록된 핸들러를 제공하여
  * 하위 프로세스에서 ILE 프로시듀어 포인터를 다시
  * 초기화하고 정적 기억장치에서 포인터나 플래그를
  * 사용하여 exec() 이후 재초기화를 강제 수행할 수
  * 있습니다.
 */

pid_t  init_pid = -1;
ILEpointer*ILEtarget;  /* ILE 프로시듀어에 대한 포인터 */

/*
  * ROUND_QUAD는 지정된 주소나 그 주위에서 16바이트
  * 정렬 메모리 위치를 찾습니다.
 */

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

/*
  * do_init는 ILE 서비스 프로그램을 로드하고 해당
  * 서비스 프로그램에서 내보낸 프로시듀어로
  * ILE 포인터를 추출합니다.
 */

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

      /* _ILELOAD()는 서비스 프로그램을 로드합니다. */
      actmark = _ILELOAD("SHUPE/ILEPASE", ILELOAD_LIBOBJ);
      if (actmark == -1)
      abort();

   /*
        * xlc는 모든 유형의 정적 변수에 대해 16바이트
        * 정렬을 보장하지 않으므로 크기가 초과된
        * 버퍼에서 정렬된 영역을 찾습니다. _ILESYM()은
        * 서비스 프로그램 활성화에서 ILE 프로시듀어
```

```
        * 포인터를 추출합니다.
    */

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

   /*
        * 현재의 PID를 정적 기억장치에 저장하므로 (포크(fork))
        * 이후의 재초기화 시기를 판별할 수 있습니다.
    */
      init_pid = getpid();
}

/*
  * "aggregate"는 by-value 인수로 전달되는 구조나
  * 결합 자료 유형의 예입니다.
 */
typedef struct {
        char        filler[5];
} aggregate;

/*
  * "result_type" 및 "signature"는 ILEtarget에서
  * 식별하는 ILE 프로시듀어에 필요한 모든 인수의
  * 순서와 유형 및 함수 결과 유형을
  * 정의합니다.
 *
  * 주: 이 인수 리스트에 4바이트 미만의 고정 소수점
  * 인수 또는 8바이트 미만의 부동 소수점 인수가
  * 포함된다는 사실은 목표 ILE C 프로시듀어가
  * #pragma 인수(ileProcedureName, nowiden)를
  * 사용하여 컴파일됨을 의미합니다.
 *
  * 이 pragma가 없으면, ILE의 표준 C 연계에서
  * 1바이트 및 2바이트 정수 인수를 4바이트로
  * 확장해야 하고 4바이트 부동 소수점 인수를
  * 8바이트로 확장해야 합니다.
 */
static result_type_tresult_type = RESULT_INT32;
static arg_type_tsignature[] =
{
        ARG_INT32,
        ARG_MEMPTR,
        ARG_FLOAT64,
        ARG_UINT8,      /* ILE 코드에 #pragma nowiden이 있어야 합니다. */
        sizeof(aggregate),
        ARG_INT16,
        ARG_END
};

/*
  * wrapper_1은 자신이 호출하는 ILE 프로시듀어와 같은
  * 인수를 채택하여 같은 결과를 리턴합니다. 이 예에서는
  * ILE 인수 리스트에 대해 사용자 정의되거나 선언된
  * 구조를 필요로 하지 않습니다. 이 랩퍼는 malloc을
  * 사용하여 기억장치를 확보합니다. 예외나 신호가
  * 발생하는 경우 기억장치를 해제하지 못할 수 있습니다.
  * 이러한 기억장치 누출 방지가 프로그램에 필요한 경우
  * 이를 처리할 신호 핸들러를 빌드하거나 wrapper_2에서
```

```
  * 메소드를 사용할 수 있습니다.
 */
int wrapper_1(int arg1, void *arg2, double arg3,
                                                    char arg4, aggregate arg5, short arg6)
{
        int          result;
    /*
         * xlc는 모든 유형의 정적 변수에 대해 16바이트
          * 정렬을 보장하지 않지만 PASE malloc()은 반드시
          * 16바이트로 정렬된 기억장치를 리턴합니다.
          * size_ILEarglist()는 서명 배열의 항목에 기초하여
          * 필요한 기억장치의 용량을 판별합니다.
     */
        ILEarglist_base *ILEarglist;
        ILEarglist = (ILEarglist_base*)malloc( size_ILEarglist(signature) );

    /*
          * build_ILEarglist()는 신호 배열의 항목에
          * 기초하여 인수 값을 ILE 인수 리스트 버퍼에
          * 복사합니다.
     */
        build_ILEarglist(ILEarglist,
                         &arg1,
                         signature);

    /*
          * 저장된 PID 값을 사용하여 ILE 포인터가 설정되었는지
          * 여부를 검사할 수 있습니다. fork()의 하위 프로세서에서
          * 계승한 ILE 프로시듀어 포인터는 사용할 수 없는데,
          * 이들은 상위 프로세스에서 ILE 활성 그룹을 가리키기
          * 때문입니다.
     */
        if (getpid() != init_pid)
        do_init();

    /*
          * _ILECALL은 ILE 프로시듀어를 호출합니다. 예외나 신호가
          * 발생하는 경우 힙(heap) 할당이 분리됩니다(기억장치 누출).
     */
        _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;
}

/*
  * ILEarglistSt는 ILE 인수 리스트의 구조를 정의합니다.
  * xlc는 ILEpointer에 128비트 long double 멤버가 들어
  * 있기 때문에 ILEpointer 멤버 필드의 16바이트(상대)
  * 정렬을 제공합니다. 명시적 채움 필드는 자연적으로
  * ILE 관리 경계 내에 들지 않는 구조와 결합 유형의
  * 앞에서만 필요합니다.
 */
```

```
typedef struct {
        ILEarglist_base base;
        int32 arg1;
        /* 컴파일러가 제공하는 내재적 12바이트 채움 */
        ILEpointer arg2;
        float64 arg3;
        uint8 arg4;
        char filler[7]; /* 8바이트 정렬에 맞춰 채움 */
        aggregate arg5; /* 5바이트 집합(8바이트 정렬) */
        /* 컴파일러가 제공하는 내재적 1바이트 채움 */
        int16 arg6;
} ILEarglistSt;

/*
  * wrapper_2는 자신이 호출하는 ILE 프로시듀어와 같은
  * 인수를 채택하여 같은 결과를 리턴합니다. 이 예에서는
  * 이 방법에서는 ILE 인수 리스트에 대해 사용자 정의된
  * 또는 선언된 구조를 사용하여 실행 효율을 높이고
  * 예외나 신호 발생 시 힙(heap) 기억장치 누출을 방지합니다.
 */
int wrapper_2(int arg1, void *arg2, double arg3,
                                                char arg4, aggregate arg5, short arg6)
{
    /*
         * xlc는 모든 유형의 정적 변수에 대해 16바이트
         * 정렬을 보장하지 않으므로 크기가 초과된 버퍼에서
         * 정렬된 영역을 찾습니다.
     */
        char ILEarglist_buf[sizeof(ILEarglistSt) + 15];
        ILEarglistSt *ILEarglist = (ILEarglistSt*)ROUND_QUAD(ILEarglist_buf);
    /*
          * 지정문이 build_ILEarglist()를 호출하는 것보다
          * 빠릅니다.
     */
    ILEarglist->arg1 = arg1;
    ILEarglist->arg2.s.addr = (address64_t)arg2;
    ILEarglist->arg3 = arg3;
    ILEarglist->arg4 = arg4;
    ILEarglist->arg5 = arg5;
    ILEarglist->arg6 = arg6;
    /*
          * 저장된 PID 값을 사용하여 ILE 포인터가 설정되었는지
          * 여부를 검사할 수 있습니다. fork()의 하위 프로세서에서
          * 계승한 ILE 프로시듀어 포인터는 사용할 수 없는데,
          * 이들은 상위 프로세스에서 ILE 활성 그룹을 가리키기
          * 때문입니다.
     */
        if (getpid() != init_pid)
        do_init();
    /*
          * _ILECALL은 ILE 프로시듀어를 호출합니다. 스택은
          * 영향을 받지 않지만 예외나 신호 발생 시 힙(heap)
          * 기억장치가 분리되지 않습니다.
     */
        _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);
        }
    }
}
```

예 2: ILE C 코드

이 예에 대한 ILE C 코드를 i5/OS 시스템에 쓸 수 있습니다. 코드를 작성할 라이브러리에서 소스 실제 파일이 필요합니다. ILE 예에는 주석이 곳곳에 있습니다. 이들 주석은 코드를 이해하는 데 있어 아주 중요합니다. 소스를 입력하거나 검토할 때 이 주석을 반드시 검토하십시오.

