IBM

System i

파일 및 파일 시스템 통합 파일 시스템

버전 6 릴리스 1

IBM

System i

파일 및 파일 시스템 통합 파일 시스템

버전 6 릴리스 1

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 157 페이지의 『주의사항』의 정보를 읽으십시오.

이 개정판은 새 개정판에서 별도로 명시하지 않는 한, IBM i5/OS(제품 번호 5761–SS1) 버전 6, 릴리스 1, 수정 0 및 모든 후속 릴리스와 수정에 적용됩니다. 이 버전은 일부 축약 명령어 세트 컴퓨터(RISC) 모델 및 CICS 모델에서도 실행되지 않습니다.

목차

통합 파일 시스템

통합 파일 시스템은 i5/OS® 오퍼레이팅 시스템의 일부로서 퍼스널 컴퓨터 및 UNIX® 오퍼레이팅 시스템과 유사한 스트림 입/출력 및 기억장치 관리를 지원하며 시스템에 저장된 모든 정보에 대해 통합된 구조를 제공합니다.

주: 해당 코드 예제를 사용하는 것은 155 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

V6R1의 새로운 사항

통합 파일 시스템 주제 콜렉션에서 새로운 또는 많이 변경된 정보에 대해 읽으십시오.

System i™ Navigator를 사용하여 폴더 속성 수집 및 검색하기

디렉토리 정보 검색(RTVDIRINF) 및 디렉토리 정보 인쇄(PRTDIRINF) 명령에 대한 대체 기능으로 System i Navigator는 통합 파일 시스템 내 오브젝트의 속성을 수집 및 분석하는 기능을 제공합니다.

세부사항은 92 페이지의 『System i Navigator를 사용하여 폴더 속성 수집 및 검색하기』의 내용을 참조하십시오.

개선된 체크 인 및 체크 아웃 기능

하나의 파일뿐만 아니라 한 폴더에 있는 모든 대상 오브젝트를 체크 인 및 체크 아웃할 수 있습니다.

세부사항은 148 페이지의 『오브젝트 체크 인』 및 149 페이지의 『오브젝트 체크 아웃』의 내용을 참조하십시오.

이름 내 추가 문자 사용을 지원하기 위한 자동 변환

V6R1 i5/OS 오퍼레이팅 시스템 설치 직후 시스템은 이름 내에서 새로운 유니코드 문자 및 새로운 대소문자 구분 규칙을 지원하기 위해 대소문자를 구분하지 않는 파일 시스템 내 디렉토리를 변환하기 시작합니다.

세부사항은 103 페이지의 『추가 문자 지원을 위한 이름 변환』의 내용을 참조하십시오.

IPv6 지원

다음 파일 시스템은 i5/OS V6R1 이전 릴리스에서는 IPv4만 지원했었지만 현재는 IPv4 및 IPv6을 모두 지원합니다.

- 58 페이지의 『i5/OS NetClient 파일 시스템(QNTC)』
- 64 페이지의 『i5/OS 파일 서버 파일 시스템(QFileSvr.400)』
- 69 페이지의 『네트워크 파일 시스템(NFS)』

일부 CL 명령에서 유니코드 사용 가능

일부 CL 명령에서 유니코드를 사용 가능합니다. 자세한 정보는 74 페이지의 『CL 명령을 사용한 액세스』에서 테이블을 참조하십시오.

QNetWare 파일 시스템 지원 중단

i5/OS V6R1부터 QNetWare 파일 시스템 및 관련 기능은 통합 파일 시스템에서 지원되지 않습니다. 그래서 QNetWare 관련 정보는 이 서적에서 삭제되었습니다.

새롭거나 변경된 사항을 보는 방법

기술 변경사항이 작성된 위치를 알아보는 데 도움을 얻기 위해 Information Center를 사용합니다.

- 새롭거나 변경된 정보가 시작되는 위치를 표시하기 위한 » 이미지
- 새롭거나 변경된 정보가 끝나는 위치를 표시하기 위한 « 이미지

PDF 파일에서 새로 추가되거나 변경된 정보의 좌측 여백에 막대(|)를 보실 수 있습니다.

이 릴리스의 새로운 사항이나 변경된 사항에 대한 다른 정보를 찾으려면 사용자 메모를 참조하십시오.

통합 파일 시스템에 대한 PDF 파일

이 정보의 PDF 파일을 보거나 인쇄할 수 있습니다.

이 문서의 PDF 버전을 보거나 다운로드하려면 통합 파일 시스템(약 1845KB)을 참조하십시오.

PDF 파일 저장

보거나 인쇄하기 위해 워크스테이션에서 PDF를 저장하려면 다음을 수행하십시오.

1. 브라우저의 해당 PDF 링크에서 마우스 오른쪽 버튼을 누르십시오.
2. 로컬로 PDF를 저장하는 옵션을 클릭하십시오.
3. PDF를 저장하려는 디렉토리를 탐색하십시오.
4. **저장**을 클릭하십시오.

Adobe Acrobat Reader 다운로드

이 PDF를 보거나 인쇄하려면 Adobe® Reader가 필요합니다. Adobe 웹 사이트 (www.adobe.com/products/acrobat/readstep.html) 에서 무료로 사본을 다운로드할 수 있습니다.

통합 파일 시스템 개요

i5/OS 오퍼레이팅 시스템의 통합 파일 시스템에 대해 설명하고 서버에서 어떻게 사용되는지 설명합니다.

통합 파일 시스템의 개념

*통합 파일 시스템*은 i5/OS 오퍼레이팅 시스템의 일부로서 PC 및 UNIX 오퍼레이팅 시스템과 유사한 스트림 입/출력 및 기억장치 관리를 지원하며 시스템에 저장된 모든 정보에 대해 통합된 구조를 제공합니다.

통합 파일 시스템은 10개의 파일 시스템으로 구성되며, 각 파일 시스템에는 기억장치에 있는 정보와 대화식 작업을 하기 위한 논리 구조 및 규칙 세트가 들어 있습니다.

통합 파일 시스템의 주요 기능은 다음과 같습니다.

- 긴 연속 데이터열이 들어 있는 스트림 파일에 정보 저장 지원. 예를 들어, 이러한 데이터열은 그림의 영상 요소나 문서의 텍스트가 될 수 있습니다. 스트림 파일 지원은 클라이언트 서버 어플리케이션에서 효율적인 사용을 위해 고안되었습니다.
- 오브젝트가 나무 가지의 열매처럼 구성되도록 하는 계층적 디렉토리 구조. 오브젝트의 디렉토리를 포함한 경로를 지정하여 오브젝트를 액세스할 수 있습니다.
- 사용자 및 어플리케이션이 스트림 파일, 데이터베이스 파일, 문서 및 기타 시스템 내 저장된 오브젝트에 액세스할 수 있도록 하는 공통 인터페이스입니다.
- 시스템, 통합 xSeries® 서버(IXS) 또는 리모트 Windows NT® 서버에 로컬로 저장된 스트림 파일의 공통 보기입니다. 또한 스트림 파일은 근거리 통신망(LAN) 서버, 다른 System i 제품 또는 네트워크 파일 시스템(NFS) 서버에 리모트로 저장될 수 있습니다.

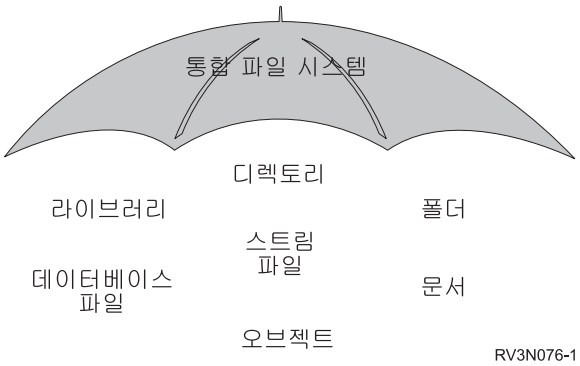

그림 1. i5/OS 오퍼레이팅 시스템에 저장된 모든 정보의 구조

관련 개념

26 페이지의『파일 시스템』

*파일 시스템*은 논리 장치(LU)를 구성하는 요소인 기억장치의 특정 세그먼트에 액세스할 수 있도록 지원합니다. 시스템에서 논리 장치는 파일, 디렉토리, 라이브러리 및 오브젝트입니다.

통합 파일 시스템 사용 이유

통합 파일 시스템은 i5/OS의 광범위한 데이터 관리 기능과 더불어 클라이언트/서버, 개방 시스템 및 멀티미디어와 같은 차세대 새로운 정보 처리 양식 지원을 향상시킨 것입니다.

통합 파일 시스템을 사용하여 다음을 수행할 수 있습니다.

- i5/OS 데이터(특히, i5/OS 파일 서버를 사용하는 System i Access와 같은 어플리케이션의 경우)에 대한 빠른 액세스를 제공할 수 있습니다.
- 이미지, 오디오 및 비디오와 같은 스트림 데이터를 보다 효율적으로 처리할 수 있습니다.
- POSIX(Portable Operationg System Interface for Computer Environments) 및 XPG와 같은 UNIX 오퍼레이팅 시스템 기반의 개방 시스템 표준을 지원하기 위한 파일 시스템 기반과 디렉토리 기반을 제공할 수 있습니다. 또한 이 파일 구조와 이 디렉토리 구조는 DOS 및 Microsoft Windows® 오퍼레이팅 시스템과 같은 PC 오퍼레이팅 시스템 사용자에게 익숙한 환경을 제공합니다.
- 고유한 기능(예: 레코드 지향 데이터베이스 파일, UNIX 오퍼레이팅 시스템 기반 스트림 파일 및 파일 제공)을 가진 파일 지원이 별도의 파일로 처리되도록 하면서 이들 모두가 공통 인터페이스를 통해 관리되도록 할 수 있습니다.

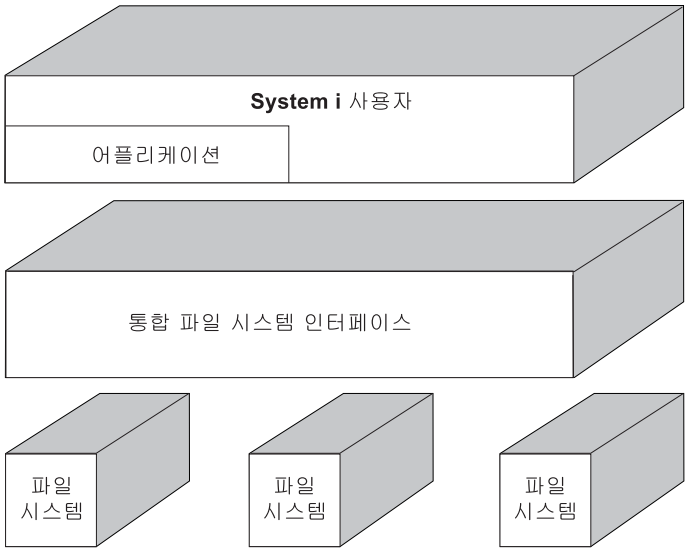

그림 2. 개별 파일 시스템에 대한 공통 인터페이스

- PC 사용자가 그래픽 사용자 인터페이스의 장점을 보다 잘 이용할 수 있도록 할 수 있습니다. 예를 들어, Windows 사용자는 Windows 그래픽 툴을 사용하여 PC에 저장된 파일에 대해 작업할 때와 동일한 방식으로 i5/OS 스트림 파일과 기타 오브젝트에 대해 작업할 수 있습니다.
- 자국어간에 연관된 오브젝트 정보를 제공하고 오브젝트명의 연속성을 제공합니다. 예를 들면, 한 언어의 코드 페이지에서 다른 언어의 코드 페이지로 전환될 때에도 개별 문자가 동일하게 남아 있게 됩니다.

관련 개념

26 페이지의『파일 시스템』

*파일 시스템*은 논리 장치(LU)를 구성하는 요소인 기억장치의 특정 세그먼트에 액세스할 수 있도록 지원합니다. 시스템에서 논리 장치는 파일, 디렉토리, 라이브러리 및 오브젝트입니다.

통합 파일 시스템 개념

이 주제에서는 디렉토리, 링크, 경로명, 스트림 파일, 이름 연속성, 확장 속성 및 스캐닝 지원과 같은 통합 파일 시스템의 기본 개념을 소개합니다.

디렉토리

*디렉토리*는 사용자가 지정한 이름으로 오브젝트를 찾는 데 사용되는 특수 오브젝트입니다. 각 디렉토리에는 이에 연결된 오브젝트 리스트가 들어 있습니다. 해당 리스트에 다른 디렉토리가 포함될 수 있습니다.

통합 파일 시스템은 시스템의 모든 오브젝트에 액세스할 수 있게 하는 계층 디렉토리 구조를 제공합니다. 이 디렉토리 구조는 루트 디렉토리가 맨 위에 있고 분기 디렉토리가 아래에 있는 역 트리라고 생각할 수 있습니다. 분기는 디렉토리 계층의 디렉토리를 표시합니다. 이러한 디렉토리 분기에는 서브디렉토리라고 하는 종속 분기가 있습니다. 다양한 디렉토리 및 서브디렉토리 분기에는 파일과 같은 오브젝트가 연결되어 있습니다. 오브젝트는 디렉토리를 통하여 오브젝트가 연결된 서브디렉토리에 대해 경로를 지정함으로써 찾을 수 있습니다. 특정 디렉토리에 연결된 오브젝트는 경우에 따라 그 디렉토리 *안에* 있다고 기술됩니다.

모든 종속 분기(서브디렉토리) 및 그 분기에 연결된 모든 오브젝트와 함께 특정 디렉토리 분기를 *서브트리*라고 합니다. 각 파일 시스템은 통합 파일 시스템 디렉토리 구조의 주요 서브트리입니다. QSYS.LIB 및 독립 ASP QSYS.LIB 파일 시스템의 서브트리에서 라이브러리는 서브디렉토리와 같은 방식으로 처리됩니다. 라이브러리의 오브젝트는 서브디렉토리의 오브젝트와 같이 처리됩니다. 데이터베이스 파일에 오브젝트(데이터베이스 파일 멤버)가 포함되어 있으므로 이들 파일은 오브젝트가 아닌 서브디렉토리인 것처럼 처리됩니다. 문서 라이브러리 서비스 파일 시스템(QDLS 서브트리)에서 폴더는 서브디렉토리와 같이 처리되고 폴더의 문서는 서브디렉토리의 오브젝트와 같이 처리됩니다.

파일 시스템 간에 차이가 있으므로 디렉토리 계층의 한 서브트리에서 수행할 수 있는 작업이 다른 서브트리에서는 작동되지 않을 수도 있습니다.

통합 파일 시스템 디렉토리 지원은 DOS 파일 시스템에서 제공하는 디렉토리 지원과 유사합니다. 또한 파일을 한 번만 저장하지만 링크를 사용하여 여러 경로를 통해 액세스할 수 있는 것과 같은 UNIX 시스템의 전형적인 기능을 제공합니다.

파일 시스템 및 오브젝트는 통합 파일 시스템 디렉토리 트리상의 분기입니다. 통합 파일 시스템 디렉토리 트리의 예는 다음 그림을 참조하십시오.

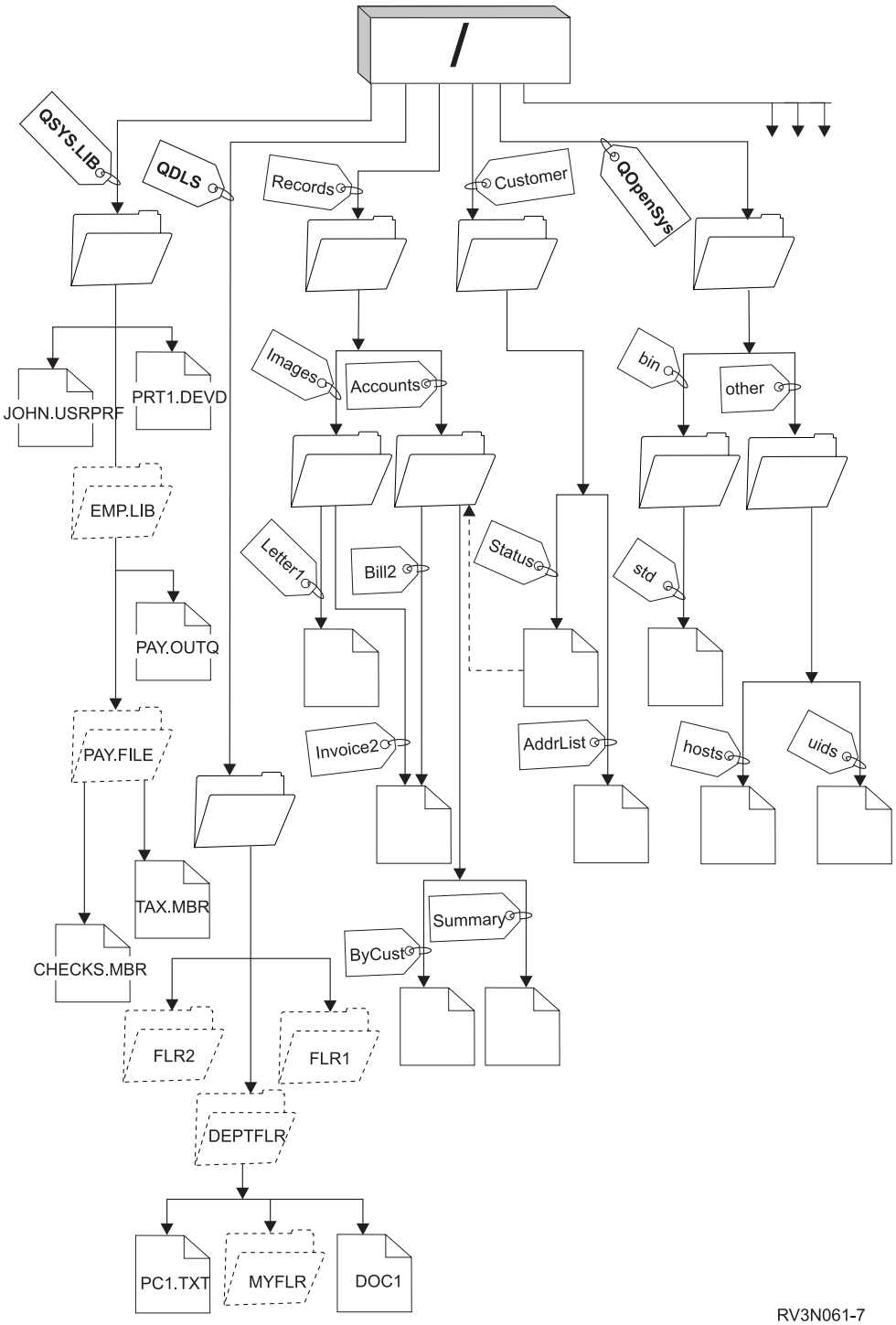

그림 3. 샘플 통합 파일 시스템 디렉토리 트리

현재 디렉토리

현재 디렉토리는 현재 라이브러리 개념과 유사합니다. 또한, *현재 작업 디렉토리* 또는 *작업 디렉토리*라고도 합니다.

*현재 디렉토리*는 오퍼레이팅 시스템이 사용자의 프로그램 및 파일을 찾고 임시 파일 및 출력을 저장하는 첫 번째 디렉토리입니다. 사용자가 파일과 같은 오브젝트에 대한 조작을 요청하면 시스템은 사용자가 다른 디렉토리 경로를 지정하지 않는 한 사용자의 현재 디렉토리에서 오브젝트를 탐색합니다.

홈 디렉토리

*홈 디렉토리*는 사용자가 시스템에 사인 온할 때 현재 디렉토리로 사용됩니다. 홈 디렉토리명은 사용자 프로파일에서 지정됩니다.

작업이 시작되면 시스템은 사용자 프로파일에서 홈 디렉토리명을 찾습니다. 시스템에 해당 디렉토리명이 없는 경우, 홈 디렉토리는 『루트』(/) 디렉토리로 변경됩니다.

일반적으로, 사용자에 대한 사용자 프로파일을 작성하는 시스템 관리자가 사용자의 홈 디렉토리도 작성합니다. /home 디렉토리 아래에 각 사용자에 대한 개별 홈 디렉토리를 작성하는 것이 바람직합니다. /home 디렉토리는 『루트』(/) 디렉토리의 서브디렉토리입니다. 시스템 디폴트는 사용자의 홈 디렉토리명이 사용자 프로파일과 같은 이름으로 기대합니다.

예를 들어, CRTUSRPRF USRPRF(John) HOMEDIR(*USRPRF) 명령은 John에 대한 홈 디렉토리를 /home/JOHN으로 할당합니다. /home/JOHN 디렉토리가 존재하지 않으면 "루트"(/) 디렉토리가 John에 대한 홈 디렉토리가 됩니다.

CHGCURDIR(현재 디렉토리 변경) CL 명령, chdir( ) API 또는 fchdir() API를 사용하여 사인 온한 후 언제든지 홈 디렉토리 이외의 디렉토리를 현재 디렉토리로 지정할 수 있습니다.

프로세스 초기 설정시에 선택된 홈 디렉토리는 각 스레드의 홈 디렉토리로 계속 사용됩니다. 이 규칙은 스레드에 대한 현재 사용자 프로파일이 초기 설정 이후 변경된 경우에도 마찬가지로 적용됩니다. 그러나 작업 변경(QWTCHGJB) API를 사용하면 스레드의 현재 홈 디렉토리를 스레드의 현재 사용자 프로파일에 지정된 홈 디렉토리(또는 홈 디렉토리가 없는 경우 "루트"(/) 디렉토리)로 변경할 수 있습니다. 두 번째 스레드는 자신을 작성한 스레드의 홈 디렉토리를 상속합니다. QWTCHGJB를 사용하여 스레드의 홈 디렉토리를 변경하더라도 프로세스의 현재 디렉토리는 변경되지 않는다는 점에 유의하십시오. 현재 디렉토리는 프로세스 레벨에 속하며, 홈 디렉토리는 스레드 레벨에 속합니다. 스레드에서 현재 작업 디렉토리를 변경하면 프로세스 전체의 작업 디렉토리가 변경됩니다. 스레드에 대한 홈 디렉토리를 변경하더라도 현재 작업 디렉토리는 변경되지 않습니다.