```
#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) /* 불필요 */

/*
  * 이 ILE 프로시듀어에 대한 인수와 함수 결과는 OS/400 PASE
  * 프로그램에서 _ILECALL 함수에 제공되는 값과 같아야
  * 합니다.
 */
int ileProcedure(int         arg1,
                                char        *arg2,
                                double      arg3,
                                char        arg4[2],
                                aggregate   arg5,
                                short       arg6)
{
        char         fromcode[33];
```

```
    char         tocode[33];
    iconv_t      cd;     /* 변환 설명자 */
    char         *src;
    char         *tgt;
    size_t       srcLen;
    size_t       tgtLen;
    int          result;

/*
      * 변환 설명자를 열어 CCSID 37(EBCDIC)을, 호출자에
      * 리턴되는 문자 자료에 사용되는 CCSID 819(ASCII)로
      * 변환하십시오.
 */
    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;
}
 /*
       * arg1이 1이면 (ASCII0로 변환된) 상수 텍스트를
       * arg2가 주소 지정하는 버퍼에 리턴하십시오. 다른
       * 모든 arg1 값에 대해서는 파일을 열고 텍스트를
       * 읽은 다음 (ASCII로 변환된) 해당 텍스트를 arg2가
       * 주소 지정하는 버퍼에 리턴하십시오.
  */
    if (arg1 == 1)
{
            src = "Sample 1 output text";
            srcLen = strlen(src) + 1;
                 tgt = arg2; /* arg2 버퍼로 iconv 출력 */
                 tgtLen = srcLen;
                 iconv(cd, &src, &srcLen, &tgt, &tgtLen);

            result = 1;
}
   else
{
            FILE *fp;
            fp = fopen("SHUPE/PASEDATA", "r");
            if (!fp) /* 파일 열기 오류 시 */
    {
                     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; /* 널(null) 종료된 빈 버퍼 */
            }
                        src = buf;
                        srcLen = strlen(buf) + 1;
                        tgt = arg2; /* arg2 버퍼로 iconv 출력 */
                        tgtLen = srcLen;
                        iconv(cd, &src, &srcLen, &tgt, &tgtLen);

                        fclose(fp);
       }
              result = 1;
    }
      /*
             * 변환 설명자를 닫고 위에서 판별된
             * 결과 값을 리턴하십시오.
       */
           iconv_close(cd);
           return result;
}
```

예 3: 프로그램을 작성하는 컴파일러 명령

i5/OS PASE 프로그램을 컴파일할 때 컴파일러 옵션 -qalign=natural 및 -qldbl128을 사용하여 상대 16 바이트 정렬을 강제로 long double 유형으로 만들어야 합니다. 이 유형은 ILEpointer 유형 내에서 사용됩니다. 이 정렬은 i5/OS의 ILE에 필요합니다. -bI: 옵션의 경우에는 as400_libc.exp 파일을 저장한 경로명을 입력해야 합니다.

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

ILE C 모듈과 서비스 프로그램을 컴파일하는 경우 teraspace 옵션을 사용하여 이들을 컴파일하십시오. 그렇지 않으면 i5/OS PASE가 이들과 대화할 수 없습니다.

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

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

마지막으로 DDS를 컴파일하고 적어도 하나의 자료 레코드를 전파해야 합니다.

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

i5/OS PASE에서 i5/OS 프로그램 호출

i5/OS PASE 어플리케이션을 작성할 때 기존 i5/OS 프로그램(*PGM 오브젝트)을 활용할 수 있습니다. 또한, systemCL() 함수를 사용하여 CL CALL 명령을 실행할 수 있습니다.

i5/OS PASE 프로그램에서 i5/OS 프로그램을 호출하려면 _PGMCALL 런타임 함수를 사용하십시오.

이 메소드는 systemCL() 런타임 함수보다 빠른 처리를 제공하지만 PGMCALL_ASCII_STRINGS를 지정하지 않을 경우 문자 스트링 인수의 자동 변환을 수행하지 않으며 다른 작업에서 프로그램을 호출하는 기능을 제공하지 않습니다.

관련 태스크

46 페이지의『i5/OS PASE에서 i5/OS 명령 실행』

i5/OS 기능을 사용하는 제어 언어(CL) 명령을 실행하여 i5/OS PASE 프로그램의 기능을 확장할 수 있습니다.

관련 정보

_PGMCALL()--Call an i5/OS Program for i5/OS PASE

예: i5/OS PASE에서 i5/OS 프로그램 호출:

이 주제의 예를 읽고 _PGMCALL 런타임 함수를 사용하여 i5/OS PASE 프로그램에서 프로그램을 호출하는 방법을 학습할 수 있습니다.

다음 예는 _PGMCALL 런타임 함수를 사용하여 i5/OS PASE 프로그램에서 프로그램을 호출하는 방법을 보여줍니다.

다음의 코드 예제에는 코드를 설명하는 주석이 여러 곳에 나와 있습니다. 예를 입력하거나 검토할 때 이 주석을 반드시 읽어주십시오.

주: 코드 예제를 사용하여 66 페이지의『코드 라이센스 및 면책사항 정보』의 조건에 동의할 수 있습니다.

```
/* 이 예에서는 i5/OS PASE _PGMCALL 함수를 사용하여 i5/OS
API QSZRTVPR을 호출합니다. QSZRTVPR API는
i5/OS 소프트웨어 제품 로드에 대한 정보를 검색하는 데 사용됩니다.
API를 호출하는 데 필요한 입력 및 출력 매개변수에 관한 특정 정보는
QSZRTVPR API 문서를 참조하십시오.
*/

#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(QSZRTVPR 포함)는 일반적으로 문자 매개변수를 EBCDIC으로
      기대합니다. 그러나 i5/OS PASE 프로그램의 문자 상수는
      보통 ASCII입니다. 그러므로 QSZRTVPR를 호출하는 데
            필요한 일부 CCSID 37(EBCDIC) 문자 매개변수 상수를
            선언하십시오. */

      /* format[]은 QSZRTVPR에 대한 입력 매개변수 3이고
            EBCDIC인 텍스트 'PRDR0100'으로 초기화됩니다. */
      const char format[] =
            {0xd7, 0xd9, 0xc4, 0xd9, 0xf0, 0xf1, 0xf0, 0xf0};

      /* prodinfo[]는 QSZRTVPR에 대한 입력 매개변수 4이고
```

```
      EBCDIC인 텍스트 '*OPSYS *CUR  0033*CODE'로 초기화됩니다.

      이 값은 현재 설치된 i5/OS 릴리스의 옵션 33에 대한
   코드 로드를 검사하고자 함을 나타냅니다. */
   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};

   /* installed는 QSZRTVPR에서 리턴한 정보의 "로드 상태"
      필드와 비교되고 EBCDIC인 텍스트 '90'으로
      초기화됩니다. */
   const char installed[] = {0xf9, 0xf0};

   /* rcvr은 QSZRTVPR의 출력 매개변수 1입니다. */
   char rcvr[108];

   /* rcvrlen은 QSZRTVPR에 대한 입력 매개변수 2입니다. */
   int rcvrlen = sizeof(rcvr);

   /* errcode는 QSZRTVPR에 대한 입력 매개변수 5입니다. */
   struct {
      int bytes_provided;
      int bytes_available;
      char msgid[7];
   } errcode;


/* qszrtvpr_pointer에는 QSZRTVPR에 대한 i5/OS 16바이트의 태그 표시된 시스템
      시스템 포인터가 들어 갑니다. */
   ILEpointer qszrtvpr_pointer;

   /* qszrtvpr_argv6은 QSZRTVPR에 대한 인수 포인터의 배열입니다. */
   void *qszrtvpr_argv[6];

   /* _RSLOBJ2 및 _PGMCALL 함수의 리턴 코드 */
  int rc;

/* i5/OS 포인터를 QSYS/QSZRTVPR *PGM 오브젝트로 설정하십시오. */
   rc = _RSLOBJ2(&qszrtvpr_pointer,
                              RSLOBJ_TS_PGM,
                              "QSZRTVPR",
                              "QSYS");

   /* QSZRTVPR 리턴 정보 구조를 초기화하십시오. */
   memset(rcvr, 0, sizeof(rcvr));

   /* QSZRTVPR 오류 코드 구조를 초기화하십시오. */
   memset(&errcode, 0, sizeof(errcode));
   errcode.bytes_provided = sizeof(errcode);

   /* QSZRTVPR API에 대한 인수 포인터의 배열을 초기화하십시오. */
   qszrtvpr_argv[0] = &rcvr;
   qszrtvpr_argv[1] = &rcvrlen;
   qszrtvpr_argv[2] = &format;
   qszrtvpr_argv[3] = &prodinfo;
   qszrtvpr_argv[4] = &errcode;
   qszrtvpr_argv[5] = NULL;
```

```
   /* i5/OS PASE에서 i5/OS QSZRTVPR API를 호출하십시오. */
      rc = _PGMCALL(&qszrtvpr_pointer,
                                (void*)&qszrtvpr_argv,
                                0);

      /* 리턴된 정보의 63-64바이트에 대한 내용을 검사하십시오.
            내용이 '90'(EBCDIC)이 아닐 경우 코드 로드가 정확히
            설치되지 않은 것입니다. */
      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 PASE에서 i5/OS 명령 실행

i5/OS 기능을 사용하는 제어 언어(CL) 명령을 실행하여 i5/OS PASE 프로그램의 기능을 확장할 수 있습니다.

i5/OS PASE 프로그램에서 i5/OS 명령을 실행하려면 systemCL 런타임 함수를 사용하십시오.

i5/OS PASE에서 i5/OS 명령을 실행하면 systemCL 런타임 함수가 문자 스트링 인수의 ASCII 대 EBCDIC 변환을 처리하므로 다른 작업에서 이 프로그램을 호출할 수 있습니다.

관련 태스크

43 페이지의『i5/OS PASE에서 i5/OS 프로그램 호출』

i5/OS PASE 어플리케이션을 작성할 때 기존 i5/OS 프로그램(*PGM 오브젝트)을 활용할 수 있습니다. 또한, systemCL() 함수를 사용하여 CL CALL 명령을 실행할 수 있습니다.

관련 정보

systemCL()--Run a CL Command for i5/OS PASE

예: i5/OS PASE에서 i5/OS 명령을 실행하십시오.:

i5/OS PASE 프로그램에서 CL 명령을 실행하는 방법을 학습하려면 이 주제에서 제공하는 예를 참조할 수 있습니다.

주: 코드 예제를 사용하여 66 페이지의『코드 라이센스 및 면책사항 정보』의 조건에 동의할 수 있습니다.

다음 예는 i5/OS PASE 프로그램에서 명령을 호출하는 방법을 보여줍니다.

```
/* sampleCL.c
      sampleCL을 사용한 CL 명령의 실행 예
      다음과 유사한 명령을 사용하여 컴파일하십시오.
      xlc -o sampleCL -I /whatever/pase -bI:/whatever/pase/as400_libc.exp sampleCL.c
      다음은 QP2SHELL()을 사용한 프로그램 예입니다.
   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]);

      /* CL 명령을 처리합니다. */
      rc = systemCL(argv[1], /* CL 명령의 첫 번째 매개변수를 사용합니다. */
                            SYSTEMCL_MSG_STDOUT
                            SYSTEMCL_MSG_STDERR ); /* 메세지를 수집합니다. */

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

i5/OS PASE 프로그램과 i5/OS의 대화 방법

i5/OS 기능을 사용하도록 i5/OS PASE 프로그램을 사용자 정의할 때 프로그램이 이들과 대화하는 방법을 고려해야 합니다.

관련 태스크

19 페이지의『i5/OS 함수를 사용하도록 i5/OS PASE 프로그램 사용자 정의』

AIX 어플리케이션에서 시스템 제공 i5/OS PASE 공유 라이브러리가 직접 지원하지 않는 i5/OS 함수를 사용하려면 어플리케이션을 준비하기 위한 몇 가지 추가 단계를 수행해야 합니다.

통신

i5/OS PASE는 일반적으로 소켓 통신에서 AIX 및 Linux와 호환 가능합니다.

i5/OS PASE는 소켓 통신에 대해 AIX와 동일한 구문을 지원합니다. Linux와 같은 다른 오퍼레이팅 시스템과는 모두 일치하지 않습니다.

i5/OS PASE 소켓 지원은 AIX 소켓 구현과 유사하지만 i5/OS PASE는 i5/OS 소켓 구현(AIX 커널의 소켓 구현 대신)을 사용하므로 AIX 작동 방식과 약간의 차이가 발생합니다.

i5/OS 소켓 구현은 UNIX 98 및 BSD(Berkeley Software Distributions) 소켓을 모두 지원합니다. 대부분의 경우, i5/OS PASE는 AIX 구현의 작동을 채택하여 이러한 스타일의 차이점을 해결합니다.

또한 실행 중인 어플리케이션에 대한 사용자 프로파일에는 소켓 API에서 level 매개변수를 IPPROTO_IP로 지정하고 option_value 매개변수를 IP_OPTIONS로 지정하기 위한 *IOSYSCFG 특수 권한이 있어야 합니다.

관련 정보

Socket programming

Berkeley Software Distributions (BSD) compatibility

UNIX 98 compatibility

데이터베이스

i5/OS PASE는 iSeries용 DB2 UDB CLI(Call Level Interface)를 지원합니다. AIX 및 i5/OS의 DB2 CLI는 서로에 대해 적절한 서브세트가 아니므로 일부 인터페이스에 약간의 차이점이 있고 구현된 일부 API가 다른 시스템에는 없을 수 있습니다.

이 때문에 다음 사항을 고려해야 합니다.

- 코드를 생성할 수는 있지만 AIX 자체에서 테스트할 수 없습니다. 대신, i5/OS PASE의 플랫폼에서 코드를 테스트해야 합니다.
- i5/OS 버전의 헤더 파일 `sqlcli.h`를 사용하여 컴파일해야 합니다. 이 헤더 파일의 AIX 버전을 사용하여 컴파일한 프로그램은 i5/OS PASE에서 실행되지 않습니다.

i5/OS는 디폴트로 EBCDIC 코드화 시스템이지만 AIX는 ASCII를 기반으로 합니다. 이러한 차이점 때문에 종종 i5/OS 데이터베이스(iSeries용 DB2 UDB) 및 i5/OS PASE 어플리케이션 간에 자료 변환이 필요합니다.

DB2 CLI를 i5/OS PASE에서 구현할 경우, i5/OS PASE 시스템 제공 라이브러리 루틴은 자동으로 문자 자료에 대해 ASCII에서 EBCDIC로 또는 그 반대로 자료 변환을 수행합니다. 변환은 액세스 중인 자료의 태그 표시된 CCSID 및 i5/OS PASE 프로그램이 실행 중인 ASCII CCSID를 기반으로 수행됩니다. 데이터베이스가 CCSID 65535를 사용하여 태그되는 경우 자동 변환이 수행되지 않습니다. 자료의 코드화 형식을 이해하고 필요한 변환을 수행하는 것은 어플리케이션입니다.

CCSID에 대한 작업

Qp2RunPase() API를 사용할 때 i5/OS PASE CCSID를 명시적으로 지정해야 합니다.

API 프로그램 QP2TERM, QP2SHELL 또는 QP2SHELL2를 호출하기 전에 ILE에서 다음 변수를 모두 설정하여 i5/OS PASE CCSID를 제어할 수 있습니다.

- PASE_LANG
- QIBM_PASE_CCSID

ILE가 이들 변수 중 하나라도 생략하면 디폴트로 QP2TERM, QP2SHELL 및 QP2SHELL2는 i5/OS PASE CCSID 및 i5/OS PASE 환경 변수 `LANG` 값을 작업의 언어 및 CCSID 속성에 해당하는 가장 적합한 i5/OS PASE 값으로 설정합니다.

libc.a에 대한 확장 기능을 통해 i5/OS PASE 어플리케이션이 _SETCCSID() 함수를 사용하여 어플리케이션의 실행 중인 CCSID를 변경할 수 있습니다.

다른 확장을 통해 i5/OS PASE 어플리케이션이 어플리케이션의 CCSID를 변경하지 않고 DB2 CLI 내부 변환을 대체할 수 있는 기능을 제공합니다. SQLOverrideCCSID400() 함수는 대체 CCSID의 정수를 단일 매개변수로 승인합니다.

주: CCSID 대체 함수 SQLOverrideCCSID400()가 대체 기능을 할 수 있으려면 다른 SQLx() API 보다 먼저 호출되어야 합니다. 그렇지 않으면 요청이 무시됩니다.

i5/OS PASE 프로그램에서 iSeries용 DB2 UDB 사용

i5/OS PASE 프로그램에서 DB2 CLI를 사용하려면 소스를 컴파일하기 전에 sqlcli.h 헤더 파일 및 libdb400.exp 내보내기 파일을 AIX 시스템에 복사해야 합니다. DB2 CLI 라이브러리 루틴은 i5/OS PASE 환경의 경우 `libdb400.a`에 있으며 pthread 인터페이스를 사용하여 구현되고 스레드세이프를 제공합니다. 대부분의 i5/OS PASE CLI 함수는 해당 ILE CLI 함수를 호출하여 필요한 조작을 수행합니다.

주: i5/OS PASE 프로그램에서 DB2 CL를 사용할 때 다음을 고려하십시오.

- SQLGetSubString은 CLOB/DBCLOB 필드를 서브 스트링할 때 항상 EBCDIC 스트링을 리턴합니다. SQLGetSubString은 LOB 자료 유형에만 사용됩니다.
- 결과 세트(표 유형)의 네 번째 열 SQLTables은 항상 EBCDIC로 리턴됩니다.
- i5/OS PASE 프로그램에서 그래픽 유형의 자료를 표시하려면 자료가 프로그램에서 `wchar`로 입력되어야 합니다. 이렇게 하면 데이터베이스가 그래픽 및 완전 2바이트 문자에서 Unicode/UCS-2로 변환됩니다. 그렇지 않으면 데이터베이스는 자료의 CCSID와 i5/OS 작업의 CCSID 사이에서 변환됩니다. 데이터베이스는 Qp2RunPase() API 또는 SQLOverrideCCSID400() API에서 EBCDIC 그래픽과 CCSID 간의 변환을 지원하지 않습니다.

관련 참조

19 페이지의『헤더 파일 복사』
이 주제의 지침을 수행하여 iSeries 서버에서 AIX 기계로 헤더 파일을 복사할 수 있습니다.

21 페이지의『내보내기 파일 복사』
이 주제의 지침을 수행하여 iSeries 서버에서 AIX 디렉토리로 내보내기 파일을 복사할 수 있습니다.

관련 정보

QP2TERM()--Run an i5/OS PASE Terminal Session

QP2SHELL() and QP2SHELL2()--Run an i5/OS PASE Shell Program

_SETCCSID()--Set i5/OS PASE CCSID

SQLOverrideCCSID400()--Override SQL CLI CCSID for i5/OS PASE

SQL call level interface

예: i5/OS PASE 프로그램에서 iSeries용 DB2 UDB CLI 함수 호출:

이 주제의 예는 iSeries용 DB2 UDB SQL 호출 레벨 인터페이스를 사용하여 iSeries용 DB2 UDB를 액세스하는 i5/OS PASE 프로그램을 보여줍니다.

주: 코드 예제를 사용하여 66 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의할 수 있습니다.