관련 정보

CHGCURDIR(현재 디렉토리 변경) 명령

chdir()--현재 디렉토리 변경 API

fchdir()--설명자로 현재 디렉토리를 변경하는 API

어플리케이션 프로그램 인터페이스(API)

제공되는 디렉토리

| 시스템이 재시작될 때 통합 파일 시스템은 여기에 나열된 디렉토리가 존재하지 않는다면 이를 생성합니다. 시
| 스템에 의해 생성된 후 이 디렉토리를 이동시키거나 이름 변경하지 않아야 합니다.

주: 다음 시스템 작성 디렉토리를 다른 오브젝트로의 기호 링크로 대체하지 마십시오. 예를 들어, /home을 독립 ASP의 디렉토리로의 기호 링크로 대체하지 마십시오. 그렇지 않으면 새 사용자 프로파일을 작성할 때 문제점이 발생할 뿐 아니라 독립 ASP에도 문제점이 발생할 수 있습니다.

/tmp /tmp 디렉토리는 어플리케이션에 임시 오브젝트를 저장할 공간을 제공합니다. 이 디렉토리는 『루트』(/) 디렉토리의 서브디렉토리이므로 경로명은 /tmp입니다.

어플리케이션이 /tmp 디렉토리에 파일을 넣으면 오브젝트는 사용자나 해당 어플리케이션이 제거할 때까지 존재합니다. 시스템은 /tmp 디렉토리에서 자동으로 오브젝트를 제거하거나 /tmp 디렉토리에 있는 오브젝트에 대해 특별한 처리를 수행하지 않습니다.

통합 파일 시스템을 지원하는 사용자 화면 및 명령을 사용하여 /tmp 디렉토리 및 해당 오브젝트를 관리할 수 있습니다. 예를 들어, 오브젝트 링크에 대한 작업 표시 화면이나 WRKLNK 명령을 사용하여 /tmp 디렉토리나 디렉토리의 오브젝트를 복사, 제거 또는 이름 변경할 수 있습니다. 모든 사용자들은 이 디렉토리에 대해 *ALL 권한을 가지며, 이것은 디렉토리에 대해 대부분의 유효한 조치를 수행할 수 있다는 의미입니다.

어플리케이션은 통합 파일 시스템을 지원하는 API(Application Programming Interface)를 사용하여 /tmp 디렉토리 및 해당 오브젝트를 관리할 수 있습니다. 예를 들어, 어플리케이션 프로그램이 unlink() API를 사용하여 /tmp 디렉토리의 오브젝트를 제거할 수 있습니다.

/tmp 디렉토리가 제거되면 다음에 시스템이 재시작되는 동안 자동으로 다시 작성됩니다.

오퍼레이팅 시스템 공통성 및 보안 목적으로 /tmp 디렉토리의 제한된 이름 변경 및 링크 해제 속성을 예로 설정할 수 있습니다.

주: 제한된 이름 변경 및 링크 해제 속성은 디렉토리의 S_ISVTX 모드 비트와 동일합니다.

제한된 이름 변경 및 링크 해제 속성이 예로 설정되면 다음 조건 중 하나가 참이 되지 않는 한 /tmp 디렉토리 내의 오브젝트의 이름을 변경하거나 링크를 해제할 수 없습니다.

- 오브젝트의 소유자입니다.
- 디렉토리의 소유자입니다.
- 모든 오브젝트(*ALLOBJ) 특수 권한을 가집니다.

이 속성이 예로 설정되고 적절한 권한을 가지지 않는 경우 다음 명령 및 API 사용 시 이름 변경 및 링크 해제 실패를 나타내는 오류 번호 3027(EPERM) 또는 메세지 MSGCPFA0B1(요구된 조작이 허용되지 않습니다. 액세스 문제점)이 표시됩니다.

- RMVLNK, DEL 및 ERASE(링크 제거) 명령
- RMVDIR, RD 및 RMDIR(디렉토리 제거) 명령
- RNM 및 REN(오브젝트 이름 변경) 명령
- MOV 및 MOVE(오브젝트 이동) 명령
- 파일 또는 디렉토리 이름 변경(rename()) API
- 파일 또는 디렉토리 이름 변경, 존재하는 경우 "새롭게" 유지(Qp0lRenameKeep()) API
- 파일 또는 디렉토리 이름 변경, 존재하는 경우 "새롭게" 링크 해제(Qp0lRenameUnlink()) API

- 디렉토리 제거(rmdir()) API
- 파일에 대한 링크 제거(unlink()) API

오브젝트의 소유자이거나 모든 오브젝트(*ALLOBJ) 특수 권한을 가지는 경우 CHGATR(속성 변경) 명령 또는 속성 설정(Qp0lSetAttr())이나 파일 권한 변경(chmod) API를 사용하여 제한된 이름 변경 및 링크 해제 속성과 S_ISVTX 모드 비트를 수정할 수 있습니다. 그러나 속성이 아니오로 변경되면 예 설정이 제공하는 오퍼레이팅 시스템 공통성 및 보안 장점을 잃게 됩니다.

시스템 재시작 시 /tmp 디렉토리가 작성되면 이 속성이 예로 설정됩니다. 시스템 재시작 시 /tmp 디렉토리가 이미 존재하는 경우 속성이 변경되지 않습니다.

/home 시스템 관리자는 /home 디렉토리를 사용하여 모든 사용자를 위한 개별 디렉토리를 저장합니다. 시스템 관리자는 종종 사용자 프로파일과 연관된 홈 디렉토리를 /home내의 사용자 디렉토리로 설정합니다. 예를 들면 /home/john과 같은 경우입니다.

/etc /etc 디렉토리는 관리, 구성 및 기타 시스템 파일들을 저장합니다.

/usr /usr 디렉토리는 시스템이 사용하는 정보를 가지고 있는 서브디렉토리를 포함하고 있습니다. /usr의 파일은 일반적으로 자주 변경되지 않는 파일들입니다.

/usr/bin

/usr/bin 디렉토리에는 표준 유틸리티 프로그램이 있습니다.

/QIBM

/QIBM 디렉토리는 시스템 디렉토리이며 시스템과 함께 제공됩니다.

/QIBM/ProdData

/QIBM/ProdData 디렉토리는 라이센스 프로그램 데이터용으로 사용되는 시스템 디렉토리입니다.

/QIBM/UserData

/QIBM/UserData 디렉토리는 구성 파일과 같은 라이센스 프로그램 사용자 데이터용으로 사용되는 시스템 디렉토리입니다.

/QOpenSys/QIBM

/QOpenSys/QIBM 디렉토리는 QOpenSys 파일 시스템용 시스템 디렉토리입니다.

/QOpenSys/QIBM/ProdData

/QOpenSys/QIBM/ProdData 디렉토리는 QOpenSys 파일 시스템용 시스템 디렉토리이며 라이센스 프로그램 데이터용으로 사용됩니다.

/QOpenSys/QIBM/UserData

/QOpenSys/QIBM/UserData 디렉토리는 QOpenSys 파일 시스템용 시스템 디렉토리이며 구성 파일과 같은 라이센스 프로그램 사용자 데이터용으로 사용됩니다.

/asp_name/QIBM

/asp_name/QIBM 디렉토리는 사용자 시스템에 있는 독립 ASP용 시스템 디렉토리입니다. 여기서 asp_name은 독립 ASP 이름입니다.

/asp_name/QIBM/UserData

/asp_name/QIBM/UserData 디렉토리는 사용자 시스템에 있는 독립 ASP의 구성 파일과 같은 라이센스 프로그램 사용자 데이터용으로 사용되는 시스템 디렉토리입니다. 여기서 asp_name은 독립 ASP 이름입니다.

/dev /dev 디렉토리는 여러 시스템 파일 및 디렉토리를 포함합니다.

/dev/xti

/dev/xti 디렉토리는 UDP 및 TCP 장치 드라이버를 포함합니다.

관련 개념

7 페이지의 『홈 디렉토리』

*홈 디렉토리*는 사용자가 시스템에 사인 온할 때 현재 디렉토리로 사용됩니다. 홈 디렉토리명은 사용자 프로파일에서 지정됩니다.

관련 참조

35 페이지의 『"루트"(/) 파일 시스템에서 UDP 및 TCP 장치』

/dev/xti 디렉토리 아래의 『루트』 파일 시스템은 이제 udp 및 tcp라고 명명된 두 개의 장치 드라이버를 보유합니다.

35 페이지의 『개방 시스템 파일 시스템(QOpenSys)』

QOpenSys 파일 시스템은 POSIX 및 XPG(X/Open Portability Guide)와 같은 UNIX 기반의 개방 시스템 표준과 호환될 수 있습니다. "루트"(/) 파일 시스템과 마찬가지로 이 파일 시스템은 통합 파일 시스템에서 제공하는 스트림 파일 및 디렉토리 지원을 이용합니다.

관련 정보

WRKLNK(오브젝트 링크에 대한 작업) 명령

*TYPE2 디렉토리

통합 파일 시스템의 "루트"(/), QOpenSys 및 사용자 정의 파일 시스템(UDFS)은 *TYPE2 디렉토리 형식을 지원합니다. *TYPE2 디렉토리 형식은 기존의 *TYPE1 디렉토리 형식을 개선한 것입니다.

주: *TYPE1 및 *TYPE2 스트림 파일의 개념은 *TYPE1 및 *TYPE2 디렉토리 형식의 개념과 다릅니다. 서로 관련되어 있지 않습니다.

*TYPE2 디렉토리의 내부 구조 및 구현은 *TYPE1 디렉토리와 다릅니다.

*TYPE2 디렉토리의 장점은 다음과 같습니다.

- 향상된 성능
- 향상된 신뢰성
- 추가된 기능
- 대부분의 경우, 줄어든 보조 기억장치 공간

*TYPE2 디렉토리는 특히 디렉토리 작성 및 삭제 시 *TYPE1 디렉토리보다 향상된 파일 시스템 성능을 보여줍니다.

*TYPE2 디렉토리는 *TYPE1 디렉토리보다 신뢰성이 높습니다. 시스템이 비정상적으로 종료되는 경우, *TYPE2 디렉토리는 보조 기억장치 실패가 발생하지 않는 한 완전히 복구됩니다. *TYPE1 디렉토리가 완전히 복구되기 위해서는 RCLSTG(기억장치 재생) 명령을 사용해야 합니다.

*TYPE2 디렉토리는 다음과 같은 추가 기능을 제공합니다.

- *TYPE2 디렉토리는 모노케이스 파일 시스템에서 이름의 대소문자 변경을 지원합니다(예: A를 a로 이름 변경).
- *TYPE2 디렉토리의 오브젝트는 *TYPE1 디렉토리 오브젝트가 32,767개의 링크를 가질 수 있는 데 비해 최대 백만 개의 링크를 가질 수 있습니다. 이는 스트림 파일에 대해 최대 백만 개의 하드 링크를 가질 수 있으며 *TYPE2 디렉토리에 최대 999,998개의 서브디렉토리가 포함될 수 있다는 것을 의미합니다.
- System i Navigator를 사용하면 *TYPE2 형식의 디렉토리를 열 때 항목 리스트가 2진 순서로 자동으로 정렬됩니다.
- 통합 파일 시스템 스캐닝 지원과 같은 일부 새로운 기능은 *TYPE2 디렉토리의 오브젝트에만 사용할 수 있습니다.

일반적으로 350개 미만의 오브젝트가 포함된 *TYPE2 디렉토리는 같은 수의 오브젝트가 포함된 *TYPE1 디렉토리보다 적은 보조 기억장치를 필요로 합니다. 350개 이상의 오브젝트가 포함된 *TYPE2 디렉토리는 *TYPE1 디렉토리보다 평균적으로 10% 더 큽니다.

시스템에서 *TYPE2 디렉토리를 만드는 몇 가지 방법이 있습니다.

- OS/400® V5R2 또는 i5/OS V5R3 이상으로 사전 설치된 새로운 System i 플랫폼에는 *TYPE2 디렉토리가 있습니다. ASP 1-32의 ″루트″(/), QOpenSys 및 UDFS의 경우에는 변환이 필요하지 않습니다.
- System i 플랫폼에 처음으로 OS/400 V5R2 또는 i5/OS V5R3 또는 그 이후 버전을 설치한 경우 이 플랫폼에는 *TYPE2 디렉토리가 있습니다. ASP 1-32의 ″루트″(/), QOpenSys 및 UDFS의 경우에는 변환이 필요하지 않습니다.
- V5R2 변환 유틸리티는 파일 시스템을 변환하는 데 사용됩니다. 변환 유틸리티에 대한 자세한 정보는 V5R2 iSeries Information Center의 *TYPE2 디렉토리로 변환 섹션을 참조하십시오.
- 독립 ASP의 UDFS가 아직 *TYPE2 형식으로 변환되지 않은 경우 독립 ASP가 OS/400 V5R2 또는 i5/OS V5R3 이상이 설치된 시스템으로 처음으로 연결변환될 때 UDFS가 변환됩니다.
- UDFS를 제외하고 여전히 *TYPE1 디렉토리를 사용 중인 독립 ASP의 다른 모든 지원 파일 시스템은 시스템에서 자동으로 변환합니다. 이 변환은 i5/OS V5R3 이상 릴리스 설치 후에 시작됩니다. 이 변환이 시스템 활동에 상당한 영향을 주어서는 안됩니다.

시스템의 파일 시스템에 대한 디렉토리 형식을 판별하려면 CVTDIR(디렉토리 변환) 명령을 사용하십시오.

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

주: *TYPE2 디렉토리는 OS/400 V5R2 또는 i5/OS V5R3 이상에서 지원되지만 보통 *TYPE2 디렉토리 지원과는 약간 다릅니다.

관련 참조

97 페이지의 『디렉토리를 *TYPE1에서 *TYPE2로 변환』
통합 파일 시스템의 "루트"(/), QOpenSys 및 사용자 정의 파일 시스템(UDFS)은 *TYPE2 디렉토리 형식을 지원합니다.

관련 정보

RCLSTG(기억장치 재생) 명령

CVTDIR(디렉토리 변환) 명령

링크

*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

디렉토리 기반 파일 시스템을 가진 사용자의 경우, 파일과 같은 오브젝트를 시스템에 이를 식별하는 이름을 가진 것으로 생각하는 것이 편리합니다. 사실상, 그것이 오브젝트를 식별하는 오브젝트로의 디렉토리 경로입니다. 때로는 오브젝트의 *이름*을 부여하기만 해도 그 오브젝트에 액세스할 수 있습니다. 이는 시스템이 일정한 조건 아래에서 경로의 디렉토리 부분이라고 간주하도록 고안되었기 때문입니다. 링크라는 개념은 오브젝트를 식별하는 디렉토리 경로인 실체를 이용합니다. 따라서, 오브젝트라기 보다는 링크에 이름이 부여됩니다.

오브젝트가 아니라 링크에 이름이 부여된다는 생각에 익숙해지면 이전에는 숨겨져 있던 가능성을 볼 수 있게 됩니다. 같은 오브젝트에 여러 링크가 있을 수 있습니다. 예를 들어, 두 사용자가 각 사용자의 홈 디렉토리로부터 파일로 링크를 가짐으로써 파일을 공유할 수 있습니다(7 페이지의 『홈 디렉토리』 참조). 특정 링크 유형은 파일 시스템 전체에 사용될 수 있고, 오브젝트가 없어도 존재할 수 있습니다.

링크에는 두 가지 유형인 하드 링크와 기호 링크가 있습니다. 프로그램 내에서 경로명을 사용할 때 하드 링크나 기호 링크를 선택할 수 있습니다. 링크 유형은 각각 장단점을 가지고 있습니다. 다음은 링크의 한 유형이 다른 유형에 비해 가지는 장점을 표시한 것입니다.

표 1. 하드 링크와 기호 링크의 비교

항목	하드 링크	기호 링크
이름 해결	보다 빠름. 하드 링크에는 오브젝트에 대한 직접적인 참조가 포함됩니다.	보다 느림. 기호 링크에는 오브젝트 찾기를 위해 해결해야 하는 오브젝트에 대한 경로명이 포함됩니다.
오브젝트 존재	필수. 오브젝트로 하드 링크를 작성하기 위해서는 오브젝트가 있어야 합니다.	선택적. 언급된 오브젝트가 존재하지 않을 때 기호 링크가 작성될 수 있습니다.
오브젝트 삭제	제한적. 오브젝트로의 모든 하드 링크는 연결해제(제거)되어 오브젝트를 삭제합니다.	무제한적. 참조하는 기호 링크가 있는 경우에도 오브젝트가 삭제될 수 있습니다.
정적 오브젝트(속성이 변경되지 않음)	보다 빠름. 정적 오브젝트인 경우, 이름 해상도가 1차적인 성능 요건입니다. 이름 해상도는 하드 링크 사용 시 보다 빨라집니다.	보다 느림. 이름 해상도는 기호 링크 사용 시 보다 느려집니다.
범위	제한적. 하드 링크는 파일 시스템간에 사용될 수 없습니다.	무제한적. 기호 링크는 파일 시스템간에 사용될 수 있습니다.

하드 링크

*하드 링크*는 때때로 단지 링크라고만 하며 실제 오브젝트로 링크되지 않는 한 존재할 수 없습니다.

디렉토리에 오브젝트가 작성될 때(예를 들면, 디렉토리로 파일을 복사하여), 첫 번째 하드 링크가 디렉토리와 오브젝트 사이에 성립됩니다. 사용자와 어플리케이션 프로그램은 다른 하드 링크를 추가할 수 있습니다. 각 하드 링크는 디렉토리 내의 별도 디렉토리 항목에 의해 표시됩니다. 동일한 디렉토리로부터의 링크는 동일한 이름을 가질 수 없으나, 다른 디렉토리로부터의 링크는 동일한 이름을 가질 수 있습니다.

파일 시스템에 의해 지원되는 경우, 동일한 디렉토리나 서로 다른 디렉토리에서 하나의 오브젝트로 여러 개의 하드 링크가 있을 수 있습니다. 한 가지 예외는 오브젝트가 또 다른 디렉토리인 경우입니다. 디렉토리부터 다른 디렉토리로는 오직 하나의 하드 링크만 있을 수 있습니다.

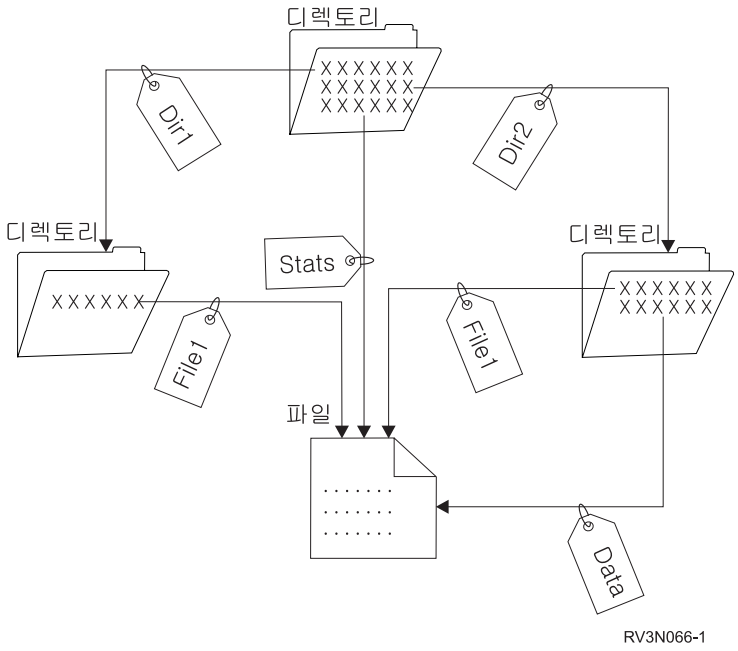

그림 4. 디렉토리 항목은 각 하드 링크를 정의합니다.

오브젝트에 적어도 하나의 하드 링크가 남아 있는 한, 하드 링크는 오브젝트의 존재에 영향을 주지 않고 제거될 수 있습니다. 최종 하드 링크가 제거될 때, 어플리케이션에 오브젝트가 열려 있지 않으면 오브젝트는 시스템에서 제거됩니다. 열린 오브젝트를 가진 각 어플리케이션은 오브젝트가 닫힐 때까지 계속 사용할 수 있습니다. 오브젝트를 사용하는 최종 어플리케이션에 의해 오브젝트가 닫힐 때, 오브젝트가 시스템에서 제거됩니다. 오브젝트는 최종 하드 링크가 제거된 후에는 열리지 않습니다.

또한 하드 링크의 개념은 QSYS.LIB 또는 독립 ASP QSYS.LIB 파일 시스템 및 문서 라이브러리 서비스(QDLS) 파일 시스템에 적용될 수 있지만 제한이 있습니다. 라이브러리에는 사실상 라이브러리 내의 각 오브젝트에 대해 하나의 링크가 있습니다. 마찬가지로, 폴더에는 폴더 내의 각 문서에 대한 하나의 하드 링크가 있습니다. 그러나 QSYS.LIB, 독립 ASP QSYS.LIB 또는 QDLSMultiple 하드 링크에서는 *같은 오브젝트*에 대해 복수 하드 링크가 허용되지 않습니다.