```
/* i5/OS PASE iSeries용 DB2 UDB 프로그램 예
 *
 * SQL CLI를 통해 i5/OS DB2 UDB에 액세스하는
 * i5/OS PASE 프로그램의 예를 보여줍니다.
 *
  * 프로그램은 모든 시스템에 존재해야 하는 iSeries Access 데이터베이스,
  * QIWS/QCUSTCDT에 액세스합니다.
 *
  * fun_Connect() 프로시듀어의 시스템명, 사용자 ID 및 암호를
  * 유효한 매개변수로 변경하십시오.
 *
  * 컴파일 호출:
 *
  * xlc -I./include -bI:./include/libdb400.exp -o paseclidb4 paseclidb4.c
 *
  * 2진으로 FTP 전송하고 QP2TERM() 단말기 쉘에서 실행하십시오.
 *
 * 출력에는 STATE 열이 MN과 일치하는 모든 행이 표시되어야 합니다.  */
/* 변경 활동: */
/* 변경 활동 끝 */

#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 */
```

자료 코드화

AIX 및 Linux와 같은 대부분의 오퍼레이팅 시스템은 ASCII 문자 코드화를 사용합니다. 대부분의 i5/OS 함수는 EBCDIC 문자 코드화를 사용합니다. 일부 i5/OS 오브젝트 유형에 대한 코드화 문자 세트 ID(CCSID) 값을 지정하여 오브젝트의 문자 자료에 대한 특정 코드화를 식별할 수 있습니다.

i5/OS PASE 바이트 스트림 파일에는 i5/OS PASE 외부의 대부분의 시스템 인터페이스가 필요한 대로 파일에 쓰거나 파일에서 읽는 텍스트 자료를 편집하는 데 사용되는 CCSID 속성이 있습니다. i5/OS PASE는 스트림 파일(AIX와 일치)에 쓰거나 이 파일에서 읽은 자료에 대해서는 CCSID 변환을 수행하지 않지만, 다른 시스템 함수가 파일에 있는 ASCII 텍스트를 올바로 처리할 수 있도록 i5/OS PASE 프로그램에서 작성한 모든 바이트 스트림 파일의 CCSID 속성을 현재 i5/OS PASE CCSID 값으로 설정합니다.

i5/OS PASE 공유 라이브러리에 제공된 AIX API를 사용할 경우 i5/OS PASE는 대부분의 자료 변환을 처리합니다. i5/OS PASE 프로그램은 i5/OS PASE 런타임으로 자동 처리되지 않은 모든 문자 자료 변환에 대해 공유 라이브러리 `libiconv.a`에 제공된 `iconv` 함수를 사용할 수 있습니다. 예를 들어 i5/OS PASE 어플리케이션은 일반적으로 i5/OS API 함수를 호출(_ILECALLX 또는 _PGMCALL 사용)하기 전에 문자 스트링을 EBCDIC로 변환해야 합니다.

관련 개념

『파일 시스템』
i5/OS PASE 프로그램은 QSYS.LIB 및 QOPT 파일 시스템의 오브젝트를 포함하여 통합 파일 시스템을 통해 액세스할 수 있는 파일 또는 자원에 액세스할 수 있습니다.

파일 시스템

i5/OS PASE 프로그램은 QSYS.LIB 및 QOPT 파일 시스템의 오브젝트를 포함하여 통합 파일 시스템을 통해 액세스할 수 있는 파일 또는 자원에 액세스할 수 있습니다.

버퍼링된 입력 및 출력

외부 장치 간의 입력 및 출력은 i5/OS에 버퍼링되며 이는 자료의 블록을 처리하는 입력 및 출력 프로세서에 의해 처리됩니다. 이와 반대로 AIX 및 Linux와 같은 오퍼레이팅 시스템은 일반적으로 문자 단위(버퍼링되지 않은) 입출력으로 작동합니다. i5/OS에서는 특정 입력 및 출력 신호(예를 들어, Enter 키, 기능 키 및 시스템 요청)만 시스템에 인터럽트를 송신할 수 있습니다.

자료 변환 지원

i5/OS PASE 프로그램은 ASCII(또는 UTF-8) 경로명을 open() 함수로 전달하여 바이트 스트림 파일을 엽니다. 여기서 이름은 i5/OS에서 사용되는 코드화 체계로 자동 변환되지만 열려진 파일에서 읽거나 기록된 자료는 변환되지 않습니다.

파일 설명자 사용

i5/OS PASE 런타임은 보통 stdin, stdout 및 stderr 파일에 ILE C 런타임 지원을 사용하여 i5/OS PASE 및 ILE 프로그램에 일관된 작동을 제공합니다.

i5/OS PASE 및 ILE C는 표준 입력 및 출력(stdin, stdout 및 stderr)에 동일한 스트림을 사용합니다. i5/OS PASE 프로그램은 항상 파일 설명자 0, 1 및 2를 사용하여 표준 입력 및 출력에 액세스합니다. 그러나 ILE C는 stdin, stdout 및 stderr에 대해 통합 파일 설명자를 항상 사용하지는 않으므로 i5/OS PASE는 i5/OS PASE 파일 설명자와 통합 파일 시스템의 설명자 간의 맵핑을 제공합니다. 이러한 맵핑때문에 i5/OS PASE 프로그램과 ILE C 프로그램은 서로 다른 설명자 번호를 사용하여 동일한 열린 파일에 액세스할 수 있습니다.

`fcntl` 함수 F_MAP_XPFFD에서 i5/OS PASE 확장 기능을 사용하여 ILE 번호에 i5/OS PASE 설명자를 할당할 수 있습니다. 이는 i5/OS PASE 어플리케이션이 i5/OS PASE에서 작성하지 않은 ILE 설명자에 대해 파일 조작을 수행할 필요가 있을 때 유용합니다.

fstatx() 함수에 대한 i5/OS 시스템 고유 확장 기능 STX_XPFFD_PASE를 통해 i5/OS PASE 프로그램은 i5/OS PASE 파일 설명자에 대한 통합 파일 시스템 설명자를 판별할 수 있습니다. stdin, stdout 및 stderr 파일에 대한 ILE C 런타임 지원에 첨부된 i5/OS PASE 설명자에 대해서는 특수 값(음수)이 리턴됩니다.

Qp2RunPase() API를 호출할 때 ILE 환경 변수 QIBM_USE_DESCRIPTOR_STDIO는 Y 또는 I로 설정되는 경우 i5/OS PASE는 설명자 0, 1 및 2를 통합 파일 시스템과 동기화시키므로 i5/OS PASE 및 ILE C 프로그램은 모두 stdin, stdout 및 stderr 파일에 대해 동일한 설명자 번호를 사용하게 됩니다. 이러한 모드에서

작동할 때 i5/OS PASE 코드 또는 ILE C 코드가 파일 설명자 0, 1 또는 2를 닫거나 다시 여는 경우 이 변경사항은 두 환경 모두에서 stdin, stdout 및 stderr 처리에 영향을 줍니다.

i5/OS PASE 런타임은 일반적으로 i5/OS PASE 파일 설명자(소켓 포함)를 통해 읽거나 기록된 자료에 대해서는 문자 코드화 변환을 수행하지 않습니다. 단, ILE C stdin에서 읽거나 ILE C stdout 및 stderr에 기록된 자료의 경우 i5/OS PASE CCSID와 작업 디폴트 CCSID 사이에서 ASCII 대 EBCDIC 변환이 수행됩니다).

두 환경 변수는 stdin, stdout 및 stderr의 자동 변환을 제어합니다.

- 일반적으로 적용되는 변수는 QIBM_USE_DESCRIPTOR_STDIO입니다. Y로 설정할 때 ILE 런타임은 이들 파일에 대해 파일 설명자 0, 1 또는 2를 사용합니다.
- i5/OS PASE 시스템 특정 환경 변수는 QIBM_PASE_DESCRIPTOR_STDIO입니다. 이 변수는 2진에 대해서는 B의 값을 가지고 텍스트에 대해서는 T의 값을 갖습니다.

ILE 환경 변수 QIBM_USE_DESCRIPTOR_STDIO를 Y로 설정하고 QIBM_PASE_DESCRIPTOR_STDIO를 B로 설정할 경우(stdin에서 2진 자료를 읽고 stdout 또는 stderr에 기록할 수 있음) i5/OS PASE stdin, stdout 및 stderr에 대한 ASCII 대 EBCDIC 변환을 사용할 수 없습니다. QIBM_PASE_DESCRIPTOR_STDIO에 대한 디폴트는 텍스트의 경우 T입니다. 이 값은 EBCDIC에서 ASCII로 변환이 이루어지게 합니다.

관련 개념

55 페이지의 『자료 코드화』

AIX 및 Linux와 같은 대부분의 오퍼레이팅 시스템은 ASCII 문자 코드화를 사용합니다. 대부분의 i5/OS 함수는 EBCDIC 문자 코드화를 사용합니다. 일부 i5/OS 오브젝트 유형에 대한 코드화 문자 세트 ID(CCSID) 값을 지정하여 오브젝트의 문자 자료에 대한 특정 코드화를 식별할 수 있습니다.

관련 정보

Integrated file system

국제화

i5/OS PASE 런타임이 AIX 런타임에 기반하므로 i5/OS PASE 프로그램에서는 AIX에 지원되는 로케일, 문자 스트링 조작, 날짜 및 시간 서비스, 메세지 카탈로그 및 문자 코드화 변환 등에 동일한 여러 프로그래밍 인터페이스 세트를 사용할 수 있습니다.

i5/OS PASE는 1바이트 및 복수 바이트 둘 다의 문자 코드화 지원을 포함하여 어플리케이션이 사용하는 로케일을 관리하고 로케일 관련 함수(예; ctype 및 strcoll)를 수행할 수 있도록 AIX 런타임의 인터페이스를 지원합니다.

i5/OS PASE는 산업 표준 코드화(코드 세트 ISO8859-x), 코드 세트 IBM-1250 및 코드 세트 UTF-8을 사용하여 여러 국가 및 언어를 지원하는 AIX 로케일의 서브세트를 포함합니다. i5/OS PASE는 IBM-1252 로케일, ISO 8859-15 로케일(둘 다 1바이트 코드화 사용) 및 UTF-8 로케일의 세 가지 서로 다른 방식으로 유로를 지원합니다.

주: i5/OS PASE에 대한 로케일 지원은 ILE C 프로그램이 사용하는 로케일 지원 형식과는 무관합니다(오브젝트 유형 *CLD 및 *LOCALE). 이러한 내부 구조의 차이 외에, ILE C 프로그램에 대해 기존에 제공된 로케일은 ASCII를 지원하지 않습니다.

새 로케일 작성

i5/OS PASE는 새 로케일을 작성하는 유틸리티를 제공하지 않습니다. 그러나 `localedef` 유틸리티를 사용하여 AIX 시스템의 i5/OS PASE에 사용할 로케일을 작성할 수 있습니다.

로케일 변경

i5/OS PASE 어플리케이션이 로케일을 변경할 경우 일반적으로 새 로케일의 코드화와 일치하도록 i5/OS PASE CCSID도 변경해야 합니다(_SETCCSID 런타임 함수를 사용하여). 이렇게 하면 i5/OS PASE 런타임이 문자 데이터 인터페이스 인수를 올바로 해석하고 EBCDIC 시스템 서비스를 호출할 때 변환될 수 있습니다. `cstoccsid` 런타임 함수를 사용하여 코드 세트 이름에 해당하는 CCSID를 판별할 수 있습니다.

i5/OS PASE 런타임은 i5/OS PASE 프로그램이 작성한 파일의 CCSID 태그를 현재 i5/OS PASE CCSID 값(최신 _SETCCSID 값을 사용하거나 프로그램이 시작될 때 제공됨)으로 설정합니다.

일본어, 한국어, 중국어 및 대만어를 지원하는 i5/OS PASE 어플리케이션에는 UTF-8 로케일을 사용해야 합니다. i5/OS에는 이들 언어에 대한 다른 로케일이 포함되어 있지만 시스템은 IBM-euc*XX* 코드 세트의 코드화와 일치시키기 위한 i5/OS PASE CCSID의 설정을 지원하지 않습니다. UTF-8 지원을 사용하면 어플리케이션이 다른 플랫폼에서 실행될 때 다른 코드화(예: Shift-JIS)로 저장될 수 있는 파일 자료 변환이 필요할 수 있습니다.

i5/OS PASE 변환 오브젝트 및 로케일이 저장되는 위치

i5/OS PASE에 대한 변환 오브젝트 및 로케일은 i5/OS 언어 피처 코드와 함께 패키지됩니다. i5/OS PASE를 설치하면 설치된 i5/OS 언어 피처와 연관된 로케일만 작성됩니다.

모든 i5/OS PASE 로케일은 ASCII 또는 UTF-8 문자 코드화를 사용하므로 모든 i5/OS PASE 런타임은 ASCII(또는 UTF-8)에서 작동됩니다.

관련 태스크

6 페이지의『i5/OS PASE 설치』

이 주제의 지침을 수행하여 i5/OS PASE를 서버에 설치할 수 있습니다.

관련 정보

i5/OS globalization

i5/OS PASE locales

_SETCCSID()--Set i5/OS PASE CCSID

메세지 서비스

i5/OS PASE 신호와 ILE 신호가 독립적이므로 한 유형의 신호를 일으켜서 다른 신호 유형에 대한 핸들러를 직접 호출할 수 없습니다.

i5/OS PASE Qp2SignalPase() API를 사용하여 사용자가 수신하는 ILE 신호에 해당하는 해당 i5/OS PASE 신호를 게시할 수 있습니다. QP2SHELL() 프로그램 및 i5/OS PASE `fork()` 함수는 항상 모든 ILE 신호를 해당 i5/OS PASE 신호로 맵핑하도록 핸들러를 설정합니다.

시스템은 Qp2RunPase, Qp2CallPase 또는 Qp2CallPase2 API를 실행하는 호출의 프로그램 메세지 대기행렬로 송신된 모든 i5/OS 예외 메세지를 해당 i5/OS PASE 신호로 자동 변환합니다. 따라서 i5/OS PASE 어플리케이션은 시스템이 변환하는 i5/OS PASE 신호를 처리함으로써 모든 i5/OS 예외를 처리할 수 있습니다.

i5/OS PASE는 i5/OS 메세지 처리에 대한 직접 제어를 부여하는 다음과 같은 런타임 함수를 제공합니다.

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

이들 함수에 대한 세부사항은 런타임 함수를 참조하십시오.

i5/OS 메세지 지원

i5/OS는 메세지 지원을 다양한 문맥으로 제공합니다.

- **작업 기록부**. 작업 기록부에는 i5/OS 또는 어플리케이션이 실행되거나 컴파일되는 동안 발행하는 모든 메세지가 들어 있습니다. 작업 기록부를 살펴보려면 명령행에 `DSPJOBLOG`를 입력하십시오. 작업 기록부 표시 화면이 나타나면 F10과 Shift + F6을 누르십시오. 이들 키 조합을 누르면 모든 메세지 표시 화면이 표시되고 가장 최신 메세지로 설정됩니다. 특정 메세지의 세부사항을 보려면 커서를 해당 메세지로 이동하여 F1을 누르십시오.
- **활동 작업에 대한 작업**. WRKACTJOB(활동 작업에 대한 작업) 명령은 i5/OS의 작업 및 작업 스택을 살펴보는 데 유용합니다.

  관련 정보

  Qp2SignalPase()--Post an i5/OS PASE Signal

  Runtime functions for use by i5/OS PASE programs

Work with active jobs (WRKACTJOB)

Work management

i5/OS PASE signal handling

i5/OS PASE 어플리케이션에서 인쇄 출력

QShell Rfile 유틸리티를사용하여 i5/OS PASE 쉘에서 출력을 읽고 쓸 수 있습니다.

다음 예에서는 스트림 파일 mydoc.ps의 내용을 스풀 프린터 장치 파일 QPRINT에 변환되지 않은 ASCII 자료로 기록한 다음 CL LPR 명령을 사용하여 스풀 파일을 다른 시스템으로 송신합니다.