하드 링크는 파일 시스템간에 사용될 수 없습니다. 예를 들어, QOpenSys 파일 시스템의 디렉토리는 QSYS.LIB 또는 독립 ASP QSYS.LIB 파일 시스템의 오브젝트 또는 QDLS 파일 시스템의 문서에 대한 하드 링크를 가질 수 없습니다.

기호 링크

소프트 링크라고도 하는 *기호 링크*는 파일에 포함된 경로명입니다.

시스템이 기호 링크를 만나게 되면 기호 링크가 제공하는 경로명을 따르게 되고, 그 후 기호 링크 뒤에 오는 나머지 경로를 계속 진행합니다. 경로명이 /로 시작되는 경우, 시스템은 /(『루트』) 디렉토리로 리턴하여 그 지점으로부터의 경로를 따릅니다. 경로명이 /로 시작하지 않은 경우, 시스템은 바로 앞의 디렉토리로 리턴하여 그 디렉토리에서 시작하는 기호 링크의 경로명을 따릅니다.

다음의 기호 링크 사용 방법 예를 참조하십시오.

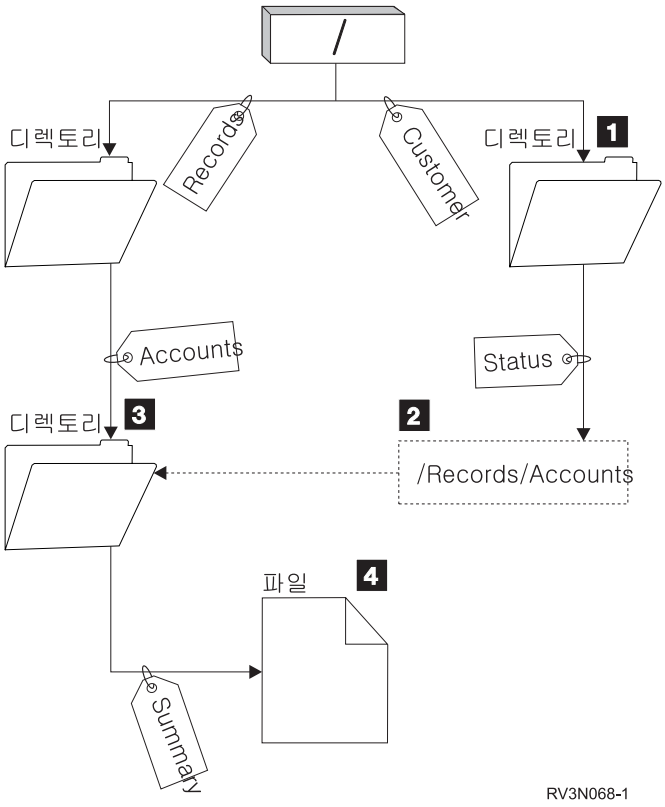

그림 5. 기호 링크 사용 예

고객 계정 상태를 보여주는 메뉴 옵션을 선택하십시오. 메뉴를 표시하는 프로그램은 다음 경로명을 사용합니다.

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

시스템은 디렉토리 1로 가는 *Customer* 링크를 따라간 다음 *Status* 링크를 따라갑니다. 경로명 2를 가지고 있는 *Status* 링크는 기호 링크입니다. 경로명이 /로 시작되기 때문에 시스템은 /(『루트』) 디렉토리로 리턴하여 차

례로 *Records*와 *Accounts* 링크를 따라갑니다. 이 경로는 다른 디렉토리 3으로 갑니다. 이제 시스템은 프로그램이 제공하는 경로명의 경로를 완료합니다. 사용자가 필요로 하는 데이터가 들어 있는 파일 4로 가는 *Summary* 링크를 따릅니다.

하드 링크와 달리 기호 링크는 오브젝트(오브젝트 유형 *SYMLNK)로서, 존재하는 오브젝트를 지정하지 않고 존재할 수 있습니다. 예를 들어, 나중에 추가되거나 교체될 파일로의 경로를 제공하기 위해 기호 링크를 사용할 수도 있습니다.

또한 하드 링크와 달리 기호 링크는 파일 시스템간에 사용될 수 있습니다. 예를 들어, 한 파일 시스템에서 작업 중인 경우 기호 링크를 사용하여 다른 파일 시스템의 파일에 액세스할 수 있습니다. QSYS.LIB, 독립 ASP QSYS.LIB 및 QDLS 파일 시스템이 기호 링크의 작성 및 저장을 지원하지 않지만 다음 기능을 지원하는 ″루트″(/) 또는 QOpenSys 파일 시스템에 기호 링크를 작성할 수 있습니다.

- QSYS.LIB 또는 독립 ASP QSYS.LIB 파일 시스템의 데이터베이스 파일 멤버에 액세스할 수 있습니다.
- QDLS 파일 시스템의 문서에 액세스할 수 있습니다.

경로명

경로명(일부 시스템에서는 *pathname*이라고도 함)은 서버에 오브젝트를 찾는 방법을 알려줍니다.

경로명은 일련의 디렉토리명과 오브젝트명으로 표현됩니다. 개별 디렉토리 및 오브젝트명은 다음과 같이 슬래시(/) 문자로 구분됩니다.

```
디렉토리1/디렉토리2/파일
```

사용자의 편의를 위해 통합 파일 시스템에서 슬래시(/) 대신 백슬래시(₩)를 사용할 수 있습니다.

경로명을 지정하기 위해서는 다음 두 가지 방법이 사용됩니다.

- *절대 경로명*은 최고 레벨에서 또는『루트』디렉토리(/ 문자로 식별됨)에서 시작합니다. 예를 들어, / 디렉토리로부터 Smith란 파일명까지의 다음 경로를 생각해 보십시오.

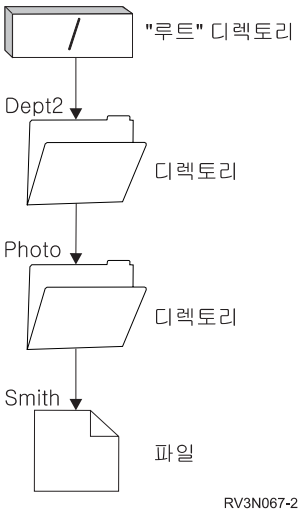

그림 6. 경로명의 구성요소

Smith 파일의 절대 경로명은 다음과 같습니다.

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

절대 경로명을 *전체 경로명*이라고도 합니다.

- 경로명이 / 문자로 시작하지 않으면 시스템은 경로가 사용자의 현재 디렉토리에서 시작된다고 간주합니다. 이러한 유형의 경로명을 *상대 경로명*이라고 합니다. 예를 들어, 사용자의 현재 디렉토리가 Dept2이고 Smith라는 파일이 들어 있는 Photo란 서브디렉토리가 있는 경우 파일의 상대 경로명은 다음과 같습니다.

  ```
  Photo/Smith
  ```

  경로명에 현재 디렉토리가 포함되어 있지 않다는 것에 유의하십시오. 이름의 첫 번째 항목은 *아래에서 다음 단계*의 오브젝트 또는 디렉토리입니다.

  관련 참조

  129 페이지의『API에 대한 경로명 규칙』
  오브젝트에 대해 조작하기 위해 통합 파일 시스템 또는 ILE C API를 사용하는 경우 디렉토리 경로를 제공하여 오브젝트를 식별합니다. 다음은 API에 경로명을 지정할 때 유의해야 할 규칙을 요약한 것입니다.

  77 페이지의『CL 명령 및 표시장치에 대한 경로명 규칙』
  통합 파일 시스템 명령이나 화면을 사용하여 오브젝트에 대해 조작할 때 경로명을 제공하여 오브젝트를 식별합니다.

스트림 파일

*스트림 파일*은 단순하게 바이트가 나열된 구조로 랜덤 액세스가 가능합니다.

통합 파일 시스템은 스트림 파일 형태의 정보에 대한 조작 및 저장을 지원합니다. 시스템의 폴더에 저장된 문서는 스트림 파일입니다. 스트림 파일의 다른 예로는 UNIX 시스템의 파일 및 PC 파일이 있습니다. 통합 파일 시스템 스트림 파일은 오브젝트 유형이 *STMF인 시스템 오브젝트입니다.

스트림 파일을 보다 잘 이해하려면 i5/OS 데이터베이스 파일과 비교하는 것이 도움이 됩니다. 데이터베이스 파일은 레코드를 중심으로 이루어지며, 데이터베이스 파일에는 길이 및 데이터 유형과 같은 특성을 가진 하나 이상의 필드로 구성된 사전 정의 부속 영역이 있습니다.

그림 7. 스트림 파일과 레코드 지향 파일의 비교

스트림 파일과 레코드 지향 파일은 구조 자체가 다르며, 이러한 구조상의 차이는 파일이 사용되는 방식에 영향을 미칩니다. 이 구조는 어플리케이션이 파일과의 상호작용을 위해 작성되는 방식 및 각 유형의 파일이 어플리케이션의 어디에서 가장 잘 사용되는지에 영향을 미칩니다. 예를 들어, 레코드 지향 파일은 이름, 주소 및 손익계산표와 같은 고객 통계치를 저장하기에 적합합니다. 레코드 지향 파일은 시스템의 확장 프로그래밍 기능을 사용하여 이들 사전 정의된 필드에 개별적으로 액세스하여 이를 조작할 수 있습니다. 그러나 스트림 파일은 다양한 색상을 표시하며 연속적인 비트열로 구성되어 있는 고객 사진과 같은 정보를 저장하기에 보다 적합합니다. 스트림 파일은 특히 문서 텍스트, 이미지, 오디오 및 비디오와 같은 데이터 스트림을 저장하기에 매우 적합합니다.

파일 형식에는 *TYPE1 스트림 파일 및 *TYPE2 스트림 파일의 두 가지 옵션이 있습니다. 파일 형식은 파일이 작성된 릴리스에 의존하거나, 파일이 사용자 정의 파일 시스템(해당 파일 시스템에 지정된 값)에 작성됩니다.

주: *TYPE1 및 *TYPE2 스트림 파일의 개념은 *TYPE1 및 *TYPE2 디렉토리 형식의 개념과 다릅니다. 서로 관련되어 있지 않습니다.

*TYPE1 스트림 파일

*TYPE1 스트림 파일은 OS/400 V4R4 이전의 릴리스에서 작성된 스트림 파일과 같은 형식을 가집니다.

*TYPE1 스트림 파일은 약 128GB(1GB는 약 1,073,741,824바이트와 같음)의 최대 오브젝트 *크기*를 가집니다.

*TYPE2 스트림 파일

*TYPE2 스트림 파일은 고성능 파일 액세스를 가집니다.

*TYPE2 스트림 파일은 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템에서 오브젝트 크기를 약 1TB(1TB는 약 1,099,511,627,776바이트와 같음)까지 허용합니다. 그렇지 않으면 최대값은 약 256GB입니다. 또한 주 기억장치 할당을 최적화하기 위해 속성을 지정하는 기능뿐 아니라 메모리를 맵핑하는 기능도 있습니다. *TYPE1의 파일 형식을 지정한 사용자 정의 파일 시스템에서 스트림 파일이 작성되지 않는 한 OS/400 V4R4 및 그 이후 시스템에서 작성된 모든 파일은 *TYPE2 스트림 파일입니다.

주: 256GB보다 대용량 파일은 i5/OS V5R3 이전 시스템으로 복원되거나 저장될 수 없습니다.

이름 연속성

『루트』(/), QOpenSys 및 사용자 정의 파일 시스템을 사용할 때 오브젝명의 문자가 동일하게 남아 있도록 하는 시스템 지원을 이용할 수 있습니다.

이는 시스템간에 이 파일 시스템을 사용할 때와 다른 문자 코드화 체계(코드 페이지)를 갖는 연결된 장치 간에 이 파일 시스템을 사용할 때도 적용됩니다. 시스템은 *TYPE1 디렉토리에 대해서는 UCS2 레벨 1(*유니코드(Unicode)*라고도 함)*TYPE2 디렉토리에 대해서는 UTF-16이라는 16비트 형식 이름으로 문자를 저장합니다. UCS2 레벨 1 및 UTF-16은 ISO 10646 표준의 한 부분입니다. 이름 사용 시 시스템은 저장된 문자 형식을 사용 중인 코드 페이지의 적절한 문자 표시로 변환합니다. 또한 각 오브젝트와 연관된 확장 속성명도 동일한 방식으로 처리됩니다.

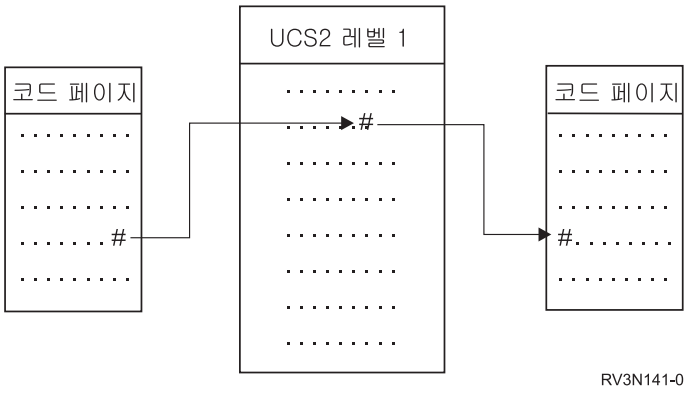

그림 8. 코드화 체계 간 문자의 동일성

이러한 지원으로 서로 다른 코드 페이지를 사용하는 장치에서 시스템과 보다 쉽게 대화할 수 있습니다. 예를 들어, PC에 시스템과 동일한 코드 페이지가 없더라도 PC 사용자는 동일한 파일명을 사용하여 i5/OS 파일에 액세스할 수 있습니다. 한 코드 페이지에서 다른 코드 페이지로의 변환은 시스템에 의해 자동으로 처리됩니다. 물론, 장치는 이름에 사용된 문자가 들어 있는 코드 페이지를 사용해야 합니다.

관련 개념

10 페이지의『*TYPE2 디렉토리』
통합 파일 시스템의 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템(UDFS)은 *TYPE2 디렉토리 형식을 지원합니다. *TYPE2 디렉토리 형식은 기존의 *TYPE1 디렉토리 형식을 개선한 것입니다.

103 페이지의『자동 이름 변환 개요』
″″루트″″(/)와 같이 대소문자 구분을 하지 않는 파일 시스템 및 CASE(*MONO)로 생성된 UDFS는 유니코드 표준 4.0으로 저장된 이름을 지원합니다. 시스템은 이름에 포함된 추가 문자를 지원하기 위해 자동 이름 변환을 실행합니다.

확장 속성

확장 속성은 오브젝트에 대한 추가 세부사항을 제공하는 오브젝트와 연관된 정보입니다. 확장 속성은 참조하기 위해 사용되는 이름 및 값으로 구성됩니다. 값은 텍스트, 2진 데이터 또는 다른 유형의 데이터가 될 수 있습니다.

오브젝트에 대한 확장 속성은 오브젝트가 있는 한 존재합니다.

확장 속성은 다양한 방식으로 이루어지며 다양한 정보를 포함하도록 사용될 수 있습니다. 특히 다음 세 가지 확장 속성에 대해 알아야 합니다.

.SUBJECT
오브젝트의 내용 또는 목적에 대한 간단한 설명

.TYPE
오브젝트 내의 데이터 유형. 데이터 유형에는 텍스트, 2진, 프로그램 소스, 컴파일된 프로그램 및 기타 정보가 있습니다.

.CODEPAGE
오브젝트에 대해 사용될 코드 페이지. 오브젝트에 대해 사용되는 코드 페이지가 해당 오브젝트와 연관된 확장 속성에 대해서도 사용됩니다.

이름의 첫 번째 문자로서의 마침표(.)는 확장 속성이 시스템 사용을 위해 예약된 표준 시스템 확장 속성(SEA)이라는 의미입니다.

다양한 파일 시스템의 다양한 오브젝트가 확장 속성에 존재할 수도 있고 그렇지 않을 수도 있습니다. QSYS.LIB 및 독립 ASP QSYS.LIB 파일 시스템은 세 가지 사전 정의된 확장 속성 .SUBJECT, .TYPE 및 .CODEPAGE를 지원합니다. 문서 라이브러리 서비스(QDLS) 파일 시스템에서 폴더 및 문서에는 모든 종류의 확장 속성이 있을 수 있습니다. 일부 폴더 및 문서에는 확장 속성이 있고 일부에는 없을 수 있습니다.『루트』(/), QOpenSys 및 사용자 정의 파일 시스템, 모든 디렉토리, 스트림 파일 및 기호 링크는 모든 종류의 확장 속성을 포함할 수 있습니다. 그러나 일부는 확장 속성을 전혀 가질 수 없습니다.

WRKLNK(오브젝트 링크에 대한 작업) 명령 및 DSPLNK(오브젝트 링크 표시) 명령을 사용하여 오브젝트에 대한 .SUBJECT 확장 속성을 표시할 수 있습니다. 어플리케이션이나 사용자가 확장 속성에 액세스하고 변경할 수 있는 다른 통합 파일 시스템 지원은 없습니다. 이 규칙에 대한 유일한 예외는 DSPUDFS(UDFS 표시) 및 DSPMFSINF(마운트된 파일 시스템 정보 표시) CL 명령으로 이것은 사용자에게 확장 속성을 표시해줍니다.

그러나 QDLS에 있는 일부 오브젝트와 연관된 확장 속성은 계층 파일 시스템(HFS)에서 제공하는 인터페이스를 통해 변경될 수 있습니다.

클라이언트 PC가 OS/2® 또는 Windows 오퍼레이팅 시스템을 통해 System i 플랫폼에 연결된 경우 각 오퍼레이팅 시스템의 프로그래밍 인터페이스(예: DosQueryFileInfo 및 DosSetFileInfo)를 사용하여 파일 오브젝트의 확장 속성을 쿼리하고 설정할 수 있습니다. 또한 OS/2 사용자는 설정 노트북을 사용하여, 다시 말해 오브젝트와 연관된 메뉴에서 설정을 선택하여 데스크탑에 있는 오브젝트의 확장 속성을 변경할 수 있습니다.

확장 속성을 정의하는 경우, 다음 명명 규칙을 사용하십시오.

- 확장 속성명은 최대 255자가 될 수 있습니다.
- 이름의 첫 번째 문자로서 마침표(.)를 사용하지 마십시오. 마침표로 시작하는 확장 속성은 표준 시스템 확장 속성으로 해석됩니다.
- 이름이 상충되는 것을 최소화하려면 일관된 확장 속성 명명 구조를 사용하십시오. 다음 형식을 사용하는 것이 좋습니다.

```
CompanyNameProductName.Attribute_Name
```

스캐닝 지원

i5/OS 오퍼레이팅 시스템을 사용하여 통합 파일 시스템 오브젝트를 스캔할 수 있습니다.

이 지원은 다양한 항목에 대해 스캔을 허용하여 사용자에 대한 유연성을 생성합니다. 사용자가 스캔 발생 시기 및 스캔의 결과에 따라 취할 조치를 결정합니다.

이 지원에 관련된 두 가지 종료점을 다음과 같습니다.

- QIBM_QP0L_SCAN_OPEN - 열기 종료 프로그램에 대한 통합 파일 시스템 스캔

  이 종료점의 경우, 통합 파일 시스템 오브젝트가 특정 조건 하에서 열릴 때 열기 종료 프로그램에 관한 통합 파일 시스템 스캔이 스캔 처리를 수행하도록 호출됩니다.
- QIBM_QP0L_SCAN_CLOSE - 닫기 종료 프로그램에 대한 통합 파일 시스템 스캔

  이 종료점의 경우, 통합 파일 시스템 오브젝트가 특정 조건 하에서 닫힐 때 닫기 종료 프로그램에 관한 통합 파일 시스템 스캔이 스캔 처리를 수행하도록 호출됩니다.

주: *TYPE2 디렉토리로 완전하게 변환된 파일 시스템의 오브젝트만이 스캔됩니다.

관련 태스크

147 페이지의『오브젝트가 스캔되어야 하는지 여부 설정』

″루트″(/), QOpenSys 및 사용자 정의 파일 시스템에서 오브젝트를 스캔할 수 있는지 여부를 지정할 수 있습니다. 이 스캔 옵션 설정을 하려면 다음 단계를 수행하십시오.

관련 정보

열기 종료 프로그램에 대한 통합 파일 시스템 스캔

닫기 종료 프로그램에 대한 통합 파일 시스템 스캔

예: 바이러스 및 열린 파일 스캔

다음 예는 종료 프로그램에서 스캔할 수 있는 사항을 나타냅니다.

- 바이러스

  종료 프로그램은 바이러스에 대해 스캔할 수 있습니다. 파일에 바이러스가 있으면 antivirus 프로그램은 문제를 치료하거나 바이러스를 격리시키려고 시도함으로써 조치할 수 있습니다. System i 플랫폼 자체는 바이러스에 감염되지 않았으므로 이를 이용해 서버 간 바이러스 전송을 감소시킬 수 있습니다.

- 언제 파일이 열렸는지 알기 위해 호출

  언제 파일이 열렸는지 알아보기 위해 스캔할 수도 있습니다. 이 스캔을 규정함으로써 특정 파일이 액세스된 날짜와 시간을 추적할 수 있습니다. 이것은 특정 사용자의 작동을 추적하려할 때 유용합니다.

스캔은 시스템 값이 설정되는 방법과 스캔 환경이 설정되는 방법에 따라 다른 두 시간에 발생할 수 있습니다. 다음 리스트는 발생한 시간에 따른 여러 종료의 스캐닝에 대해 설명한 것입니다.