```
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
```

관련 정보

Rfile - Read or write record files

유사 단말기(PTY)

i5/OS PASE는 AT&T 및 BSD(Berkeley Software Distributions) 스타일 장치를 지원합니다. 프로그래밍 관점에서 볼 때 이러한 장치는 AIX에서 작동하는 것과 동일한 방법으로 i5/OS PASE에서 작동합니다.

i5/OS PASE를 사용하면 AT&T 스타일 장치에 대해 최대 1024개의 인스턴스와 최대 592개의 BSD 스타일 장치를 사용할 수 있습니다. 시스템이 시작되면 각 장치 유형의 처음 32개 인스턴스가 자동으로 작성됩니다.

i5/OS PASE에서 PTY 장치 구성

AIX에서 관리자는 smit를 사용하여 각 유형의 사용 가능한 장치 수를 구성합니다. i5/OS PASE에서 이들 장치는 다음과 같은 방법으로 구성됩니다.

- AT&T 스타일 장치의 경우, i5/OS PASE는 자동 구성을 지원합니다. 처음 32개 인스턴스가 사용 중이고 어플리케이션이 다른 인스턴스를 열려고 하면 최대 1024개의 한도까지 통합 파일 시스템에 CHRSF 장치가 자동으로 작성됩니다.
- BSD 스타일 장치의 경우 i5/OS PASE mknod 유틸리티를 사용하여 CHRSF 장치를 수동으로 작성해야 합니다. 이를 수행하려면 명명 규칙과 함께 BSD 종속 및 BSD 1차 장치에 대한 주요 번호를 알고 있어야 합니다. 다음 예의 쉘 스크립트는 추가 BSD PTY(Pseudo-terminal) 장치를 작성하는 방법을 보여줍니다. 16개 그룹으로 이 장치를 작성합니다.

  주: 코드 예제를 사용하여 66 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의할 수 있습니다.

```
#!/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 장치에 대한 자세한 정보는 AIX 문서 웹 사이트를 참조하십시오.

보안

보안 관점에서 볼 때 i5/OS PASE 프로그램은 i5/OS의 기타 프로그램과 동일한 보안 제한사항이 적용됩니다.

i5/OS에서 i5/OS PASE 프로그램을 실행하려면 통합 파이 시스템에서 AIX 2진에 대한 권한이 있어야 합니다. 또한 사용자 프로그램이 액세스하는 각 자원에 대해 적절한 레벨의 권한이 있어야 하고 그렇지 않으면 이 자원에 액세스하려 할 때 오류가 수신됩니다.

다음 정보는 i5/OS PASE 프로그램을 실행할 때 특히 중요합니다.

사용자 프로파일 및 권한 관리

시스템 권한 관리는 오브젝트이기도 한 사용자 프로파일에 기반합니다. 시스템에 작성되는 모든 오브젝트는 특정 사용자가 소유합니다. 오브젝트에 대한 각 조작이나 액세스는 사용자의 권한을 확인하기 위해 시스템에서

검증합니다. 소유자 또는 적절한 권한이 있는 사용자 프로파일은 오브젝트를 조작할 수 있는 여러 가지 유형의 권한을 다른 사용자 프로파일에 위임할 수 있습니다. 모든 유형의 오브젝트에 동일하게 권한 검사가 제공됩니다.

오브젝트 권한 메카니즘은 여러 가지 제어 레벨을 제공합니다. 사용자의 권한은 정확하게 필요한 권한으로 제한될 수 있습니다. QOpenSys 파일 시스템에 저장된 파일에는 UNIX 파일과 동일한 방식으로 권한이 부여됩니다. 다음 표는 UNIX 권한과 i5/OS 데이터베이스 파일에 사용된 보안 값 사이의 관계를 보여줍니다. i5/OS에서 *OBJOPR은 *오브젝트 사용* 권한이고, *EXCLUDE는 *권한 없음*입니다. *READ, *ADD, *UPD, *DLT 및 *EXECUTE는 자료 권한입니다. 파일을 i5/OS PASE 프로그램으로 실행하려면 파일에 대해 *EXECUTE 권한(및 때때로 *READ 권한)이 있어야 합니다.

UNIX 권한	***OBJOPR**	***READ**	***ADD**	***UPD**	***DLT**	***EXECUTE**
r(read)	X	X	-	-	-	-
w(write)	X	-	X	X	X	-
x(execute)	X	-	-	-	-	X
권한 없음	-	-	-	-	-	-

i5/OS PASE의 사용자 프로파일

i5/OS에서 인증 정보는 `/etc/passwd`와 같은 파일이 아니라 개별 *profiles*에 저장됩니다. 사용자와 그룹은 프로파일을 갖습니다. 이들 모든 프로파일이 이름공간을 공유하며 각 프로파일은 대소문자가 혼합되지 않은 고유한 이름을 가져야 합니다. 소문자 이름을 getpwnam() 또는 getgrnam() API로 전달하면 시스템은 이 이름 스트링을 예상되는 대소문자로 변환합니다.

리턴된 프로파일 이름을 가져오기 위해 getpwuid() 또는 getgrgid()를 호출할 경우, 결과를 대문자로 리턴하는 i5/OS PASE 환경 변수를 PASE_USRGRP_LOWERCASE=N으로 설정하지 않으면 프로파일 이름은 소문자로 됩니다.

모든 사용자는 사용자 ID(`UID`)를 갖습니다. 모든 그룹은 그룹 ID(`GID`)를 갖습니다. 이들은 POSIX(Portable Operation System Interface X) 1003.1 표준에 따라 정의됩니다. 두 개의 숫자 공간이 분리되므로 UID가 104인 사용자와 GID가 104인 그룹을 가질 수 있습니다.

i5/OS에는 UID가 0인 보안 담당자 QSECOFR에 대한 사용자 프로파일이 있습니다. 다른 프로파일은 UID를 0으로 가질 수 없습니다. QSECOFR은 시스템에서 가장 권한이 있는 프로파일로서 루트 사용자의 역할을 합니다. 그러나, i5/OS는 시스템 관리자가 개별 사용자에게 지정할 수 있는 특정 권한 세트도 제공합니다. 이러한 권한 중 하나인 *ALLOBJ는 파일 액세스에 대한 임의의 액세스 제어(예를 들어 AIX 및 Linux와 같은 오퍼레이팅 시스템에서 전형적인 루트 권한의 사용)를 대체합니다.