1. 런타임 스캐닝

   런타임 스캔은 보통 일일 활동 중의 파일 또는 파일들에 관한 스캔입니다. 이것은 액세스될 때마다 파일의 무결성을 확인합니다. 보통 활동 중의 스캐닝은 파일 또는 파일들이 사용자가 스캐닝하고 있는 어떤 표준이든지 간에 보증합니다.

   바이러스에 대한 런타임 스캔 예

   사용자의 PC에서 통합 파일 시스템의 파일에 액세스하기로 선택했습니다. PC에서 파일이 열릴 때 파일이 스캔됩니다. 열기 종료 프로그램이 등록되고 QSCANFS 시스템 값이 ″루트″(/), QOpenSys 및 UDFS 파일 시스템에서 스캔 파일로 설정되기 때문입니다. 스캔은 하나의 바이러스가 발견되었음을 표시하며 안티바이러스 종료 프로그램이 문제점을 치료합니다. 종료 프로그램이 파일을 치료하면 파일은 감염되지 않은 상태로 됩니다. 따라서 PC로부터의 액세스가 감염되지 않고 감염을 확산할 수 없습니다.

   이제 해당 액세스에 대해 바이러스를 스캔하는 대신에 런타임 스캔을 수행하지 않도록 선택했다고 하십시오. 사용자의 PC에서 감염된 파일에 액세스하면 바이러스가 사용자 PC로 전송될 것입니다. 런타임 스캔을 사용하여 바이러스가 사용자 PC에 퍼지기 전에 감지할 수 있습니다.

   이 방법의 주요한 단점은 스캔을 수행하는 데 자원 시간이 필요하다는 것입니다. 파일에 액세스하려는 사용자는 파일을 사용하기 전에 스캔이 완료될 때까지 기다려야 합니다. 시스템은 스캐닝이 액세스마다 수행되는 것이 아니라 필요한 경우에만 수행되도록 합니다.

2. 대량 또는 수동으로 활성화되는 스캐닝

   복수 항목을 동시에 스캔하려는 경우 이 옵션을 사용할 수 있습니다. 이 예에서는 주말과 같이 시스템이 오프라인 시 스캔이 발생하도록 설정할 수 있습니다. 이것은 정상적인 일일 활동 중에 파일에 액세스하는 데 거의 영향을 주지 않습니다. 스캔은 오프라인으로 수행됩니다. 따라서 대량 스캔이 완료된 후에 변경되지 않는 파일에 대한 런타임 스캔 오버헤드를 줄일 수 있습니다. 이러한 파일에 다시 액세스할 때 재스캔이 필요하지 않기 때문입니다.

관련 개념

『관련 시스템 값』

시스템에 대해 원하는 스캐닝 환경을 설정하기 위해 QSCANFS 및 QSCANFSCTL 시스템 값을 사용할 수 있습니다.

관련 정보

열기 종료 프로그램에 대한 통합 파일 시스템 스캔

닫기 종료 프로그램에 대한 통합 파일 시스템 스캔

관련 시스템 값

시스템에 대해 원하는 스캐닝 환경을 설정하기 위해 QSCANFS 및 QSCANFSCTL 시스템 값을 사용할 수 있습니다.

스캔 관련 시스템 값의 이름과 각각에 대한 설명이 아래에 나열됩니다. 이 시스템 값과 제어 옵션은 System i Navigator에 대해 설명합니다. 비교 가능한 문자 기반 인터페이스 값은 System i Navigator 이름 뒤의 괄호에 나열됩니다. 예를 들어, 시스템 값 QSCANFSCTL의 경우 System i Navigator에서 제어 옵션 파일 서버만을 통한 액세스 스캔이 선택되면 본질적으로 문자 기반 제어 옵션 *FSVRONLY를 지정하여 같은 결과를 작성할 것입니다.

시스템 값의 이름 및 설명은 다음과 같습니다.

1. 등록된 종료 프로그램을 사용하여 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템(QSCANFS) 스캔

   이 시스템 값은 파일 시스템이 스캔되어야 하는지 여부를 지정하는 데 사용될 수 있습니다. 파일 시스템이 완전하게 변환된 경우 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템의 오브젝트만 스캔됩니다. 이 값은 통합 파일 시스템 스캔 관련 종료점과 함께 등록된 종료 프로그램에 의해 오브젝트가 스캔되어야 하는지를 지정합니다.

   디폴트 값은 종료 프로그램이 등록될 때 오브젝트가 스캔되는 것입니다.

2. 제어 스캔(QSCANFSCTL)

   이 시스템 값으로 디폴트 제어 옵션을 사용하거나 지정된 제어 옵션을 사용할 수 있습니다. System i Navigator 시스템 값에 기반한 지정된 다른 제어 옵션에 대한 간략한 설명은 아래를 참조하십시오.

   - 파일 서버만을 통한 액세스 스캔(*FSVRONLY가 지정됨)

     파일 서버에서 System i 플랫폼에 액세스하는 경우 유일하게 스캔이 발생합니다. 이 옵션을 선택하지 않은 경우 모든 액세스가 스캔됩니다.

   - 종료 프로그램이 실패한 경우 요청 실패(*ERRFAIL이 지정됨)

     종료 프로그램이 호출될 때 오류가 있으면 종료 프로그램의 호출을 트리거하는 요청 또는 조작이 실패합니다. 이 옵션을 선택하지 않으면 시스템은 실패한 종료 프로그램을 건너뛰고 오브젝트가 스캔되지 않은 것처럼 처리합니다.

   - 쓰기 액세스 업그레이드 수행(*NOWRTUPG가 지정되지 않음)

액세스 업그레이드는 쓰기 액세스를 포함하는 종료 프로그램에 전달되는 스캔 설명자를 위해 발생합니다. *NOWRTUPG 옵션을 선택하지 않으면 시스템은 쓰기 액세스 업그레이드 수행을 시도하지 않습니다.

*NOWRTUPG를 지정하면 시스템은 쓰기 액세스를 포함하는 종료 프로그램에 전달되는 스캔 설명자를 위해 액세스 업그레이드를 시도하지 **않습니다**. *NOWRTUPG를 지정하지 않으면 시스템은 쓰기 액세스 업그레이드 수행을 시도합니다.

- 제어 스캔에 '오브젝트가 변경되었을 때만' 속성 사용(*USEOCOATR가 지정됨)

  '오브젝트 변경만' 속성(오브젝트가 수정된 경우 오브젝트 스캔만)이 사용됩니다. 이 옵션을 선택하지 않으면 이 속성은 사용되지 않으며 오브젝트가 수정된 이후 및 스캔 소프트웨어가 갱신을 표시할 때 오브젝트가 스캔됩니다.
- 닫기 중 스캔이 실패한 경우 닫기 요청 실패(*NOFAILCLO가 지정되지 않음)

  오브젝트가 닫기 처리 중에 스캔에 실패하면 닫기 요청이 실패합니다. 이 옵션을 선택하지 않으면 닫기 요청은 실패하지 않습니다. 선택되지 않은 경우 이 값은 '종료 프로그램이 실패한 경우 요청 실패' 값의 스펙을 대체합니다.

  *NOFAILCLO가 지정되면 오브젝트가 닫기 처리의 부분으로 완료된 스캔에 실패했더라도 시스템은 스캔 실패의 표시와 함께 닫기 요청에 실패하지 **않습니다**.
- 오브젝트가 복원된 이후 다음 액세스 시 스캔(*NOPOSTRST가 지정되지 않음)

  오브젝트는 복원된 후에 스캔됩니다. '오브젝트가 스캔되지 않음' 속성이 지정되면 오브젝트는 복원된 후에 한 번 스캔됩니다. '오브젝트 변경만' 속성이 지정되면 오브젝트는 복원된 후에 스캔됩니다.

  오브젝트가 복원되는 동안 *NOPOSTRST를 지정하면 오브젝트가 복원되었으므로 스캔되지 않습니다. 오브젝트 속성이 '오브젝트가 스캔되지 않음'이면 오브젝트는 언제든지 스캔되지 않습니다. 오브젝트 속성이 '오브젝트 변경만'이면 오브젝트는 복원된 후에 수정된 경우 스캔됩니다.

관련 정보

보안 시스템 값: 등록된 종료 프로그램을 사용하여 루트(/), QOpenSys 및 사용자 정의 파일 시스템 스캔

보안 시스템 값: 스캔 제어

스캐닝 발생

스캐닝은 다양한 이유로 발생할 수 있습니다. 다음은 스캔이 발생하는 시기와 이유에 대한 정보입니다.

오브젝트의 현재 스캔 상태 및 속성을 보기 위해 WRKLNK(오브젝트 링크에 대한 작업) 명령, DSPLNK(오브젝트 링크 표시) 명령, 속성 가져오기(Qp0lGetAttr()) API 또는 System i Navigator의 등록 정보 페이지를 사용할 수 있습니다.

관련 정보

WRKLNK(오브젝트 링크에 대한 작업) 명령

DSPLNK(오브젝트 링크 표시) 명령

Qp0lGetAttr()--속성 얻기 API

오브젝트 변경:

오브젝트가 변경되거나 수정된 후에 오브젝트가 액세스된 경우 스캔이 발생할 수 있습니다.

일반적으로 수정은 오브젝트의 데이터에서 발생합니다. 오브젝트에 대한 수정 예는 메모리 맵핑, 오브젝트 절단 또는 오브젝트 지우기를 통해 또는 직접 오브젝트를 작성하는 것입니다. 오브젝트의 CCSID 속성을 변경하는 경우 다음 액세스에 대한 스캔이 트리거됩니다.

서명 변경:

글로벌 서명이 오브젝트의 서명과 다르면 오브젝트가 액세스될 때 스캔이 발생합니다.

글로벌 또는 독립 ASP 그룹 서명은 스캔 관련 종료 프로그램과 연관된 소프트웨어 레벨을 나타냅니다. 오브젝트 서명은 오브젝트가 마지막으로 스캔될 때 글로벌 또는 독립 ASP 서명을 반영합니다. 오브젝트가 독립 ASP 그룹에 없으면 오브젝트 서명은 글로벌 스캔 서명과 비교됩니다. 오브젝트가 독립 ASP에 있으면 오브젝트 서명은 연관된 독립 ASP 그룹 스캔 서명과 비교됩니다.

주: 다음 예에서는 문구 스캔 키 및 스캔 키 서명이 사용됩니다. 스캔 키는 스캐닝 소프트웨어의 한 개 세트를 식별하는 방법입니다. 이 예는 특정 회사에 있습니다. 스캔 키 서명은 스캐닝 소프트웨어의 세트가 제공하는 지원 레벨을 표시할 수 있게 합니다. 하나의 예는 일련의 바이러스 정의입니다.

다음은 오브젝트가 독립 ASP 그룹에 없고 스캔이 발생할 때의 예입니다.

1. 종료 프로그램이 QIBM_QP0L_SCAN_OPEN 종료점에 등록됩니다. 스캔 키와 스캔 키 서명은 다음과 같이 지정됩니다.

   스캔 키: XXXXXX
   스캔 키 서명: 0000000000

   글로벌 스캔 서명은 0000이고 갱신되지 않습니다.
2. 종료 프로그램이 QIBM_QP0L_SCAN_CLOSE 종료점에 등록됩니다. 스캔 키와 스캔 키 서명은 다음과 같이 지정됩니다.

   스캔 키: XXXXXX
   스캔 키 서명: 1111111111

   글로벌 스캔 서명은 0001로 갱신됩니다.
3. 다음, 현재 0000의 오브젝트 서명을 가지고 있는 파일이 열립니다. 글로벌 스캔 서명(0000 ~ 0001)의 차이점과 결부된 종료 프로그램의 존재는 스캔을 시작합니다. 스캔이 정상적으로 완료되면 파일 서명은 0001로 갱신됩니다.
4. 다른 사용자가 파일을 열면 오브젝트 및 글로벌 서명 일치 이후 재스캔되지 않습니다.

아래 예는 종료 프로그램이 재스캔을 발생시키는 원인을 나타냅니다.

1. 지원이 새로운 유형의 바이러스에 대해 스캔하기 위해 시스템에 추가되었습니다. 스캔 서명 변경(QP0LCHSG) API는 스캔 키의 스캔 키 서명을 갱신하도록 호출됩니다. 스캔 키와 스캔 키 서명은 다음과 같이 지정됩니다.

   스캔 키: XXXXXX
   스캔 키 서명: 2222222222

   글로벌 스캔 키 서명은 0002로 갱신됩니다.
2. 이전에 스캔된 파일이 지금 열리면 서명의 차이점은 재스캔의 원인이 됩니다.

예는 언제 오브젝트가 독립 ASP 그룹에 있는지 보여주기 위해 계속됩니다.

1. 독립 ASP는 처음으로 연결변환되며 독립 ASP의 파일이 열립니다. 첫 번째 파일이 열리면 독립 ASP 스캔 키 리스트는 시스템 스캔 키 리스트와 비교됩니다. 독립 ASP 스캔 키 리스트가 없다는 사실 때문에 두 개는 다릅니다. 이 경우 독립 ASP 스캔 키 리스트는 글로벌 스캔 키 리스트를 얻습니다. 독립 ASP 스캔 키 리스트는 XXXXXX의 스캔 키 및 2222222222의 스캔 키 서명을 가집니다. 그 결과 독립 ASP 스캔 서명은 0001로 변경됩니다. 독립 ASP의 파일이 열리면 현재 0000의 오브젝트 서명을 가지며, 이는 0001의 독립 ASP 스캔 서명과 비교되고 차이 때문에 파일이 스캔됩니다. 정상적으로 스캔되면 파일 서명은 0001로 갱신됩니다.

주: 오브젝트가 '오브젝트 변경만' 속성을 가지고 있고 *USEOCOATR 시스템 값이 지정되지 않는 한 서명 변경은 스캔을 트리거합니다.

관련 정보

열기 종료 프로그램에 대한 통합 파일 시스템 스캔

닫기 종료 프로그램에 대한 통합 파일 시스템 스캔

스캔 서명 변경(QP0LCHSG) API

다른 CCSID:

오브젝트가 이전에 해당 오브젝트에 대해 스캔된 것과 다른 CCSID(코드화 문자 세트 ID)로 액세스되면 스캔이 트리거될 수 있습니다.

이 스캔 예는 CCSID 819에 저장된 데이터가 있는 파일이 CCSID 1200에서 열리고 정상적으로 스캔된 것입니다. 파일의 데이터가 변경되지 않는 한 그 파일이 CCSID 1200에서 열릴 때마다 스캔이 트리거되지 않습니다. 그러나 그 파일이 다른 CCSID에서 열리면(예: 37) 스캔은 CCSID 37에 대해 트리거됩니다. 그 스캔이 정상적이면 CCSID 1200 및 37을 가진 후속 액세스는 추가 스캔을 트리거하지 않습니다.

단지 두 CCSID와 하나의 2진 표시는 시스템에 저장된 데이터를 최소화하기 위한 노력을 계속합니다. 일반적으로 다른 많은 CCSID를 가진 같은 오브젝트에 액세스하면 많은 추가 스캐닝이 발생할 수 있습니다.

저장 조작 중:

이는 스캔이 발생할 수 있는 시기의 다른 예를 제공합니다. 오브젝트가 저장될 때 스캔이 요청될 수 있습니다.

이제 SAV(오브젝트 저장) 명령이 저장할 때 파일이 스캔되는지에 대한 스펙을 허용하는 SCAN 매개변수를 포함합니다. 또한 이전에 실패한 스캔이 있거나 저장 중에 스캔이 실패했으면 오브젝트에 대한 요청은 저장되지 않습니다. 실패한 이 파일 때문에 스캔은 매체에 넣지 못하며 아마도 다른 시스템으로 이동됩니다.

주: 이것은 오브젝트가 복원된 시기를 의미하지 않으며 스캔된 것처럼 표시됩니다. 오브젝트가 복원될 때마다 전제 스캔 상태 이력이 지워집니다.

관련 정보

SAV(오브젝트 저장) 명령

오브젝트 무결성 검사:

마지막으로 오브젝트 무결성 검사(CHKOBJITG) 명령에 대한 SCANFS 매개변수가 *YES 값으로 지정되면 스캔이 요청될 수 있습니다.

파일을 열지 않고 적합한지 판별하려면 이 옵션이 적합합니다. SCANFS(*STATUS)가 지정되면 이전 스캔에 실패한 모든 오브젝트가 스캔 실패 위반을 기록합니다.

관련 정보

CHGOBJITG(오브젝트 무결성 변경) 명령

파일 시스템

*파일 시스템*은 논리 장치(LU)를 구성하는 요소인 기억장치의 특정 세그먼트에 액세스할 수 있도록 지원합니다. 시스템에서 논리 장치는 파일, 디렉토리, 라이브러리 및 오브젝트입니다.

각 파일 시스템에는 기억장치에 있는 정보와 대화식 작업을 하기 위한 논리 구조 및 규칙 세트가 들어 있습니다. 이러한 구조 및 규칙은 파일 시스템마다 다릅니다. 사실상 구조와 규칙의 측면에서 보면 라이브러리를 통해 데이터베이스 파일과 기타 다양한 오브젝트 유형에 액세스하기 위한 i5/OS 지원은 하나의 파일 시스템으로 간주될 수 있습니다. 마찬가지로 폴덜르 통해 실제로는 스트림 파일인 문서에 액세스하기 위한 i5/OS 지원은 별도의 파일 시스템으로 간주될 수 있습니다.

통합 파일 시스템은 라이브러리 지원 및 폴더 지원을 별도의 파일 시스템으로 취급합니다. 다른 기능을 가진 다른 파일 관리 지원 유형도 별도의 파일 시스템으로 처리됩니다.

공통 인터페이스를 통해 모든 파일 시스템과 대화식 작업을 할 수 있습니다. 이 인터페이스는 데이터 관리 인터페이스를 통하여 제공되는 레코드 입출력과는 달리 스트림 데이터의 입출력을 위하여 최적화되어 있습니다. 공통 인터페이스를 통한 파일 시스템과 상호작용은 제공된 명령, 메뉴 및 표시 화면과 API(Application Programming Interface)를 통해 이루어집니다.

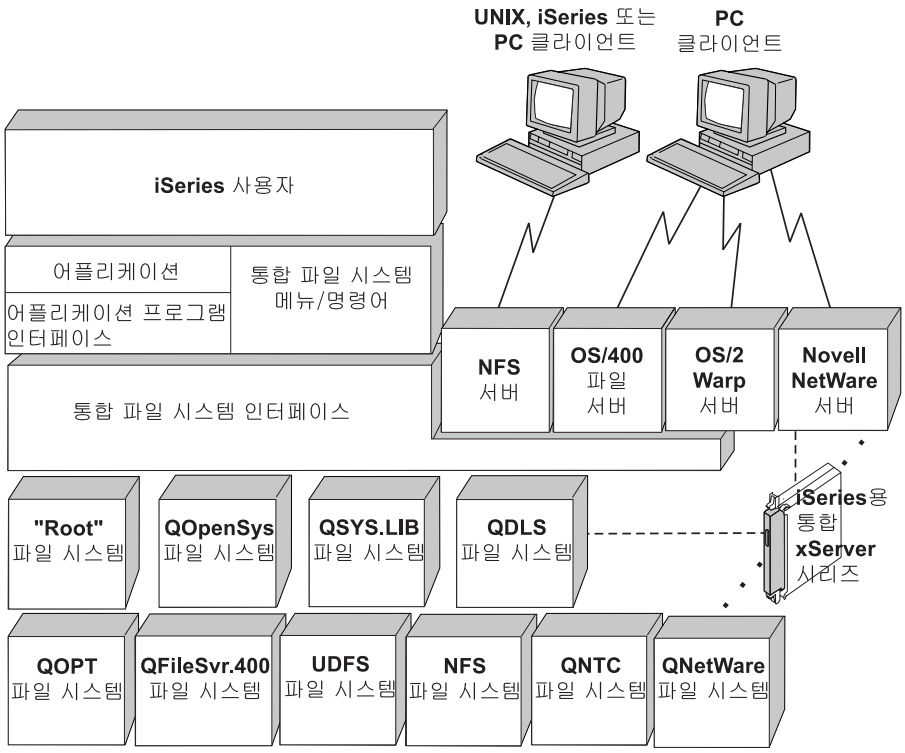

그림 9. 파일 시스템, 파일 서버 및 통합 파일 시스템 인터페이스

통합 파일 시스템 인터페이스를 통한 네트워크 파일 시스템 사용

통합 파일 시스템 인터페이스를 통해 네트워크 파일 시스템(NFS)에 액세스할 수 있습니다. 고려사항 및 제한사항에 유의하십시오.

관련 정보

광 기억장치

i5/OS 네트워크 파일 시스템 지원 PDF

통합 파일 시스템 보안 계획

통합 파일 시스템 보안 계획

파일 시스템 비교

다음 표는 각 파일 시스템의 기능 및 제한사항을 요약한 것입니다.