루트 액세스를 사용하는 이식된 어플리케이션에서는 *ALLOBJ 권한이 주어지는 *어플리케이션 사용자*에 대한 특정 사용자 프로파일을 작성하여, 단일 어플리케이션이 필요로 하는 것 이상의 권한을 갖는 QSECOFR의 사용을 방지하는 것이 좋습니다. AIX 또는 Linux와 같은 오퍼레이팅 시스템과는 달리 i5/OS는 사용자에 대한 그룹 멤버쉽이 필요하지 않습니다. i5/OS에서 사용자 프로파일에 대한 GID가 0이면 더 많은 권한을 가진 그룹이 아닌 *할당된 그룹 없음*을 의미합니다.

i5/OS 보안은 시스템에 빌드된 통합 보안을 사용합니다. 오브젝트에 대한 모든 액세스는 보안 검사를 통과해야 합니다. 보안 검사는 액세스 시 프로세스가 실행 중이던 사용자 프로파일에 대해 수행됩니다.

i5/OS PASE는 각 프로세스에 무결성 및 보안을 유지하기 위한 별도의 주소 공간을 제공합니다. i5/OS PASE 주소 공간에서 자원을 사용할 수 없으면 이 자원에 액세스할 수 없습니다. 파일 시스템 보안은 누군가가 적절한 권한 없이 자신의 주소 공간으로 자원을 로드하는 것을 막아줍니다. 주소 공간에 있는 자원은 프로세스가 실행되는 ID에 관계없이 프로세스에서 사용할 수 있습니다.

i5/OS PASE 프로그램은 시스템 호출을 사용하여 시스템 함수를 요청합니다. i5/OS PASE 프로그램에 대한 시스템 호출은 i5/OS에서 처리합니다. 이 인터페이스는 시스템 내부에 대해 i5/OS PASE 프로그램의 간접적이고 안전한 액세스만 제공합니다.

관련 정보

Security

작업 관리

i5/OS는 시스템에서 다른 작업을 처리하는 것과 같은 방법으로 i5/OS PASE 프로그램을 처리합니다.

관련 정보

Work management

---

i5/OS PASE 프로그램 디버그

i5/OS PASE 런타임 환경은 syslog() 런타임 함수에 대한 라이브러리 지원을 제공하고 더 복잡한 메세지 라우팅을 위해 `syslogd` 2진을 제공합니다. 또한 i5/OS 시스템 오퍼레이터 메세지 대기행렬 QSYSOPR로 심각한 메세지 송신 및 진단 메세지에 대한 작업 기록부와 같은 i5/OS의 기존 기능을 사용할 수 있습니다.

어플리케이션에 따라 i5/OS PASE 어플리케이션을 디버깅하는 전략이 달라질 수 있습니다.

- 어플리케이션을 i5/OS 통합(예: iSeries용 DB2 UDB 또는 ILE 함수와의 통합)할 필요가 없으면 먼저 AIX에서 어플리케이션을 디버그해야 합니다.
- 그런 다음 i5/OS PASE `dbx` 및 i5/OS 디버그 기능(예: 작업 기록부) 조합을 사용하여 i5/OS에서 어플리케이션을 디버그합니다.

데이터베이스 또는 ILE 함수를 사용하도록 코딩한 어플리케이션을 AIX에서 전부 테스트할 수는 없지만 AIX에서 어플리케이션의 나머지 부분을 디버그하여 적절한 구조 및 설계를 할 수 있습니다.

i5/OS PASE에서 *dbx* 사용

i5/OS PASE는 AIX `dbx` 디버거 유틸리티를 지원합니다. 이 유틸리티를 사용하면 적절히 컴파일되지 않은 경우 소스 코드 레벨에서 관련 프로세스(상위 및 하위)를 디버그할 수 있습니다. 네트워크 파일 시스템(NFS)을 사용하여 i5/OS PASE에서 실행되는 디버거에 AIX 소스를 표시할 수 있습니다.

i5/OS PASE의 xterm 및 aixterm 지원은 `dbx`를 사용하여 상위 및 하위 프로세스를 모두 디버그할 수 있도록 합니다. `dbx`는 `dbx`를 두 번째 프로세스에 접속한 상태로 다른 `xterm` 창을 시작합니다.

dbx에 대한 자세한 정보는 AIX 문서 웹 사이트를 참조하십시오. dbx 명령행에 help를 입력할 수도 있습니다.

i5/OS 디버깅 툴 사용

i5/OS에서 다음 툴을 사용하여 i5/OS PASE 어플리케이션을 디버그할 수 있습니다.

- iSeries 시스템 디버거는 i5/OS PASE 어플리케이션 디버깅에 대한 특정 지원을 제공합니다.
- ILE C 소스 디버거는 코드의 문제점을 판별하기 위한 효과적 툴입니다.

  관련 정보

  iSeries System Debugger

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

성능 최적화

최상의 성능을 얻으려면 어플리케이션 2진을 로컬 스트림 파일 시스템에 저장하도록 하십시오.

파일 맵핑을 수행할 수 없으므로 2진(기본 프로그램 및 라이브러리)가 로컬 스트림 파일 시스템에 있는 경우 i5/OS PASE 프로그램을 시작하는 속도는 훨씬 더 느립니다.

여러 개의 fork() 조작을 수행하는 i5/OS PASE에서 어플리케이션을 실행할 경우, 이 어플리케이션은 AIX에서처럼 빨리 실행되지 않습니다. 이는 각 i5/OS PASE fork() 조작이 성능에 상당한 영향을 미칠 수 있는 새 i5/OS 작업을 시작하기 때문입니다.

성능 자료 수집 및 분석에 대한 정보는 시스템 관리 범주에서 성능 주제를 참조하십시오

예

다음은 i5/OS PASE 정보에 제공된 예입니다.

주: 코드 예제를 사용하여 66 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의할 수 있습니다.

ILE 프로그램에서 i5/OS PASE 프로그램 및 프로시듀어 실행

- ILE 프로그램에서 i5/OSPASE 프로그램 실행
- ILE 프로그램에서 i5/OS PASE 프로시듀어 호출

i5/OS PASE 프로그램에서 i5/OS 프로그램 호출

- i5/OS PASE 프로그램에서 ILE 프로시듀어 호출
- i5/OS PASE 프로그램에서 i5/OS 프로그램 호출
- i5/OS PASE에서 CL 명령 실행

i5/OS PASE 프로그램에서 iSeries용 DB2 UDB 함수 사용

- i5/OS PASE 프로그램에서 iSeries용 DB2 UDB 호출 레벨 인터페이스(CLI) 호출

i5/OSPASE 관련 정보

다음은 i5/OSPASE 주제와 관련된 Information Center 주제 및 웹 사이트에 대한 정보입니다.

IBM 레드북™ 및 Redpaper

Bringing PHP to your iSeries server  이 Redpaper에서 설명한 단계별 구현은 PASE(i5/OSPortable Application Solutions Environmenti5/OS)에서 실행되는 하이퍼텍스트 프리프로세서(PHP)의 CGI 버전을 포함합니다.

웹 사이트

- Enablement roadmaps & resources  (http://www.ibm.com/servers/enable/site/porting/index.html)

  이 웹 사이트는 어플리케이션을 iSeries 서버로 이식하는 데 있어서 i5/OS PASE를 다른 솔루션과 비교합니다.

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

  이 웹 사이트는 i5/OS PASE를 사용하여 어플리케이션을 iSeries 서버로 이식하는 데 대한 정보를 제공합니다.

- API 분석 툴  (http://www.ibm.com/servers/enable/site/porting/iseries/overview/apitool.html)

  이 분석 툴은 i5/OS PASE가 어플리케이션의 AIX 명령, API 및 유틸리티 사용을 지원하는 방법에 대한 자세한 정보를 제공합니다.

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

  이 웹 사이트는 AIX 명령 및 유틸리티에 대한 정보를 제공합니다.

뉴스 그룹

i5/OS PASE 뉴스 그룹(news://news.software.ibm.com/ibm.software.iseries.pase)은 i5/OS PASE와 관련된 사용자 질문 및 응답을 설명합니다.

기타 정보

- i5/OS PASE API

  다음과 같은 i5/OS PASE API의 일반 범주에 대한 자세한 내용은 이 주제를 참조하십시오.