표 2. 파일 시스템 요약(1/2)

성능	"루트" (/)	QOpenSys	QSYS.LIB[16]	QDLS	QNTC
i5/OS의 표준 부분	예	예	예	예	예
파일 유형	스트림	스트림	레코드[12]	스트림	스트림
파일 크기 제한	T2=1 TB; T1=128 GB	T2=1 TB; T1=128 GB	데이터베이스 파일 크기	4GB	가변[17]
i5/OS 파일 서버를 통한 액세스	예	예	예	예	예

표 2. 파일 시스템 요약(1/2) (계속)

성능	"루트" (/)	QOpenSys	QSYS.LIB[16]	QDLS	QNTC
파일 서버 I/O 프로세서를 통한 직접 액세스[1]	아니오	아니오	아니오	아니오	예
열기/닫기의 비교 속도	보통[2]	보통[2]	저속[2]	저속[2]	보통[2]
대소문자로 구분되는 탐색	아니오	예	아니오[4]	아니오[5]	아니오
경로명에서 각 구성요소의 최대 길이	255자[3]	255자[3]	10/6자[6]	8/3자[7]	255자[3]
경로명의 최대 길이[8]	16MB	16MB	55 – 66자[4]	82자	255자
오브젝트에 대한 확장 속성의 최대 길이	2GB	2GB	가변[9]	32KB	0[18]
파일 시스템 내 디렉토리 계층의 최대 레벨	제한 없음[10]	제한 없음[10]	3	32	127
오브젝트당 최대 링크 수[11]	가변[15]	가변[15]	1	1	1
기호 링크 지원	예	예	아니오	아니오	아니오
오브젝트 또는 파일의 소유자 여부	예	예	예	예	아니오
통합 파일 시스템 명령 지원 여부	예	예	예	예	예
통합 파일 시스템 API 지원 여부	예	예	예	예	예
계층 파일 시스템(HFS) API 지원	아니오	아니오	아니오	예	아니오
스레드세이프[13]	예	예	예	아니오	예
오브젝트 저널링 지원	예	예	예[14]	아니오	아니오

표 2. 파일 시스템 요약(1/2) (계속)

성능	"루트" (/)	QOpenSys	QSYS.LIB[16]	QDLS	QNTC

주:

1. 파일 서버 I/O 프로세서는 LAN 서버에 의해 사용되는 하드웨어입니다.
2. 이 속도는 i5/OS 파일 서버를 통해 파일 시스템에 접근하는 경우 적용됩니다.
3. 특정 CCSID 값의 경우 최대 길이가 255자 미만일 수 있습니다.
4. QSYS.LIB 파일 시스템의 최대 경로명 길이는 55자입니다. 독립 ASP QSYS.LIB 파일 시스템의 최대 경로명 길이는 66자입니다.
5. 자세한 내용은 53 페이지의 『문서 라이브러리 서비스 파일 시스템(QDLS)』을 참조하십시오.
6. 이 값은 오브젝트명에는 최대 10자, 오브젝트 유형에는 최대 6자가 될 수 있습니다.
7. 이 값은 파일명에는 최고 8자, 파일 유형 확장자에는 최대 1-3자가 될 수 있습니다.
8. /로 시작하여 다음에 파일 시스템명이 오는 절대 경로명(예: `/QDLS...`)을 가정합니다.
9. QSYS.LIB 및 독립 ASP QSYS.LIB 파일 시스템은 세 가지 사전정의된 확장 속성 .SUBJECT, .CODEPAGE 및 .TYPE을 지원합니다. 최대 길이는 이들 3개의 확장 속성의 결합된 길이에 의해 결정됩니다.
10. 실제로, 디렉토리 레벨은 프로그램 및 시스템 공간 한계에 의해 제한됩니다.
11. 다른 디렉토리에 하나의 링크만을 가지는 디렉토리는 제외됩니다.
12. QSYS.LIB 및 독립 ASP QSYS.LIB 파일 시스템의 사용자 공간은 스트림 파일 입출력을 지원합니다.
13. 통합 파일 시스템 API는 스레드세이프 파일 시스템에 있는 오브젝트로 조작이 지시되는 경우 스레드세이프입니다. 다중 스레드가 작업에서 실행되고 있을 때 API가 스레드세이프가 아닌 파일 시스템의 오브젝트에 대하여 조작하는 경우, API는 실패합니다.
14. QSYS.LIB 및 독립 ASP QSYS.LIB 파일 시스템은 "루트"(/), UDFS 및 QOpenSys 파일 시스템 이외의 다른 오브젝트 유형 저널링을 지원합니다.
15. *TYPE2 디렉토리는 오브젝트당 백만 개의 링크 및 999,998개의 서브디렉토리만을 허용합니다. *TYPE1 디렉토리는 오브젝트당 32,767개의 링크만 허용합니다.
16. 이 열의 데이터는 QSYS.LIB 파일 시스템 및 독립 ASP QSYS.LIB 파일 시스템 모두를 나타냅니다.
17. 이 한계는 액세스되는 시스템에 따라 달라집니다.
18. QNTC는 확장 속성을 지원하지 않습니다.

약어

- T1 = *TYPE1 *STMF
- T2 = *TYPE2 *STMF
- B = 바이트 KB = 킬로바이트 MB = 메가바이트 GB = 기가 바이트 TB = 테라 바이트

표 3. 파일 시스템 요약(2/2)

성능	QOPT	QFileSvr.400	UDFS	NFS
i5/OS의 표준 부분	예	예	예	예
파일 유형	스트림	스트림	스트림	스트림
파일 크기 제한	4GB	연결변환[3]	T2 = 1 TB; T1=128 GB	연결변환[14]
i5/OS 파일 서버를 통한 액세스	예	예	예	예

표 3. 파일 시스템 요약(2/2) (계속)

성능	QOPT	QFileSvr.400	UDFS	NFS
파일 서버 I/O 프로세서를 통한 직접 액세스[1]	아니오	아니오	아니오	아니오
열기/닫기의 비교 속도	저속	저속[2]	보통[2]	보통[2]
대소문자로 구분되는 탐색	아니오	아니오[2]	예[11]	가변[2]
경로명에서 각 구성요소의 최대 길이	가변[4]	가변[2]	255자[15]	가변[2]
경로명의 최대 길이[5]	294자	제한 없음[2]	16MB	제한 없음[2]
오브젝트에 대한 확장 속성의 최대 길이	8MB	0[6]	2GB[10]	0[6]
파일 시스템 내 디렉토리 계층의 최대 레벨	제한 없음[7]	제한 없음[2]	제한 없음[7]	제한 없음[2]
오브젝트당 최대 링크 수[8]	1	1	연결변환[13]	가변[2]
기호 링크 지원	아니오	아니오	예	예[2]
오브젝트 또는 파일의 소유자 여부	아니오	아니오[9]	예	예[2]
통합 파일 시스템 명령 지원 여부	예	예	예	예
통합 파일 시스템 API 지원 여부	예	예	예	예
계층 파일 시스템(HFS) API 지원	예	아니오	아니오	아니오[2]
스레드세이프[12]	예	예	예	예
오브젝트 저널링 지원	아니오	아니오	예	아니오

표 3. 파일 시스템 요약(2/2) (계속)

성능	QOPT	QFileSvr.400	UDFS	NFS

주:

1. 파일 서버 I/O 프로세서는 LAN 서버에 의해 사용되는 하드웨어입니다.
2. 이 값은 액세스되는 리모트 파일 시스템에 따라 다릅니다.
3. V6R1 이전 시스템에 연결한 경우 이 파일 크기 한계는 2GB-1입니다. 이외의 경우 파일 크기 한계는 액세스되는 파일 시스템에 따라 다릅니다.
4. 자세한 내용은 56 페이지의 『광 파일 시스템(QOPT)』을 참조하십시오.
5. / 다음에 파일 시스템명으로 시작하는 절대 경로명을 가정합니다.
6. QFileSvr.400 파일 시스템은 액세스 중인 파일 시스템이 확장된 속성을 지원하는 경우에도 확장 속성을 리턴하지 않습니다.
7. 실제로, 디렉토리 레벨은 프로그램 및 시스템 공간 한계에 의해 제한됩니다.
8. 다른 디렉토리에 하나의 링크만을 가지는 디렉토리는 제외됩니다.
9. 액세스되는 파일 시스템은 오브젝트 소유자를 지원할 수 있습니다.
10. UDFS 자체에 대한 확장 속성의 최대 길이는 40바이트를 초과할 수 없습니다.
11. UDFS를 작성할 때, 대소문자 구분이 지정될 수 있습니다. UDFS를 작성할 때 *MIXED 매개변수가 사용되면, 대소문자 탐색이 허용됩니다.
12. 통합 파일 시스템 API가 다중 스레드 프로세스에서 액세스될 때 이들은 스레드세이프 방식으로 처리됩니다. 파일 시스템은 스레드세이프가 아닌 파일 시스템에 대해 액세스를 허용하지 않습니다.
13. *TYPE2 디렉토리는 오브젝트당 백만 개의 링크만 허용합니다. *TYPE1 디렉토리는 오브젝트당 32,767개의 링크만 허용합니다.
14. 이 한계는 액세스되는 시스템에 따라 달라집니다.
15. 특정 CCSID 값의 경우 최대 길이가 255자 미만일 수 있습니다.

약어

- T1 = *TYPE1 *STMF
- T2 = *TYPE2 *STMF
- B = 바이트 KB = 킬로바이트 MB = 메가바이트 GB = 기가 바이트 TB = 테라 바이트

관련 참조

32 페이지의 『"루트"(/) 파일 시스템』

『루트』(/) 파일 시스템은 통합 파일 시스템의 스트림 파일 지원 및 계층 디렉토리 구조의 장점을 이용합니다. 이 파일 시스템은 DOS 및 OS/2 파일 시스템의 특성을 가집니다.

35 페이지의 『개방 시스템 파일 시스템(QOpenSys)』

QOpenSys 파일 시스템은 POSIX 및 XPG(X/Open Portability Guide)와 같은 UNIX 기반의 개방 시스템 표준과 호환될 수 있습니다. "루트"(/) 파일 시스템과 마찬가지로 이 파일 시스템은 통합 파일 시스템에서 제공하는 스트림 파일 및 디렉토리 지원을 이용합니다.

37 페이지의 『사용자 정의 파일 시스템(UDFS)』

사용자 정의 파일 시스템(UDFS)은 사용자가 선택한 보조 기억장치 풀(ASP) 또는 독립 보조 기억장치 풀(ASP)에 상주합니다. 사용자가 이들 파일 시스템을 작성하고 관리할 수 있습니다.

45 페이지의 『라이브러리 파일 시스템(QSYS.LIB)』
QSYS.LIB 파일 시스템은 i5/OS 라이브러리 구조를 지원합니다.

49 페이지의 『독립 ASP QSYS.LIB』
독립 ASP QSYS.LIB 파일 시스템은 보조 기억장치 풀(ASP)의 i5/OS 라이브러리 구조를 지원합니다. 이 파일 시스템은 독립 ASP에서 데이터베이스 파일과 라이브러리 지원이 관리하는 다른 모든 i5/OS 오브젝트 유형에 대한 액세스를 제공합니다.

53 페이지의 『문서 라이브러리 서비스 파일 시스템(QDLS)』
QDLS 파일 시스템은 폴더 구조를 지원합니다. 문서 및 폴더로의 액세스를 제공합니다.

56 페이지의 『광 파일 시스템(QOPT)』
QOPT 파일 시스템은 광 매체에 저장된 스트림 데이터에 대한 액세스를 제공합니다.

58 페이지의 『i5/OS NetClient 파일 시스템(QNTC)』
QNTC 파일 시스템은 Windows NT 4.0 이상 또는 Linux® 오퍼레이팅 시스템을 실행 중인 통합 xSeries 서버(IXS)에 저장된 데이터 및 오브젝트에 대한 액세스를 제공합니다. QNTC 파일 시스템은 Windows NT 4.0 이상, Linux Samba 3.0 이상 또는 지원되는 버전의 i5/OS NetServer™를 실행 중인 리모트 서버에 저장된 데이터 및 오브젝트에 대한 액세스도 제공합니다.

64 페이지의 『i5/OS 파일 서버 파일 시스템(QFileSvr.400)』
QFileSvr.400 파일 시스템은 리모트 System i 플랫폼에 상주하는 라은 파일 시스템에 대한 투명한 액세스를 제공합니다. 이는 계층 디렉토리 구조를 통해 액세스됩니다.

69 페이지의 『네트워크 파일 시스템(NFS)』
네트워크 파일 시스템(NFS)은 리모트 NFS 서버에 저장된 오브젝트 및 데이터에 대한 액세스를 제공합니다.

관련 정보

저널 관리

"루트"(/) 파일 시스템

『루트』(/) 파일 시스템은 통합 파일 시스템의 스트림 파일 지원 및 계층 디렉토리 구조의 장점을 이용합니다. 이 파일 시스템은 DOS 및 OS/2 파일 시스템의 특성을 가집니다.

또한,

- 스트림 파일 입출력(I/O)에 대해 최적화되었습니다.
- 복수 하드 링크 및 기호 링크를 지원합니다.
- 로컬 소켓을 지원합니다.
- 스레드세이프 API를 지원합니다.
- *FIFO 오브젝트를 지원합니다.
- 기타 *CHRSF 오브젝트 및 /dev/null, /dev/zero *CHRSF 오브젝트를 지원합니다.
- 오브젝트 변경사항의 저널링을 지원합니다.
- 통합 파일 시스템 스캔 관련 종료점을 사용하여 오브젝트의 스캐닝을 지원합니다.

『루트』(/) 파일 시스템은 /dev/null 및 /dev/zero라는 문자 특수 파일(*CHRSF)을 지원합니다. 문자 특수 파일은 장치나 컴퓨터 시스템 자원과 연관됩니다. 문자 특수 파일에는 디렉토리에 나타나는 경로명이 있으며, 일반 파일과 동일한 액세스 보호가 있습니다. /dev/null 또는 /dev/zero 문자 특수 파일은 항상 비어 있으며, /dev/null 또는 /dev/zero에 기록된 다른 데이터는 삭제됩니다. /dev/null 및 /dev/zero 파일의 오브젝트 유형은 *CHRSF이며 일반 파일처럼 사용될 수 있지만, /dev/null 파일에서 데이터를 읽을 수 없으며 /dev/zero 파일은 0으로 지워진 데이터를 항상 리턴합니다.

"루트"(/) 파일 시스템 액세스

i5/OS 파일 서버나 통합 파일 시스템 명령, 사용자 화면 및 API를 사용하여 통합 파일 시스템 인터페이스를 통해 『루트』(/) 파일 시스템에 액세스할 수 있습니다.

"루트"(/) 파일 시스템에서 대소문자 구분

파일 시스템은 입력된 오브젝트명과 동일한 대소문자 양식을 보유하지만, 시스템에서 이름을 탐색할 때는 대소문자를 구분하지 않습니다.

"루트"(/) 파일 시스템에서 경로명

『루트』(/) 파일 시스템에서 경로명은 고유 양식을 가집니다.

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

- 경로명의 각 구성요소는 최대 255자가 될 수 있습니다(QSYS.LIB 또는 QDLS 파일 시스템보다 더 긴). 완전한 경로명은 최대 16MB가 될 수 있습니다.
- 프로그램 및 시스템 공간 제한을 제외하고 디렉토리 계층 깊이에는 제한이 없습니다.
- 이름 저장 시 문자는 UCS2 레벨 1 양식(*TYPE1 디렉토리) 및 UTF-16(*TYPE2 디렉토리)으로 변환됩니다.

관련 개념

18 페이지의 『이름 연속성』

『루트』(/), QOpenSys 및 사용자 정의 파일 시스템을 사용할 때 오브젝명의 문자가 동일하게 남아 있도록 하는 시스템 지원을 이용할 수 있습니다.

10 페이지의 『*TYPE2 디렉토리』

통합 파일 시스템의 "루트"(/), QOpenSys 및 사용자 정의 파일 시스템(UDFS)은 *TYPE2 디렉토리 형식을 지원합니다. *TYPE2 디렉토리 형식은 기존의 *TYPE1 디렉토리 형식을 개선한 것입니다.

15 페이지의 『경로명』

경로명(일부 시스템에서는 *pathname*이라고도 함)은 서버에 오브젝트를 찾는 방법을 알려줍니다.

"루트"(/) 파일 시스템에서 링크

『루트』(/) 파일 시스템에서는 동일한 오브젝트에 여러 개의 하드 링크가 허용됩니다. 기호 링크가 전면 지원됩니다.

기호 링크는 『루트』(/) 파일 시스템에서 QSYS.LIB, 독립 ASP QSYS.LIB 또는 QDLS와 같은 다른 파일 시스템으로 링크하는 데 사용될 수 있습니다.

관련 개념

12 페이지의 『링크』

*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

"루트"(/) 파일 시스템에서 통합 파일 시스템 명령 사용

CL 명령을 사용한 액세스 주제에 나열된 모든 명령과 메뉴 및 표시 화면을 사용한 액세스 주제에 설명된 표시 화면은 『루트』(/) 파일 시스템에서 작동할 수 있습니다. 그러나 다중 스레드가 가능한 프로세스에서 이러한 명령을 사용하는 것은 안전하지 않을 수도 있습니다.

관련 태스크

72 페이지의 『메뉴 및 표시 화면을 사용한 액세스』

시스템에서 제공하는 메뉴 및 표시장치 세트를 사용하여 통합 파일 시스템의 파일 및 기타 오브젝트에 대한 조작을 수행할 수 있습니다.

관련 참조

74 페이지의 『CL 명령을 사용한 액세스』

통합 파일 시스템 메뉴 및 표시 화면을 통해 수행할 수 있는 모든 조작은 제어 언어(CL) 명령을 입력하여 수행될 수 있습니다. 이러한 명령은 통합 파일 시스템 인터페이스를 통해 액세스할 수 있는 파일 시스템의 파일 및 디타 오브젝트에 대해 조작할 수 있습니다.

"루트"(/) 파일 시스템에서 통합 파일 시스템 API 사용

API를 사용한 조작 수행 주제에 나열된 모든 API는 『루트』(/) 파일 시스템에서 작동할 수 있습니다.

관련 참조

122 페이지의 『API를 사용한 조작 수행』

통합 파일 시스템 오브젝트에서 조작을 수행하는 API(Application Programming Interface)의 대부분은 C 언어 함수 형태입니다.

관련 정보

어플리케이션 프로그램 인터페이스(API)

"루트"(/) 파일 시스템에서 오브젝트 변경사항 저널링

| 『루트』(/) 파일 시스템의 일부 오브젝트 유형은 저널링될 수 있습니다. 이 기능을 사용하여 오브젝트가 마지막
| 으로 저장된 이후에 발생한 오브젝트의 변경사항을 복구할 수 있습니다.

관련 개념

106 페이지의 『오브젝트 저널링』

저널링의 1차 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 복구할 수 있도록 하는 것입니다. 또한 저널링의 주요 사용은 고가용성 또는 작업부하 균형 조절을 위해 다른 시스템에 오브젝트 변경사항을 복제할 때 도움을 주기 위한 것입니다.

″루트″(/) 파일 시스템에서 UDP 및 TCP 장치

/dev/xti 디렉토리 아래의 『루트』 파일 시스템은 이제 udp 및 tcp라고 명명된 두 개의 장치 드라이버를 보유합니다.

두 드라이버는 문자 특수 파일(*CHRSF)이며 첫 번째 초기 프로그램 로드(IPL) 중에 작성됩니다. 사용자 데이터그램 프로토콜(UDP) 및 TCP 장치 드라이버는 UDP 및 TCP 전송 제공자에 대한 연결을 열기 위해 사용됩니다. 이 두 드라이버는 사용자 장치이며 새로운 장치 주요 번호를 수신합니다. 이들은 또한 복제된 열기 조작을 가지며 이것은 각각의 열기 조작이 장치의 고유 인스턴스를 얻는 것을 의미합니다. 이러한 장치 사용은 i5/OS PASE(Portable Application Solutions Environment)에서만 지원됩니다. 다음 표에는 작성될 오브젝트와 해당 등록 정보가 포함되어 있습니다.

표 4. 장치 드라이버 오브젝트 및 등록 정보

경로명	유형	주	부	소유자	소유자 데이터 권한	그룹	그룹 데이터 권한	공용 데이터 권한
/dev/xti	*DIR	N/A	N/A	QSYS	*RWX	없음	*RX	*RX
/dev/xti/tcp	*CHRSF	복제	TCP	QSYS	*RW	없음	*RW	*RW
/dev/xti/udp	*CHRSF	복제	UDP	QSYS	*RW	없음	*RW	*RW

관련 정보

i5/OS PASE

개방 시스템 파일 시스템(QOpenSys)

QOpenSys 파일 시스템은 POSIX 및 XPG(X/Open Portability Guide)와 같은 UNIX 기반의 개방 시스템 표준과 호환될 수 있습니다. ″루트″(/) 파일 시스템과 마찬가지로 이 파일 시스템은 통합 파일 시스템에서 제공하는 스트림 파일 및 디렉토리 지원을 이용합니다.

또한,

- UNIX 시스템과 유사한 계층적 디렉토리 구조를 통해 액세스됩니다.
- 스트림 파일 입출력(I/O)에 대해 최적화되었습니다.
- 복수 하드 링크 및 기호 링크를 지원합니다.
- 대소문자 구분 이름을 지원합니다.
- 로컬 소켓을 지원합니다.
- 스레드세이프 API를 지원합니다.
- *FIFO 오브젝트를 지원합니다.
- 오브젝트 변경사항의 저널링을 지원합니다.
- 통합 파일 시스템 스캔 관련 종료점을 사용하여 오브젝트의 스캐닝을 지원합니다.

QOpenSys 파일 시스템은 UNIX 기반의 개방 시스템 표준을 지원할 수 있게 하는 대소문자 구분 이외에 『루트』(/) 파일 시스템과 같은 특성을 지닙니다.

QOpenSys 액세스

i5/OS 파일 서버나 통합 파일 시스템 명령, 사용자 화면 및 API를 사용하여 통합 파일 시스템 인터페이스를 통해 QOpenSys에 액세스할 수 있습니다.

QOpenSys 파일 시스템에서 대소문자 구분

『루트』(/) 파일 시스템과 달리 QOpenSys 파일 시스템은 오브젝트명 탐색 시 대소문자를 구분합니다.

예를 들어, 모두 대문자로 된 문자 스트링은 모두 소문자로 된 문자 스트링과 같은 것으로 인식되지 않습니다.

대소문자 구분 기능으로 인해, 대소문자에 차이가 있는 경우, 이중 이름을 사용할 수도 있습니다. 예를 들어, QOpenSys의 동일한 디렉토리에 `Payroll`, `PayRoll` 및 `PAYROLL` 오브젝트를 모두 사용할 수 있습니다.

QOpenSys 파일 시스템에서 경로명

QOpenSys 파일 시스템에서 경로명은 고유 양식을 가집니다.

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

- 경로명의 각각 구성요소는 최대 255자가 될 수 있습니다. 전체 경로명은 최대 16MB 길이가 될 수 있습니다.
- 프로그램 및 시스템 공간 제한을 제외하고 디렉토리 계층 깊이에는 제한이 없습니다.
- 이름 저장 시 문자는 UCS2 레벨 1 양식(*TYPE1 디렉토리) 및 UTF-16(*TYPE2 디렉토리)으로 변환됩니다.

관련 개념

18 페이지의 『이름 연속성』
『루트』(/), QOpenSys 및 사용자 정의 파일 시스템을 사용할 때 오브젝명의 문자가 동일하게 남아 있도록 하는 시스템 지원을 이용할 수 있습니다.

10 페이지의 『*TYPE2 디렉토리』
통합 파일 시스템의 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템(UDFS)은 *TYPE2 디렉토리 형식을 지원합니다. *TYPE2 디렉토리 형식은 기존의 *TYPE1 디렉토리 형식을 개선한 것입니다.

15 페이지의 『경로명』
경로명(일부 시스템에서는 *pathname*이라고도 함)은 서버에 오브젝트를 찾는 방법을 알려줍니다.

QOpenSys 파일 시스템에서 링크

QOpenSys 파일 시스템에서 같은 오브젝트에 복수 하드 링크가 허용됩니다. 기호 링크가 전면 지원됩니다.

기호 링크는 QOpenSys 파일 시스템에서 다른 파일 시스템의 오브젝트로 링크하는 데 사용될 수 있습니다.

관련 개념

12 페이지의 『링크』
*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

QOpenSys 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

CL 명령을 사용한 액세스 주제에 나열된 모든 명령과 메뉴 및 표시 화면을 사용한 액세스 주제에 설명된 모든 표시 화면은 QOpenSys 파일 시스템에서 작동할 수 있습니다. 그러나 다중 스레드가 가능한 프로세스에서 이 명령을 사용하는 것은 안전하지 않을 수도 있습니다.

관련 태스크

72 페이지의『메뉴 및 표시 화면을 사용한 액세스』
시스템에서 제공하는 메뉴 및 표시장치 세트를 사용하여 통합 파일 시스템의 파일 및 기타 오브젝트에 대한 조작을 수행할 수 있습니다.

관련 참조

74 페이지의『CL 명령을 사용한 액세스』
통합 파일 시스템 메뉴 및 표시 화면을 통해 수행할 수 있는 모든 조작은 제어 언어(CL) 명령을 입력하여 수행될 수 있습니다. 이러한 명령은 통합 파일 시스템 인터페이스를 통해 액세스할 수 있는 파일 시스템의 파일 및 디타 오브젝트에 대해 조작할 수 있습니다.

QOpenSys 파일 시스템에서 통합 파일 시스템 API 사용

API를 사용한 조작 수행 주제에 나열된 모든 API는 QOpenSys 파일 시스템에서 작동할 수 있습니다.

관련 참조

122 페이지의『API를 사용한 조작 수행』
통합 파일 시스템 오브젝트에서 조작을 수행하는 API(Application Programming Interface)의 대부분은 C 언어 함수 형태입니다.

관련 정보

어플리케이션 프로그램 인터페이스(API)

QOpenSys 파일 시스템에서 오브젝트 변경사항 저널링

| QOpenSys 파일 시스템의 일부 오브젝트 유형은 저널링될 수 있습니다. 이 기능을 사용하여 오브젝트가 마지
| 막으로 저장된 이후에 발생한 오브젝트의 변경사항을 복구할 수 있습니다.

관련 개념

106 페이지의『오브젝트 저널링』
저널링의 1차 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 복구할 수 있도록 하는 것입니다. 또한 저널링의 주요 사용은 고가용성 또는 작업부하 균형 조절을 위해 다른 시스템에 오브젝트 변경사항을 복제할 때 도움을 주기 위한 것입니다.

사용자 정의 파일 시스템(UDFS)

사용자 정의 파일 시스템(UDFS)은 사용자가 선택한 보조 기억장치 풀(ASP) 또는 독립 보조 기억장치 풀(ASP)에 상주합니다. 사용자가 이들 파일 시스템을 작성하고 관리할 수 있습니다.

또한,

- DOS 및 OS/2와 같은 PC 오퍼레이팅 시스템과 유사한 계층 디렉토리 구조를 제공합니다.

- 스트림 파일 입출력에 최적화되었습니다.
- 복수 하드 링크 및 기호 링크를 지원합니다.
- 로컬 소켓을 지원합니다.
- 스레드세이프 API를 지원합니다.
- *FIFO 오브젝트를 지원합니다.
- 오브젝트 변경사항의 저널링을 지원합니다.
- 통합 파일 시스템 스캔 관련 종료점을 사용하여 오브젝트의 스캐닝을 지원합니다.

각 UDFS에 대해 고유한 이름을 부여하여 복수 UDFS를 작성할 수 있습니다. UDFS를 작성하는 동안 다음을 포함하여 UDFS에 대한 다른 속성을 지정할 수 있습니다.

- UDFS에 위치한 오브젝트가 저장된 독립 ASP 이름 또는 ASP 번호.
- UDFS에 있는 오브젝트명의 대소문자 구분 특성

  UDFS에 있는 오브젝트명을 탐색할 때 대소문자의 일치 여부를 결정하는 UDFS의 대소문자 특성
- UDFS에서 작성되는 오브젝트에 대해 설정되어야 하는 스캔 속성을 정의하는 오브젝트 스캐닝 속성을 작성합니다.
- 이름 변경 및 링크되지 않은 속성 제한을 위한 값
- UDFS에서 작성된 오브젝트에 대한 감사 값
- UDFS에서 작성된 스트림 파일용 다른 형식(*TYPE1 및 *TYPE2)
- USFS에서 작성된 스트림 파일용 디스크 기억장치 옵션
- UDFS에서 작성된 스트림 파일용 주 기억장치 옵션

사용자 정의 파일 시스템 개념

사용자 정의 파일 시스템(UDFS)에서, 『루트』(/) 및 QOpenSys 파일 시스템에서와 같이 디렉토리, 스트림 파일, 기호 링크, 로컬 소켓 및 *FIFO 오브젝트를 작성할 수 있습니다.

단일 블록 특수 파일 오브젝트(*BLKSF)는 UDFS를 나타냅니다. UDFS를 작성할 때 블록 특수 파일도 자동으로 작성됩니다. 블록 특수 파일은 통합 파일 시스템 총칭 명령, API 및 QFileSvr.400 인터페이스를 통해서만 사용자에 액세스할 수 있습니다.

주: 블록 특수 파일의 속성 및 권한을 변경하면 UDFS의 루트 디렉토리에도 동일한 변경사항이 적용되며 그 역도 가능합니다. 변경사항은 UDFS 내 다른 오브젝트에 영향을 미치지 않습니다.

UDFS에는 **마운트**와 **마운트 해제**의 두 가지 상태만이 존재합니다. UDFS를 마운트할 때, 그 안의 오브젝트에 액세스할 수 있습니다. UDFS를 마운트 해제하면 그 안에 있는 오브젝트에 액세스할 수 없습니다.

UDFS 내의 오브젝트에 액세스하려면, 반드시 디렉토리(예: /home/JON)에 UDFS를 마운트해야 합니다. 디렉토리에 UDFS를 마운트할 때 오브젝트와 서브디렉토리를 포함하여 해당 디렉토리의 원래 내용에 액세스할 수 없게 됩니다. UDFS를 마운트할 때, UDFS 내용은 UDFS를 마운트했던 디렉토리 경로를 통해 액세스될 수

| 있습니다. 예를 들어, /home/JON 디렉토리에 파일 /home/JON/payroll이 있다고 가정하십시오. UDFS에는
| mail, action, 및 outgoing의 세 개의 디렉토리가 있습니다. UDFS를 /home/JON에 마운트한 후에는,
| /home/JON/payroll 파일로 액세스할 수 없으며, 세 개의 UDFS 디렉토리는 /home/JON/mail,
| /home/JON/action 그리고 /home/JON/outgoing으로 액세스할 수 있게 됩니다. UDFS를 마운트 해제시키
| 고 난 후, /home/JON/payroll 파일의 액세스가 다시 가능해지며 UDFS에 있는 3개 디렉토리의 액세스가
| 불가능해집니다.

시스템의 초기 프로그램 로드(IPL) 또는 디렉토리의 기억장치 재생 연산(RCLSTG)은 모든 UDFS의 마운트를 해제합니다. 따라서 IPL 수행 또는 해당 디렉토리에서 RCLSTG 명령을 실행한 후 UDFS를 다시 마운트해야 합니다.

주: 독립 ASP에 있는 UDFS는 마운트될 수 없습니다.

통합 파일 시스템 인터페이스를 통한 사용자 정의 파일 시스템 액세스

i5/OS 파일 서버나 통합 파일 시스템 명령, 사용자 화면 및 API를 사용하여 통합 파일 시스템 인터페이스를 통해 사용자 정의 파일 시스템(UDFS)에 액세스할 수 있습니다.

통합 파일 시스템 인터페이스를 사용할 경우 다음 고려사항 및 제한사항을 알고 있어야 합니다.

관련 개념

12 페이지의 『링크』

*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

16 페이지의 『스트림 파일』

*스트림 파일*은 단순하게 바이트가 나열된 구조로 랜덤 액세스가 가능합니다.

관련 정보

CRTUDFS(사용자 정의 파일 시스템 작성) 명령

통합 파일 시스템 사용자 정의 파일 시스템에서 대소문자 구분

사용자 정의 파일 시스템(UDFS)을 작성할 때 UDFS의 오브젝트명이 대소문자를 구분하는지 여부를 지정할 수 있습니다.

대소문자 구분을 선택했을 때는 오브젝트 이름을 탐색할 때 대소문자의 구분이 이루어집니다. 예를 들어, 모두 대문자로 제공된 이름은 소문자로된 같은 이름과 일치하지 않게 됩니다. 따라서, /home/MURPH/와 /home/murph/은 다른 디렉토리로 인식됩니다. 대소문자 구분 UDFS를 작성하기 위해 CRTUDFS(사용자 정의 파일 시스템 작성) 명령을 사용할 때 CASE 매개변수에 *MIXED를 지정할 수 있습니다.

대소문자 구분 안함을 선택하면 시스템에서 이름을 탐색하는 동안 대소문자를 구분하지 않습니다. 따라서 시스템은 /home/CAYCE 및 /HOME/cayce를 분리된 디렉토리가 아닌 동일한 디렉토리로 인식합니다. 대소문자를 구분하지 않는 UDFS를 작성하기 위해 CRTUDFS 명령을 사용할 때 CASE 매개변수에 *MONO를 지정할 수 있습니다.

어떤 경우든, 파일 시스템은 사용자가 입력한 오브젝트 이름의 대소문자 형식을 저장합니다. 대소문자 구분 옵션은 사용자가 시스템을 통해 이름을 탐색하는 방법에만 적용됩니다.

관련 정보

CRTUDFS(사용자 정의 파일 시스템 작성) 명령

통합 파일 시스템 사용자 정의 파일 시스템에서 경로명

블록 특수 파일(*BLKSF)은 전체 UDFS 및 UDFS 내의 모든 오브젝트를 조작해야 할 때의 사용자 정의 파일 시스템(UDFS)을 나타냅니다.

UDFS가 시스템 또는 기본 사용자 ASP에 상주하는 경우, 블록 특수 파일명은 반드시 다음 형식이어야 합니다.

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

여기서, XX는 UDFS를 저장하는 ASP 번호이며 udfs_name은 해당 ASP 내에 있는 UDFS의 고유명입니다. UDFS 이름은 반드시 .udfs 확장자로 끝나야 한다는 것을 기억하십시오.

UDFS가 독립 ASP에 상주하는 경우, 블록 특수 파일명은 반드시 다음 형식이어야 합니다.

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

여기서 asp_name은 UDFS를 저장하는 독립 ASP의 이름이며, udfs_name은 해당 독립 ASP에 있는 UDFS의 고유명입니다. UDFS 이름은 반드시 .udfs 확장자로 끝나야 한다는 것을 기억하십시오.

UDFS에 있는 오브젝트의 경로명은 UDFS를 마운트한 디렉토리와 관련됩니다. 예를 들어, UDFS /dev/qasp01/wysocki.udfs를 /home/dennis에 마운트하고 나면, UDFS에 있는 모든 오브젝트의 경로명이 /home/dennis로 시작됩니다.

추가 경로명 규칙은 다음과 같습니다.

- 경로명의 각각 구성요소는 최대 255자가 될 수 있습니다. 전체 경로명은 최대 16MB 길이가 될 수 있습니다.
- 프로그램 및 서버 공간 제한을 제외하고 디렉토리 계층 깊이에는 제한이 없습니다.
- 이름 저장 시 문자는 UCS2 레벨 1 양식(*TYPE1 디렉토리) 및 UTF-16(*TYPE2 디렉토리)으로 변환됩니다.

관련 개념

18 페이지의 『이름 연속성』

『루트』(/), QOpenSys 및 사용자 정의 파일 시스템을 사용할 때 오브젝명의 문자가 동일하게 남아 있도록 하는 시스템 지원을 이용할 수 있습니다.

10 페이지의 『*TYPE2 디렉토리』

통합 파일 시스템의 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템(UDFS)은 *TYPE2 디렉토리 형식을 지원합니다. *TYPE2 디렉토리 형식은 기존의 *TYPE1 디렉토리 형식을 개선한 것입니다.

15 페이지의 『경로명』

경로명(일부 시스템에서는 *pathname*이라고도 함)은 서버에 오브젝트를 찾는 방법을 알려줍니다.

통합 파일 시스템 사용자 정의 파일 시스템에서 링크

사용자 정의 파일 시스템(UDFS)을 사용하면 복수 하드 링크가 같은 오브젝트에 액세스할 수 있으며 기호 링크가 완전히 지원됩니다.

기호 링크는 UDFS로부터 다른 파일 시스템의 오브젝트로 링크를 작성할 수 있습니다.

관련 개념

12 페이지의 『링크』

*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

사용자 정의 파일 시스템에서 통합 파일 시스템 명령 사용

CL 명령을 사용한 액세스 주제에 나열된 모든 명령과 메뉴 및 표시 화면을 사용한 액세스 주제에 설명된 모든 표시 화면은 사용자 정의 파일 시스템에서 작동할 수 있습니다.

일반적으로 UDFS 및 기타 마운트된 파일 시스템에 고유한 몇 가지 CL 명령이 있습니다. 다음은 이에 관한 설명입니다.

표 5. 사용자 정의 파일 시스템 CL 명령

명령	설명
ADDMFS	마운트된 파일 시스템 추가. 로컬 클라이언트 디렉토리에 내보낸, 리모트 서버 파일 시스템을 위치시킵니다.
CRTUDFS	UDFS 작성. 사용자 정의 파일 시스템을 작성합니다.
DLTUDFS	UDFS 삭제. 사용자 정의 파일 시스템을 삭제합니다.
DSPMFSINF	마운트된 파일 시스템 정보 표시. 마운트된 파일 시스템에 관한 정보를 표시합니다.
DSPUDFS	UDFS 표시. 사용자 정의 파일 시스템에 관한 정보를 표시합니다.
MOUNT	파일 시스템 마운트. 로컬 클라이언트 디렉토리에 내보낸, 리모트 서버 파일 시스템을 위치시킵니다. 이 명령은 ADDMFS 명령에 대한 별명입니다.
RMVMFS	마운트된 파일 시스템 제거. 내보낸, 리모트 서버 파일 시스템을 로컬 클라이언트 이름공간에서 제거합니다.
UNMOUNT	파일 시스템 마운트 해제. 내보낸, 리모트 서버 파일 시스템을 로컬 클라이언트 이름공간에서 제거합니다. 이 명령은 RMVMFS 명령에 대한 별명입니다.

주: UDFS에 저장된 오브젝트에 대해 통합 파일 시스템 명령을 작동하기 전에 반드시 해당 UDFS를 마운트해야 합니다.

관련 태스크

72 페이지의 『메뉴 및 표시 화면을 사용한 액세스』

시스템에서 제공하는 메뉴 및 표시장치 세트를 사용하여 통합 파일 시스템의 파일 및 기타 오브젝트에 대한 조작을 수행할 수 있습니다.

관련 참조

74 페이지의『CL 명령을 사용한 액세스』
통합 파일 시스템 메뉴 및 표시 화면을 통해 수행할 수 있는 모든 조작은 제어 언어(CL) 명령을 입력하여 수행될 수 있습니다. 이러한 명령은 통합 파일 시스템 인터페이스를 통해 액세스할 수 있는 파일 시스템의 파일 및 디타 오브젝트에 대해 조작할 수 있습니다.

사용자 정의 파일 시스템에서 통합 파일 시스템 API 사용

API를 사용한 조작 수행 주제에 나열된 모든 API는 사용자 정의 파일 시스템에서 작동할 수 있습니다.

주: UDFS에 저장된 오브젝트에서 통합 파일 시스템 API를 조작하기 전에 반드시 해당 UDFS를 마운트해야 합니다.

관련 참조

122 페이지의『API를 사용한 조작 수행』
통합 파일 시스템 오브젝트에서 조작을 수행하는 API(Application Programming Interface)의 대부분은 C 언어 함수 형태입니다.

관련 정보

어플리케이션 프로그램 인터페이스(API)

사용자 정의 파일 시스템에 대한 그래픽 사용자 인터페이스

System i Navigator, 사용자 PC의 그래픽 사용자 인터페이스는 사용자 정의 파일 시스템(UDFS)에 대한 쉽고 편리한 액세스를 제공합니다.

이 인터페이스를 사용하여 Windows 클라이언트로부터 UDFS를 작성, 삭제, 표시, 마운트 및 마운트 해제할 수 있습니다.

System i Navigator를 통해 UDFS에 대한 조작을 수행할 수 있습니다. 기본 타스크는 다음과 같습니다.

- 145 페이지의『새로운 사용자 정의 파일 시스템 작성』
- 145 페이지의『사용자 정의 파일 시스템 마운트』
- 146 페이지의『사용자 정의 파일 시스템 마운트 해제』

통합 파일 시스템 사용자 정의 파일 시스템 작성

CRTUDFS(사용자 정의 파일 시스템 작성) 명령은 통합 파일 시스템 이름공간, API 및 CL 명령을 통해 볼 수 있는 파일 시스템을 작성합니다.

ADDMFS 또는 MOUNT 명령은 기존 로컬 디렉토리의 맨 위에 사용자 정의 파일 시스템(UDFS)을 위치시킵니다. 사용자가 선택한 ASP나 독립 ASP에서 UDFS를 작성할 수 있습니다.

또한 UDFS에 다음 항목을 지정할 수 있습니다.

- UDFS에 위치한 오브젝트가 저장된 독립 ASP 이름 또는 ASP 번호.
- UDFS에 있는 오브젝트명의 대소문자 구분 특성

UDFS에 있는 오브젝트명을 탐색할 때 대소문자의 일치 여부를 결정하는 UDFS의 대소문자 특성

- UDFS에서 작성되는 오브젝트에 대해 설정되어야 하는 스캔 속성을 정의하는 오브젝트 스캐닝 속성을 작성합니다.
- 이름 변경 및 링크되지 않은 속성 제한을 위한 값
- UDFS에서 작성된 오브젝트에 대한 감사 값
- UDFS에서 작성된 스트림 파일용 다른 형식(*TYPE1 및 *TYPE2)
- USFS에서 작성된 스트림 파일용 디스크 기억장치 옵션
- UDFS에서 작성된 스트림 파일용 주 기억장치 옵션

관련 정보

CRTUDFS(사용자 정의 파일 시스템 작성) 명령

ADDMFS(마운트된 파일 시스템 추가) 명령

통합 파일 시스템 사용자 정의 파일 시스템 삭제

DLTUDFS(사용자 정의 파일 시스템 삭제) 명령은 기존의 마운트 해제된 사용자 정의 파일 시스템(UDFS) 및 그 내부의 모든 오브젝트를 삭제합니다.

UDFS를 마운트한 경우, 명령은 실패하게 됩니다. UDFS를 삭제하면 UDFS 내에 있는 모든 오브젝트가 삭제됩니다. UDFS 내의 모든 오브젝트를 삭제하기 위한 적절한 권한이 없으면, 어떤 오브젝트도 삭제되지 않습니다.

관련 정보

DLTUDFS(사용자 정의 파일 시스템 삭제) 명령

통합 파일 시스템 사용자 정의 파일 시스템 표시

DSPUDFS(사용자 정의 파일 시스템 표시) 명령은 기존 사용자 정의 파일 시스템(UDFS)의 속성(마운트 또는 마운트 해제 여부)을 나타냅니다.

DSPMFSINF(마운트된 파일 시스템 정보 표시) 명령은 마운트된 파일 시스템뿐 아니라 마운트된 UDFS에 대한 정보도 표시합니다.