  - 호출 가능 프로그램 API
  - ILE 프로시듀어 API
  - i5/OS PASE 프로그램에서 사용할 런타임 함수

i5/OS PASE 프로그램을 실행하려면 시스템 API를 호출해야 합니다. 이 시스템은 i5/OS PASE 프로그램을 실행하기 위해 호출할 수 있는 프로그램 API 및 ILE 프로시듀어 API를 제공합니다. 호출 가능 프로그램 API가 사용하기는 더 쉽지만, ILE 프로시듀어 API에서 사용할 수 있는 모든 제어를 제공하지는 않습니다.

- i5/OS PASE 쉘 및 유틸리티

  i5/OS PASE에는 세 개의 쉘(Korn, Bourne 및 C 쉘) 및 i5/OS PASE 프로그램으로 실행되는 거의 200개의 유틸리티가 있습니다. i5/OS PASE 쉘 및 유틸리티는 여러 산업 표준 및 사실상의 표준 명령을 포함하는 확장 가능한 스크립팅 환경을 제공합니다.

- i5/OS PASE 명령

  이 주제에서 설명한 대부분의 i5/OS PASE 명령은 AIX 명령과 동일한 옵션 및 작동을 제공합니다. i5/OS PASE 명령과 함께 각 i5/OS PASE 쉘은 여러 내장 명령(예: `cd`, `exec` 및 `if`)을 지원합니다.

- i5/OS PASE 런타임 라이브러리

  i5/OS PASE 런타임은 AIX 런타임에서 제공하는 많은 인터페이스 서브세트를 지원합니다. i5/OS PASE가 지원하는 대부분의 런타임 인터페이스는 AIX와 동일한 옵션 및 작동을 제공합니다. i5/OS PASE 런타임 라이브러리는 /usr/lib에 기호 링크로 설치됩니다.

PDF 파일 저장

PDF를 보거나 인쇄하기 위해 워크스테이션에 저장하려면 다음을 수행하십시오.

1. 브라우저에서 PDF를 마우스 오른쪽 버튼으로 클릭하십시오(위의 링크를 마우스 오른쪽 단추로 클릭).
2. PDF를 로컬로 저장하는 옵션을 클릭하십시오.
3. PDF를 저장하려는 디렉토리를 탐색하십시오.
4. **저장**을 클릭하십시오.

Adobe Reader 다운로드

이 PDF를 보거나 인쇄하려면 시스템에 Adobe Reader를 설치해야 합니다. Adobe 웹 사이트 (www.adobe.com/products/acrobat/readstep.html)  에서 무료로 다운로드할 수 있습니다.

코드 라이센스 및 면책사항 정보

IBM은 귀하에게 유사한 기능을 귀하의 특정 요구에 맞게 조정하여 생성할 수 있도록 모든 프로그래밍 코드 예제를 사용할 수 있는 비독점적인 저작권 라이센스를 부여합니다.

강행 법규에 규정된 보증 조항의 적용을 제외하고, IBM은 해당 프로그램 또는 기술 지원에 대한 상품성, 특정 목적에의 적합성 및 타인의 권리 비침해에 대한 묵시적 보증을 포함한(단, 이에 한하지 않음) 일체의 묵시적 또는 명시적인 보증이나 주장도 제공하지 않습니다.

IBM, IBM 프로그램 개발자 또는 공급자는, 손해 발생의 가능성을 통지 받은 경우를 포함한 어떠한 경우에도 다음에 대하여 책임 지지 않습니다.

1. 데이터의 손실 또는 손상
2. 직접적인, 특별한, 우연에 의한 또는 간접적인 손상 또는 이에 따른 경제적 손실 또는
3. 기대했던 이익, 사업, 수익, 영업권 또는 비용 절감이 실현되지 못함으로 인해 발생하는 손해

일부 관할권에서는 부수적 또는 결과적 손해의 제외사항이나 제한사항을 허용하지 않으므로, 상기 제외사항이나 제한사항이 귀하에게 적용되지 않을 수도 있습니다.

부록. 주의사항

이 정보는 미국에서 제공되는 제품과 서비스용으로 작성된 것입니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 묵시적이든 명시적이든 일체의 보증없이 이 책을『현상태대로』제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 이 변경사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(1) 독립적으로 작성된 프로그램과 기타 프로그램(본 프로그램 포함) 간의 정보 교환 및 (2) 교환된 정보의 상호 이용을 목적으로 정보를 원하는 프로그램 라이센스 사용자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조항 및 조건에 따라(예를 들면, 사용료 지불 포함) 사용할 수 있습니다.

이 정보에 기술된 라이센스가 있는 프로그램 및 이 프로그램에 대해 사용 가능한 모든 라이센스가 있는 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA), 기계 코드에 대한 IBM 라이센스 계약 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 성능은 개발 레벨 상태의 시스템에서 측정되었을 수 있으므로 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한, 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 문서의 사용자는 해당 데이터를 사용자의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 기타 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM의 향후 방향 또는 의도에 관한 모든 언급은 별도의 통지없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위해 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원시 언어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 추가 비용없이 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 그러므로 IBM은 이 프로그램들의 신뢰성, 서비스 및 기능을 보장할 수 없습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 그 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.

© (귀하의 회사명) (연도). 이 코드 부분은 IBM Cop. 샘플 프로그램에서 파생됩니다. © Copyright IBM Corp. Copyright IBM Corp. _연도_. All rights reserved.All rights reserved.

이 정보를 소프트카피로 보는 경우에는 사진과 컬러 삽화가 제대로 나타나지 않을 수도 있습니다.

프로그래밍 인터페이스 정보

이 i5/OS PASE 서적은 고객이 IBM i5/OS의 서비스를 얻기 위한 프로그램을 작성하는 데 사용할 수 있는 프로그래밍 인터페이스를 설명합니다.

상표

다음 용어는 미국 또는 기타 국가에서 사용되는 IBM Corporation의 상표입니다.

| AIX
| e(로고)server
| eServeri5/OS
| IBM
| IBM(로고)
| iSeries
| pSeries
| AFS
| DFS
| ILE(Integrated Language Environment)
| NetServer
| PartnerWorld
| POWER
| PowerPC
| DB2
| DB2 Universal Database
| OS/400

Java 및 모든 Java 기반 상표는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc의 상표입니다.

UNIX는 미국 또는 기타 국가에서 사용되는 The Open Group의 등록상표입니다.

Microsoft, Windows, Windows NT 및 Windows 로고는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

기타 회사, 제품 또는 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

조건

다음 조건에 따라 본 발행물을 사용할 수 있습니다.

개인적 사용: 귀하는 모든 소유권 사항을 표시하는 것을 조건으로 본 발행물을 개인적, 비상업적 용도로 복제할 수 있습니다. 귀하는 IBM의 명시적 동의없이 본 발행물 또는 그 일부를 배포 또는 게시하거나 이에 대한 2차적 저작물을 만들 수 없습니다.

상업적 사용: 귀하는 모든 소유권 사항을 표시하는 것을 조건으로 본 발행물을 귀하 사업장 내에서만 복제, 배포 및 게시할 수 있습니다. 귀하의 사업장 외에서는 IBM의 명시적 동의없이 본 발행물의 2차적 저작물을 만들거나 본 발행물 또는 그 일부를 복제, 배포 또는 게시할 수 없습니다.

본 허가에서 명시적으로 부여된 경우를 제외하고, 본 발행물이나 본 발행물에 포함된 정보, 데이터, 소프트웨어 또는 기타 지적 재산권에 대해서는 어떠한 허가나 라이센스 또는 권리도 명시적 또는 묵시적으로 부여되지 않습니다.

IBM은 본 발행물의 사용이 IBM의 이익을 해친다고 판단하거나 위에서 언급된 지시사항이 준수되지 않는다고 판단하는 경우 언제든지 부여한 허가를 철회할 수 있습니다.

귀하는 미국 수출법 및 관련 규정을 포함하여 모든 적용 가능한 법률 및 규정을 철저히 준수하는 것을 조건으로 본 정보를 다운로드, 송신 또는 재송신할 수 있습니다.

IBM은 본 발행물의 내용에 대해 어떠한 보증도 하지 않습니다. IBM은 상품성 및 특정 목적에의 적합성에 대한 보증을 포함하여 명시적이든 묵시적이든 일체의 보증없이 ″현상태대로″ 본 발행물을 제공합니다.