관련 정보

DSPUDFS(사용자 정의 파일 시스템 표시) 명령

DSPMFSINF(마운트 파일 시스템 정보 표시) 명령

통합 파일 시스템 사용자 정의 파일 시스템 마운트

ADDMFS(마운트된 파일 시스템 추가) 및 MOUNT 명령은 파일 시스템의 오브젝트가 통합 파일 시스템 이름공간에 액세스할 수 있도록 합니다.

사용자 정의 파일 시스템(UDFS)을 마운트하려면 ADDMFS 명령의 TYPE 매개변수에 *UDFS를 지정해야 합니다.

시스템의 초기 프로그램 로드(IPL) 또는 디렉토리의 기억장치 재생 연산(RCLSTG)은 모든 UDFS의 마운트를 해제합니다. 따라서 IPL 수행 또는 해당 디렉토리에서 RCLSTG 명령을 실행한 후 UDFS를 다시 마운트해야 합니다.

주: 독립 ASP에 있는 UDFS는 마운트될 수 없습니다.

관련 정보

ADDMFS(마운트된 파일 시스템 추가) 명령

통합 파일 시스템 사용자 정의 파일 시스템 마운트 해제

마운트 해제 명령은 사용자 정의 파일 시스템(UDFS)의 내용을 통합 파일 시스템 인터페이스에 액세스할 수 없도록 합니다.

주: 시스템의 초기 프로그램 로드(IPL) 또는 디렉토리의 기억장치 재생 연산(RCLSTG)은 모든 UDFS의 마운트를 해제합니다.

UDFS안에 있는 오브젝트는 UDFS가 일단 마운트 해제되면, 개별적으로 액세스가 불가능합니다. RMVMFS(마운트된 파일 시스템 제거) 또는 UNMOUNT 명령은 마운트된 파일 시스템이 통합 파일 시스템 이름공간에 액세스하지 못하게 합니다. 명령 사용 시 파일 시스템의 임의의 오브젝트가 사용 중이면(예를 들어, 파일이 열려 있는 경우) 오류 메세지가 수신됩니다. UDFS는 마운트된 채로 남아 있습니다. UDFS의 한 부분에 마운트한 경우 그 부분을 떼어낼 때까지 이 UDFS를 마운트 해제할 수 없습니다.

예를 들어, UDFS `/dev/qasp02/jenn.udfs`를 통합 파일 시스템 이름공간의 `/home/judy`에 마운트합니다. 그런 다음, 다른 파일 시스템 `/pubs`를 `/home/judy`에 마운트하면 `jenn.udfs`의 내용에 액세스할 수 없게 됩니다. 또한, `/home/judy`로부터 두 번째 파일 시스템을 제거할 때까지는 `jenn.udfs`를 마운트 해제할 수 없습니다.

주: 독립 ASP에 있는 UDFS는 마운트될 수 없습니다.

관련 정보

RMVMFS(마운트된 파일 시스템 제거) 명령

통합 파일 시스템 사용자 정의 파일 시스템 저장 및 복원

연관된 권한뿐 아니라 모든 사용자 정의 파일 시스템(UDFS) 오브젝트를 저장 및 복원할 수 있습니다.

SAV(오브젝트 저장) 명령을 사용하여 UDFS에 오브젝트를 저장할 수 있으며 RST(오브젝트 복원) 명령을 사용하여 UDFS 오브젝트를 복원할 수 있습니다. 두 명령은 UDFS의 마운트 또는 마운트 해제 상태와 관계없이 기능합니다. 그러나 UDFS 내의 오브젝트와 함께 UDFS 속성도 제대로 저장하려면 UDFS를 마운트 해제해야 합니다.

관련 정보

SAV(오브젝트 저장) 명령

RST(오브젝트 복원) 명령

사용자 정의 파일 시스템에서 오브젝트 변경사항 저널링

사용자 정의 파일 시스템(UDFS)의 일부 오브젝트 유형은 저널링될 수 있습니다. 이 기능을 사용하여 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트의 변경사항을 복구할 수 있습니다.

관련 개념

106 페이지의『오브젝트 저널링』

저널링의 1차 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 복구할 수 있도록 하는 것입니다. 또한 저널링의 주요 사용은 고가용성 또는 작업부하 균형 조절을 위해 다른 시스템에 오브젝트 변경사항을 복제할 때 도움을 주기 위한 것입니다.

사용자 정의 파일 시스템 및 독립 보조 기억장치 풀(ASP)

독립 보조 기억장치 풀(ASP)가 연결 변환되면 "루트"(/) 파일 시스템에 몇 가지 변경사항이 발생합니다.

변경사항은 다음과 같습니다.

- 디렉토리는 독립 ASP의 /dev 디렉토리 내부에 작성됩니다. 이 디렉토리의 이름은 ASP와 연관된 장치 설명 이름과 일치합니다. 요청이 연결변환되기 전에 이 디렉토리가 존재하고 이 디렉토리가 비어 있지 않으면 연결변환은 진행되지만 ASP에서 UDFS에 대한 작업을 할 수 없습니다. 독립 ASP 단절변환이 발생하면 디렉토리명을 변경하거나 디렉토리 내용을 제거하며 연결변환 요청을 다시 시도하십시오.
- /dev/asp_name 디렉토리에서 독립 ASP에 상주하는 모든 UDFS와 연관된 블록 특수 파일 오브젝트를 발견합니다. 시스템이 제공하는 디폴트 UDFS가 항상 있습니다. 디폴트 UDFS의 블록 특수 파일 경로는 /dev/asp_name/QDEFAULT.UDFS입니다.
- 디폴트 UDFS는 /asp_name 디렉토리에 마운트됩니다. /asp_name 디렉토리는 연결변환 요청 전에 존재할 필요는 없습니다. 그러나 이 디렉토리가 존재하면 비어 있어야 합니다. 디렉토리가 비어 있지 않은 경우 ASP는 여전히 연결변환되지만 디폴트 UDFS는 마운트되지 않습니다. 이것이 발생하면 디렉토리명을 변경하거나 디렉토리 내용을 제거한 후에 단절변환 및 연결변환을 다시 시도하거나 디폴트 UDFS를 마운트하기 위해 MOUNT 명령을 사용하십시오.
- 독립 ASP가 1차 또는 2차 ASP이고 디폴트 UDFS가 정상적으로 마운트된 경우 추가 파일 시스템이 마운트됩니다. 독립 ASP QSYS.LIB 파일 시스템은 /asp_name/QSYS.LIB에 마운트됩니다.

  주: 이 파일 시스템은 디폴트 UDFS를 독립적으로 마운트하거나 마운트 해제할 수 없습니다. 이는 항상 자동으로 마운트되거나 마운트 해제됩니다.

  관련 참조

  49 페이지의『독립 ASP QSYS.LIB』

  독립 ASP QSYS.LIB 파일 시스템은 보조 기억장치 풀(ASP)의 i5/OS 라이브러리 구조를 지원합니다. 이 파일 시스템은 독립 ASP에서 데이터베이스 파일과 라이브러리 지원이 관리하는 다른 모든 i5/OS 오브젝트 유형에 대한 액세스를 제공합니다.

라이브러리 파일 시스템(QSYS.LIB)

QSYS.LIB 파일 시스템은 i5/OS 라이브러리 구조를 지원합니다.

이 파일 시스템은 시스템 및 기본 사용자 ASP(보조 기억장치 풀)에서 데이터베이스 파일과 라이브러리 지원이 관리하는 다른 모든 i5/OS 오브젝트 유형에 대한 액세스를 제공합니다.

또한,

- i5/OS 라이브러리와 해당 라이브러리의 오브젝트에 대해 작동하는 모든 사용자 인터페이스와 프로그래밍 인터페이스 지원
- 데이터베이스 파일상에서 조작되는 모든 프로그래밍 언어 및 기능 지원
- i5/OS 오브젝트 관리를 위한 광범위한 관리 지원 제공
- 실제 파일 멤버, 사용자 공간, 저장 파일에 대한 스트림 I/O 조작 지원

통합 파일 시스템이 OS/400 버전 3에 도입되기 전 QSYS.LIB 파일 시스템만이 있었습니다. 어플리케이션을 개발하기 위해 RPG 또는 COBOL과 같은 언어와 DDS와 같은 기능을 사용한 프로그래머는 QSYS.LIB 파일 시스템을 사용했습니다. 출력 대기행렬을 조작하기 위해 명령, 메뉴 및 표시 화면을 사용한 시스템 오퍼레이터는 사용자 프로파일을 작성 및 변경한 시스템 관리자와 같이 QSYS.LIB 파일 시스템을 사용했습니다.

모든 기능 및 여기에 기초한 어플리케이션은 통합 파일 시스템의 도입 이전과 마찬가지로 작동됩니다. 그러나 이러한 기능은 통합 파일 시스템 인터페이스를 통해 QSYS.LIB에 액세스할 수 없습니다.

통합 파일 시스템 인터페이스를 통한 QSYS.LIB 액세스

i5/OS 파일 서버나 통합 파일 시스템 명령, 사용자 화면 또는 API를 사용하여 통합 파일 인터페이스를 통해 QSYS.LIB 파일 시스템에 액세스할 수 있습니다.

QSYS.LIB 파일 시스템에서 QPWFSERVER 권한 부여 리스트

QPWFSERVER는 리모트 클라이언트를 통해 액세스되는 QSYS.LIB 파일 시스템의 모든 오브젝트에 대한 추가 액세스 요구사항을 제공하는 권한 부여 리스트(오브젝트 유형 *AUTL)입니다.

이 권한 부여 리스트에 지정된 권한들은 QSYS.LIB 파일 시스템 내의 모든 오브젝트에 적용됩니다.

이 오브젝트에 대한 디폴트 권한은 PUBLIC *USE 권한입니다. 관리자는 EDTAUTL(권한 부여 리스트 편집) 또는 WRKAUTL(권한 부여 리스트에 대한 작업) 명령을 사용하여 이 권한의 값을 변경할 수 있습니다. 관리자는 권한 부여 리스트에 PUBLIC *EXCLUDE 권한을 할당하여 일반 사용자들이 리모트 클라이언트에서 QSYS.LIB 오브젝트에 액세스할 수 없게 합니다.

QSYS.LIB 파일 시스템에서 파일 처리 제한사항

다음은 ASP QSYS.LIB 파일 시스템에서 파일을 처리할 때 알아두어야 할 제한사항입니다.

- 논리 파일은 지원되지 않습니다.
- 텍스트 모드 액세스를 위해 지원되는 실제 파일은 단일 텍스트 필드가 들어 있는 소스 실제 파일 및 단일 필드가 들어 있는 프로그램 설명 실제 파일입니다. 2진 모드 액세스를 위해 지원되는 실제 파일은 텍스트 모드 액세스를 위해 지원되는 파일에 추가하여 외부 설명 실제 파일을 포함합니다.
- 바이트 범위 잠금 기능은 지원되지 않습니다. 바이트 범위 잠금에 대한 자세한 정보는 fcntl()--파일 제어 명령 수행 주제를 참조하십시오.

- 데이터베이스 파일 멤버를 열어 작업을 하는 경우, 어느 경우라도 하나의 작업만이 파일 멤버에 대한 쓰기 권한을 가집니다. 기타 요구에는 읽기 액세스만이 허용됩니다.

QSYS.LIB 파일 시스템에서 사용자 공간 지원

QSYS.LIB는 사용자 공간 오브젝트에 대한 스트림 입출력 조작을 지원합니다.

예를 들어, 프로그램은 스트림 데이터를 사용자 공간에 쓰고 사용자 공간으로부터 데이터를 읽습니다. 사용자 공간의 최대 크기는 16,776,704바이트입니다.

사용자 공간은 CCSID(코드화 문자 세트 식별자)로 표시되지 않음에 유의하십시오. 따라서 리턴되는 CCSID가 해당 작업의 디폴트 CCSID입니다.

QSYS.LIB 파일 시스템에서 저장 파일 지원

QSYS.LIB 파일 시스템은 저장 파일 오브젝트에 스트림 I/O 조작을 지원합니다.

예를 들면, 기존의 다른 빈 저장 파일 오브젝트에 데이터를 배치할 필요가 있을 때까지 기존 저장 파일에는 다른 파일로 복사되거나 읽을 수 있는 데이터가 있습니다. 쓰기에 대해 저장 파일이 열려 있을 경우 파일의 기타 열기 인스턴스는 허용되지 않습니다. 읽기에 대해 파일의 열기 인스턴스가 하나 이상 있는 작업이 없으면 저장 파일은 읽기에 대한 복수의 열기 인스턴스를 허용합니다. 저장 파일은 읽기/쓰기 액세스에 대해 열리지 않을 수 있습니다. 저장 파일 데이터에 스트림 I/O 조작은 작업에서 다중 스레드가 실행 중인 경우 허용되지 않습니다.

저장 파일에 대한 스트림 I/O 조작은 저장 파일이나 디렉토리가 네트워크 파일 시스템을 통해 내보내지고 있는 경우 지원되지 않습니다. 그러나 PC 클라이언트에서 QFileSvr.400 파일 시스템을 통해 액세스할 수 있습니다.

QSYS.LIB 파일 시스템에서 대소문자 구분

일반적으로 QSYS.LIB 파일 시스템은 오브젝명의 대소문자를 구분하지 않습니다.

오브젝트명에 대한 탐색시 이름의 문자가 대문자인지 소문자인지에 관계없이 동일한 것으로 인식합니다.

그러나 이름이 따옴표로 인용된 경우, 이름의 각 문자별 대소문자는 그대로 유지됩니다. 인용된 이름을 포함하여 탐색하는 경우, 인용된 이름의 대소문자를 구분하여 인식합니다.

QSYS.LIB 파일 시스템에서 경로명

경로명의 각 구성요소는 오브젝트명과 오브젝트 유형으로 구성되어야 합니다.

- 예를 들면, 다음과 같습니다.

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

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

  오브젝트명과 오브젝트 유형은 마침표(.)로 분리됩니다. 라이브러리 내의 오브젝트는 서로 다른 오브젝트 유형인 경우 동일한 이름을 가질 수 있으므로, 오브젝트 유형은 고유하게 그 오브젝트를 식별하도록 지정되어야 합니다.

- 각 구성요소의 오브젝트명은 최대 10자가 될 수 있으며, 오브젝트 유형은 최대 6자가 될 수 있습니다.
- QSYS.LIB 내의 디렉토리 계층은 액세스되는 오브젝트의 유형에 따라 2 또는 3 레벨(경로명의 경우 2개 또는 3개의 구성요소)이 될 수 있습니다. 오브젝트가 데이터베이스 파일인 경우, 계층에는 세 개의 레벨(라이브러리, 파일, 멤버)이 포함될 수 있습니다. 그렇지 않은 경우, 두 개의 레벨(라이브러리, 오브젝트)만이 가능합니다. 각 구성요소명의 길이 및 디렉토리 레벨의 수가 경로명의 최대 길이를 결정합니다.

  "루트"(/) 및 QSYS.LIB가 처음 두 레벨로 포함된 경우 QSYS.LIB의 디렉토리 계층은 최대 5개 레벨이 될 수 있습니다.
- 이름 저장시 이름의 문자들은 CCSID 37로 변환됩니다. 그러나 인용된 이름은 작업의 CCSID를 사용하여 저장됩니다.

  CCSID에 대한 자세한 정보는 i5/OS 국제화 주제를 참조하십시오.

  관련 개념

  15 페이지의 『경로명』
  경로명(일부 시스템에서는 *pathname*이라고도 함)은 서버에 오브젝트를 찾는 방법을 알려줍니다.

QSYS.LIB 파일 시스템에서 링크

기호 링크는 QSYS.LIB 파일 시스템에서 작성되거나 저장될 수 없습니다.

라이브러리와 라이브러리 내 오브젝트간의 관계는 라이브러리와 라이브러리 내의 각 오브젝트 사이의 하드 링크와 같습니다. 통합 파일 시스템은 라이브러리-오브젝트 관계를 링크로 처리합니다. 따라서 기호 링크를 지원하는 파일 시스템에서 QSYS.LIB 파일 시스템의 오브젝트로의 링크가 가능합니다.

관련 개념

12 페이지의 『링크』
*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

QSYS.LIB 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

대부분의 통합 파일 시스템 명령 및 화면은 QSYS.LIB 파일 시스템에서 유효합니다.

다음 제한사항을 제외하고 74 페이지의 『CL 명령을 사용한 액세스』에 나열된 명령은 QSYS.LIB 파일 시스템에서 작동할 수 있습니다.

- ADDLNK(링크 추가) 명령은 QSYS.LIB의 오브젝트에 *대한* 기호 링크를 작성하는 경우에만 사용될 수 있습니다.
- 파일 조작은 프로그램 설명 실제 파일 및 소스 실제 파일에서만 수행될 수 있습니다.
- 저널 시작(STRJRN) 및 저널 종료(ENDJRN) 명령은 데이터베이스 실제 파일(PF) 또는 라이브러리에 사용될 수 없습니다.
- 다음 명령은 지원되지 않습니다.
  - 오브젝트 체크 인(CHKIN)

– 오브젝트 체크 아웃(CHKOUT)
– RCLLNK(오브젝트 링크 재생)

동일한 제한사항이 72 페이지의 『메뉴 및 표시 화면을 사용한 액세스』에 설명된 사용자 화면에 적용됩니다.

QSYS.LIB 파일 시스템에서 통합 파일 시스템 API 사용

대부분의 통합 파일 시스템 API는 QSYS.LIB 파일 시스템에서 유효합니다.

다음 제한사항을 제외하고 122 페이지의 『API를 사용한 조작 수행』에 나열된 API는 QSYS.LIB 파일 시스템에서 작동할 수 있습니다.

- 파일 조작은 프로그램 설명 실제 파일 및 소스 실제 파일에서만 수행될 수 있습니다.
- symlink() 함수는 기호 링크를 지원하는 다른 파일 시스템에서 QSYS.LIB에 있는 *오브젝트로의* 링크에만 사용될 수 있습니다.
- QjoStartJournal() 및 QjoEndJournal() API는 데이터베이스 실제 파일 또는 라이브러리에서 사용할 수 없습니다.

관련 정보

어플리케이션 프로그램 인터페이스(API)

독립 ASP QSYS.LIB

독립 ASP QSYS.LIB 파일 시스템은 보조 기억장치 풀(ASP)의 i5/OS 라이브러리 구조를 지원합니다. 이 파일 시스템은 독립 ASP에서 데이터베이스 파일과 라이브러리 지원이 관리하는 다른 모든 i5/OS 오브젝트 유형에 대한 액세스를 제공합니다.

또한,

- 독립 ASP에서 i5/OS 라이브러리와 해당 라이브러리의 오브젝트에 대해 작동하는 모든 사용자 인터페이스와 프로그래밍 인터페이스 지원
- 데이터베이스 파일상에서 조작되는 모든 프로그래밍 언어 및 기능 지원
- i5/OS 오브젝트 관리를 위한 광범위한 관리 지원 제공
- 실제 파일 멤버, 사용자 공간, 저장 파일에 대한 스트림 I/O 조작 지원

통합 파일 시스템 인터페이스를 통한 독립 ASP QSYS.LIB 액세스

i5/OS 파일 서버나 통합 파일 시스템 명령, 사용자 화면 또는 API를 통해 독립 ASP QSYS.LIB 파일 시스템에 액세스할 수 있습니다.

통합 파일 시스템 인터페이스를 사용할 경우 일부 고려사항 및 제한사항을 알고 있어야 합니다.

독립 ASP QSYS.LIB 파일 시스템에서 QPWFSERVER 권한 부여 리스트

QPWFSERVER는 리모트 클라이언트를 통해 액세스되는 독립 ASP QSYS.LIB 파일 시스템의 모든 오브젝트에 대한 추가 액세스 요구사항을 제공하는 권한 부여 리스트(오브젝트 유형 *AUTL)입니다.

이 권한 부여 리스트에 지정된 권한은 독립 ASP QSYS.LIB 파일 시스템 내의 모든 오브젝트에 적용됩니다.

이 오브젝트에 대한 디폴트 권한은 PUBLIC *USE 권한입니다. 관리자는 EDTAUTL(권한 부여 리스트 편집) 또는 WRKAUTL(권한 부여 리스트에 대한 작업) 명령을 사용하여 이 권한의 값을 변경할 수 있습니다. 관리자는 권한 부여 리스트에 PUBLIC *EXCLUDE 권한을 할당하여 일반 사용자들이 리모트 클라이언트에서 독립 ASP QSYS.LIB 오브젝트에 액세스할 수 없게 합니다.

독립 ASP QSYS.LIB 파일 시스템에서 파일 처리 제한사항

다음은 독립 ASP QSYS.LIB 파일 시스템에서 파일을 처리할 때 알아두어야 할 제한사항입니다.

- 논리 파일은 지원되지 않습니다.
- 텍스트 모드 액세스를 위해 지원되는 실제 파일은 단일 텍스트 필드가 들어 있는 소스 실제 파일 및 단일 필드가 들어 있는 프로그램 설명 실제 파일입니다. 2진 모드 액세스를 위해 지원되는 실제 파일은 텍스트 모드 액세스를 위해 지원되는 파일에 추가하여 외부 설명 실제 파일을 포함합니다.
- 바이트 범위 잠금 기능은 지원되지 않습니다. 바이트 범위 잠금에 대한 자세한 정보는 fcntl()--파일 제어 명령 수행 주제를 참조하십시오.
- 데이터베이스 파일 멤버를 열어 작업을 하는 경우, 어느 경우라도 하나의 작업만이 파일 멤버에 대한 쓰기 권한을 가집니다. 기타 요구에는 읽기 액세스만이 허용됩니다.

독립 ASP QSYS.LIB 파일 시스템에서 사용자 공간 지원

독립 ASP QSYS.LIB는 사용자 공간 오브젝트에 대한 스트림 입/출력 조작을 지원합니다.

예를 들어, 프로그램은 스트림 데이터를 사용자 공간에 쓰고 사용자 공간으로부터 데이터를 읽습니다. 사용자 공간의 최대 *크기*는 16,776,704바이트입니다.

사용자 공간은 CCSID(코드화 문자 세트 식별자)로 표시되지 않음에 유의하십시오. 따라서 리턴되는 CCSID가 해당 작업의 디폴트 CCSID입니다.

독립 ASP QSYS.LIB 파일 시스템에서 저장 파일 지원

독립 ASP QSYS.LIB는 저장 파일 오브젝트에 스트림 I/O 조작을 지원합니다.

예를 들면, 기존의 다른 빈 저장 파일 오브젝트에 데이터를 배치할 필요가 있을 때까지 기존 저장 파일에는 다른 파일로 복사되거나 읽을 수 있는 데이터가 있습니다. 쓰기에 대해 저장 파일이 열려 있을 경우 파일의 기타 열기 인스턴스는 허용되지 않습니다. 읽기에 대해 파일의 열기 인스턴스가 하나 이상 있는 작업이 없으면 저장 파일은 읽기에 대한 복수의 열기 인스턴스를 **허용**합니다. 저장 파일은 읽기/쓰기 액세스에 대해 열리지 않을 수 있습니다. 저장 파일 데이터에 스트림 I/O 조작은 작업에서 다중 스레드가 실행 중인 경우 허용되지 않습니다.

저장 파일에 대한 스트림 I/O 조작은 저장 파일이나 디렉토리가 네트워크 파일 시스템을 통해 내보내지고 있는 경우 지원되지 않습니다. 그러나 PC 클라이언트에서 QFileSvr.400 파일 시스템을 통해 액세스할 수 있습니다.

독립 ASP QSYS.LIB 파일 시스템에서 대소문자 구분

독립 ASP QSYS.LIB 파일 시스템은 오브젝트명의 대소문자를 구분하지 않습니다.

오브젝트명에 대한 탐색시 이름의 문자가 대문자인지 소문자인지에 관계없이 동일한 것으로 인식합니다.

그러나 이름이 따옴표로 인용된 경우, 이름의 각 문자별 대소문자는 그대로 유지됩니다. 인용된 이름을 포함하여 탐색하는 경우, 인용된 이름의 대소문자를 구분하여 인식합니다.

독립 ASP QSYS.LIB 파일 시스템에서 경로명

경로명의 각 구성요소는 오브젝트명과 오브젝트 유형으로 구성되어야 합니다.

- 예를 들면, 다음과 같습니다.

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

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

  여기서, asp_name은 독립 ASP의 이름입니다. 오브젝트명과 오브젝트 유형은 마침표(.)로 분리됩니다. 라이브러리 내의 오브젝트는 서로 다른 오브젝트 유형인 경우 동일한 이름을 가질 수 있으므로, 오브젝트 유형은 고유하게 그 오브젝트를 식별하도록 지정되어야 합니다.
- 각 구성요소의 오브젝트명은 최대 10자가 될 수 있으며, 오브젝트 유형은 최대 6자가 될 수 있습니다.
- 독립 ASP QSYS.LIB 내의 디렉토리 계층은 액세스되는 오브젝트의 유형에 따라 2 또는 3 레벨(경로명의 경우 2개 또는 3개의 구성요소)이 될 수 있습니다. 오브젝트가 데이터베이스 파일인 경우, 계층에는 세 개의 레벨(라이브러리, 파일, 멤버)이 포함될 수 있습니다. 그렇지 않은 경우, 두 개의 레벨(라이브러리, 오브젝트)만이 가능합니다. 각 구성요소명의 길이 및 디렉토리 레벨의 수가 경로명의 최대 길이를 결정합니다.

  /, asp_name 및 QSYS.LIB가 처음 세 레벨로 포함된 경우 독립 ASP QSYS.LIB 파일 시스템의 디렉토리 계층은 최대 5 레벨이 될 수 있습니다.
- 이름 저장시 이름의 문자들은 CCSID 37로 변환됩니다. 그러나 인용된 이름은 작업의 CCSID를 사용하여 저장됩니다.

  CCSID에 대한 자세한 정보는 i5/OS Information Center의 i5/OS 국제화를 참조하십시오.

  관련 개념

  15 페이지의『경로명』

  경로명(일부 시스템에서는 *pathname*이라고도 함)은 서버에 오브젝트를 찾는 방법을 알려줍니다.

독립 ASP QSYS.LIB 파일 시스템에서 링크

기호 링크는 독립 ASP QSYS.LIB 파일 시스템에서 작성되거나 저장될 수 없습니다.

라이브러리와 라이브러리 내 오브젝트간의 관계는 라이브러리와 라이브러리 내의 각 오브젝트 사이의 하드 링크와 같습니다. 통합 파일 시스템은 라이브러리-오브젝트 관계를 링크로 처리합니다. 따라서 기호 링크를 지원하는 파일 시스템에서 독립 ASP QSYS.LIB 파일 시스템의 오브젝트로의 링크가 가능합니다.

관련 개념

12 페이지의 『링크』
*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

독립 ASP QSYS.LIB 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

대부분의 통합 파일 시스템 명령 및 화면은 독립 ASP QSYS.LIB 파일 시스템에서 유효합니다.

74 페이지의 『CL 명령을 사용한 액세스』에 나열된 거의 모든 명령은 독립 ASP QSYS.LIB 파일 시스템에서 작동할 수 있습니다. 하지만 몇 가지 예외가 있습니다.

- ADDLNK(링크 추가) 명령은 독립 ASP QSYS.LIB의 오브젝트에 대한 기호 링크를 작성하는 경우에만 사용될 수 있습니다.
- 파일 조작은 프로그램 설명 실제 파일 및 소스 실제 파일에서만 수행될 수 있습니다.
- 저널 시작(STRJRN) 및 저널 종료(ENDJRN) 명령은 데이터베이스 실제 파일(PF) 또는 라이브러리에 사용될 수 없습니다.
- MOV(오브젝트 이동) 명령을 사용하여 독립 ASP QSYS.LIB 파일 시스템의 라이브러리를 기본 보조 기억장치 풀(ASP)로 이동할 수 없습니다. 그러나 독립 ASP QSYS.LIB의 라이브러리를 시스템 ASP나 다른 독립 ASP로 이동할 수는 있습니다.
- SAV(오브젝트 저장) 또는 RST(오브젝트 복원)를 사용하여 독립 ASP의 라이브러리 오브젝트를 저장하거나 복원하려면 해당 독립 ASP를 SAV 또는 RST 조작을 수행하는 작업에 연관시키거나 독립 ASP를 ASPDEV 매개변수에 지정해야 합니다. /asp_name/QSYS.LIB/object.type의 경로명 명명 규칙은 SAV와 RST에서 지원되지 않습니다.
- 다음 명령은 지원되지 않습니다.
  - 오브젝트 체크 인(CHKIN)
  - 오브젝트 체크 아웃(CHKOUT)
  - RCLLNK(오브젝트 링크 재생)

동일한 제한사항이 72 페이지의 『메뉴 및 표시 화면을 사용한 액세스』에 설명된 사용자 화면에 적용됩니다.

독립 ASP QSYS.LIB 파일 시스템에서 통합 파일 시스템 API 사용

대부분의 통합 파일 시스템 API는 독립 ASP QSYS.LIB 파일 시스템에서 유효합니다.

다음 상황을 제외하고 122 페이지의 『API를 사용한 조작 수행』에 나열된 API는 독립 ASP QSYS.LIB 파일 시스템에서 작동할 수 있습니다.

- 파일 조작은 프로그램 설명 실제 파일 및 소스 실제 파일에서만 수행될 수 있습니다.
- symlink() 함수는 기호 링크를 지원하는 다른 파일 시스템에서 독립 ASP QSYS.LIB에 있는 오브젝트로의 링크에만 사용될 수 있습니다.
- QjoStartJournal() 및 QjoEndJournal() API는 데이터베이스 실제 파일 또는 라이브러리에서 사용할 수 없습니다.

- QsrSave() 또는 QsrRestore() API를 사용하여 독립 ASP의 라이브러리 오브젝트를 저장하거나 복원하는 경우 해당 독립 ASP를 저장 또는 복원 조작을 수행하는 작업에 연관시키거나 ASPDEV 키에 독립 ASP를 지정해야 합니다. QsrSave() 및 QsrRestore() API에서는 경로명의 명명 규칙(/asp_name/QSYS.LIB/object.type)이 지원되지 않습니다.

관련 정보

어플리케이션 프로그램 인터페이스(API)

문서 라이브러리 서비스 파일 시스템(QDLS)

QDLS 파일 시스템은 폴더 구조를 지원합니다. 문서 및 폴더로의 액세스를 제공합니다.

또한,

- i5/OS 폴더 및 문서 라이브러리 오브젝트(DLO)를 지원합니다.
- 스트림 파일에 저장된 데이터를 지원합니다.

통합 파일 시스템 인터페이스를 통한 QDLS 액세스

i5/OS 파일 서버나 통합 파일 시스템 명령, 사용자 화면 및 API를 사용하여 통합 파일 시스템 인터페이스를 통해 QDLS 파일 시스템에 액세스할 수 있습니다.

통합 파일 시스템 인터페이스를 사용할 경우 다음 고려사항 및 제한사항을 알고 있어야 합니다.

QDLS 파일 시스템에서 통합 파일 시스템 및 HFS

문서 라이브러리 오브젝트(DLO) CL 명령뿐 아니라 계층 파일 시스템(HFS)에서 제공하는 API나 통합 파일 시스템 인터페이스를 통해 QDLS 파일 시스템의 오브젝트에 대한 조작을 수행할 수 있습니다.

통합 파일 시스템이 Integrated Language Environment®(ILE) 프로그램 모델을 기반으로 하는 반면 HFS는 원래 System i 프로그램 모델을 기반으로 합니다.

HFS API를 사용하여 통합 파일 시스템이 지원하지 않는 몇 개의 추가 조작을 수행할 수 있습니다. 특히, HFS API를 사용하여 디렉토리 확장 속성(*디렉토리 항목 속성*이라고도 함)에 액세스하고 변경할 수 있습니다. HFS API 사용을 위한 명명 규칙은 통합 파일 시스템 인터페이스를 사용하는 API의 명명 규칙과 다르다는 점에 유의하십시오.

관련 정보

계층 파일 시스템 API

QDLS 파일 시스템에서 사용자 등록

QDLS 파일 시스템에서 오브젝트에 대한 작업을 할 때 사용자는 시스템 분배 디렉토리에 등록되어야 합니다.

QDLS 파일 시스템에서 대소문자 구분

QDLS 파일 시스템은 오브젝트명에 사용될 때 소문자(**a** - **z**)를 대문자로 변환합니다. 따라서, 문자만을 사용한 오브젝트명을 탐색하는 경우, 대소문자를 구분하지 않습니다.

QDLS에서 다른 모든 문자는 대소문자를 구분합니다.

관련 정보

폴더 및 문서명

QDLS 파일 시스템에서 경로명

경로명의 각 구성요소는 이름만으로 구성될 수 있습니다.

- 예를 들면, 다음과 같습니다.

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

  또는 다음과 같이 이름과 확장자(DOS 파일 확장자와 유사)로 구성될 수도 있습니다.

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

- 각 구성요소의 이름은 최대 8자로 될 수 있고, 확장자는 최대 3자가 될 수 있습니다. 경로명의 최대 길이는 /QDLS로 시작하는 절대 경로명이라고 가정할 때 82자입니다.
- 문서 라이브러리 서비스(QDLS) 파일 시스템 내 디렉토리 계층은 32 레벨까지 가능합니다. / 및 QDLS가 처음 두 레벨로 포함된 경우 디렉토리 계층은 34 레벨이 될 수 있습니다.
- 이름 문자는 데이터 영역 Q0DEC500이 작성되지 않는 한, 이름이 저장될 때 작업의 코드 페이지로 변환됩니다. 데이터 영역이 존재하면, 이름이 저장될 때 이름에 있는 문자는 코드 페이지 500으로 변환됩니다. 이 기능은 이전 릴리스의 QDLS 파일 시스템 방식과의 호환성을 제공합니다. 해당 코드 페이지로 변환될 수 없는 경우에는 이름이 거부될 수 있습니다.

  코드 페이지에 대한 자세한 정보는 i5/OS Information Center의 i5/OS 국제화를 참조하십시오.

관련 개념

15 페이지의 『경로명』

경로명(일부 시스템에서는 *pathname*이라고도 함)은 서버에 오브젝트를 찾는 방법을 알려줍니다.

QDLS 파일 시스템에서 링크

기호 링크는 QDLS 파일 시스템에서 작성 또는 저장될 수 없습니다.

통합 파일 시스템은 폴더와 폴더 내 문서 라이브러리 오브젝트 간의 관계를 폴더와 폴더 내 각 오브젝트 간의 링크와 같은 것으로 처리합니다. 따라서 기호 링크를 지원하는 파일 시스템에서 QDLS 파일 시스템에 있는 오브젝트와의 링크가 가능합니다.

관련 개념

12 페이지의 『링크』

*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

QDLS 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

대부분의 통합 파일 시스템 명령 및 화면은 QDLS 파일 시스템에서 유효합니다.

다음 명령을 제외하고 74 페이지의 『CL 명령을 사용한 액세스』에 나열된 명령은 QDLS 파일 시스템에서 작동할 수 있습니다.

- ADDLNK 명령은 기호 링크를 지원하는 다른 파일 시스템에서 QDLS의 오브젝트로 링크하기 위해서만 사용될 수 있습니다.
- CHKIN 및 CHKOUT 명령은 문서에 대해서 지원되고 폴더에 대해서는 지원되지 않습니다.
- 다음 명령은 지원되지 않습니다.
  - APYJRNCHG
  - CHGJRNOBJ
  - DSPJRN
  - ENDJRN
  - RCLLNK
  - RCVJRNE
  - RTVJRNE
  - SNDJRNE
  - STRJRN

동일한 제한사항이 72 페이지의 『메뉴 및 표시 화면을 사용한 액세스』에 설명된 사용자 화면에 적용됩니다.

QDLS 파일 시스템에서 통합 파일 시스템 API 사용

대부분의 통합 파일 시스템 API는 QDLS 파일 시스템에서 유효합니다.

다음 API를 제외하고 122 페이지의 『API를 사용한 조작 수행』에 나열된 API는 QDLS 파일 시스템에서 작동할 수 있습니다.

- symlink() 함수는 기호 링크를 지원하는 다른 파일 시스템에서 QDLS에 있는 오브젝트로의 링크에만 사용될 수 있습니다.
- 다음 함수는 지원되지 않습니다.
  - givedescriptor()
  - ioctl()
  - link()
  - QjoEndJournal()
  - QjoRetrieveJournalEntries()
  - QjoRetrieveJournalInformation()
  - QJORJIDI()
  - QJOSJRNE()
  - QjoStartJournal()
  - readlink()

– takedescriptor()

관련 정보

어플리케이션 프로그램 인터페이스(API)

광 파일 시스템(QOPT)

QOPT 파일 시스템은 광 매체에 저장된 스트림 데이터에 대한 액세스를 제공합니다.

또한,

- DOS 및 OS/2와 같은 오퍼레이팅 시스템과 유사한 계층적 디렉토리 구조를 제공합니다.
- 스트림 파일 입출력(I/O)에 대해 최적화되었습니다.
- 스트림 파일에 저장된 데이터를 지원합니다.

통합 파일 시스템을 통한 QOPT 액세스

QOPT 파일 시스템은 PC 서버 또는 통합 파일 시스템 명령, 사용자 화면 및 API를 통해 액세스할 수 있습니다.

통합 파일 시스템 인터페이스를 사용할 경우 다음 고려사항 및 제한사항을 알고 있어야 합니다.

관련 정보

광 기억장치

QOPT 파일 시스템에서 통합 파일 시스템 및 HFS

계층 파일 시스템(HFS)에서 제공하는 API 또는 통합 파일 시스템 인터페이스를 통해 QOPT 파일 시스템의 오브젝트에 대해 조작을 수행할 수 있습니다.

통합 파일 시스템이 Integrated Language Environment(ILE) 프로그램 모델을 기반으로 하는 반면 HFS는 원래 System i 프로그램 모델을 기반으로 합니다.

HFS API를 사용하여 통합 파일 시스템이 지원하지 않는 몇 개의 추가 조작을 수행할 수 있습니다. 특히, HFS API를 사용하여 디렉토리 확장 속성(*디렉토리 항목 속성*이라고도 함)을 변경 또는 액세스하고, 보유하고 있는 광 파일에 대한 작업을 할 수 있습니다. HFS API 사용을 위한 명명 규칙은 통합 파일 시스템 인터페이스를 사용하는 API의 명명 규칙과 다르다는 점에 유의하십시오.

HFS API에 대한 자세한 정보는 광 장치 프로그래밍 주제 콜렉션을 참조하십시오.

관련 정보

계층 파일 시스템 API

QOPT 파일 시스템에서 대소문자 구분

QOPT에서 파일이나 디렉토리를 작성할 때 광 매체 형식에 따라 대소문자가 보존될 수도 있고 그렇지 않을 수도 있습니다. 그러나 파일 및 디렉토리 탐색은 광 매체 형식에 관계없이 대소문자를 구분합니다.

QOPT 파일 시스템에서 경로명

경로명은 슬래시(/)로 시작해야 합니다. 경로는 파일 시스템명, 디렉토리명, 서브디렉토리명 및 파일명으로 구성됩니다.

- 예를 들면, 다음과 같습니다.

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

- 파일 시스템명 QOPT가 필요합니다.
- 볼륨 및 경로명 길이는 광 매체 형식에 따라 다릅니다.
- 경로명에 /QOPT를 지정하거나 하나 이상의 디렉토리 또는 서브디렉토리를 경로명에 포함시킬 수 있습니다. 디렉토리명 및 파일명에는 X'00'에서 X'3F', X'FF'를 제외한 모든 문자를 사용할 수 있습니다. 추가 제한사항은 광 매체 형식을 기준으로 적용됩니다.
- 파일명은 경로명의 최종 요소가 됩니다. 파일명 길이는 경로 내의 디렉토리명의 길이에 의해 제한을 받습니다.

QOPT 파일 시스템의 경로명 규칙에 대한 자세한 내용은 경로명의 『경로명 규칙』 설명을 참조하십시오.

관련 개념

15 페이지의 『경로명』

경로명(일부 시스템에서는 *pathname*이라고도 함)은 서버에 오브젝트를 찾는 방법을 알려줍니다.

QOPT 파일 시스템에서 링크

QOPT 파일 시스템은 한 오브젝트에 대해 하나의 링크만을 지원합니다. 기호 링크는 QOPT에 작성되거나 저장될 수 없습니다. 그러나 QOPT의 파일은 "루트"(/), QOpenSys 또는 사용자 정의 파일 시스템으로부터 기호 링크를 사용하여 액세스할 수 있습니다.

관련 개념

12 페이지의 『링크』

*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

QOPT 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

대부분의 통합 파일 시스템 명령 및 화면은 QOPT 파일 시스템에서 유효합니다.

74 페이지의 『CL 명령을 사용한 액세스』에 나열된 대부분의 명령을 QOPT 파일 시스템에서 작동시킬 수 있습니다. 그러나 QOPT 파일 시스템에 몇 가지 예외가 있습니다. 그러나 다중 스레드가 가능한 프로세스에서 CL 명령들을 사용하는 것은 안전하지 않을 수 있습니다. 광 매체 형식에 따라 특정 제한사항이 적용됩니다. 동일한 제한사항이 72 페이지의 『메뉴 및 표시 화면을 사용한 액세스』에 설명된 사용자 화면에 적용됩니다.

다음 통합 파일 시스템 명령은 QOPT 파일 시스템에서 지원되지 않습니다.

- ADDLNK
- APYJRNCHG

- CHGJRNOBJ
- CHKIN
- CHKOUT
- DSPJRN
- ENDJRN
- RCLLNK
- RCVJRNE
- RTVJRNE
- SNDJRNE
- STRJRN
- WRKOBJOWN
- WRKOBJPGP

QOPT 파일 시스템에서 통합 파일 시스템 API 사용

대부분의 통합 파일 시스템 API는 QOPT 파일 시스템에서 유효합니다.

다음 API를 제외하고 122 페이지의 『API를 사용한 조작 수행』에 나열된 모든 API는 다음 API를 제외하고 스레드세이프 방식으로 QOPT 파일 시스템에서 작동할 수 있습니다.

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

관련 정보

어플리케이션 프로그램 인터페이스(API)

i5/OS NetClient 파일 시스템(QNTC)

QNTC 파일 시스템은 Windows NT 4.0 이상 또는 Linux 오퍼레이팅 시스템을 실행 중인 통합 xSeries 서버(IXS)에 저장된 데이터 및 오브젝트에 대한 액세스를 제공합니다. QNTC 파일 시스템은 Windows NT 4.0 이상, Linux Samba 3.0 이상 또는 지원되는 버전의 i5/OS NetServer를 실행 중인 리모트 서버에 저장된 데이터 및 오브젝트에 대한 액세스도 제공합니다.

QNTC 파일 시스템은 기본 i5/OS 오퍼레이팅 시스템의 일부입니다. /QNTC에 액세스하기 위해 통합 서버 지원, 오퍼레이팅 시스템의 옵션 29를 설치할 필요가 없습니다.

통합 파일 시스템 인터페이스를 통한 QNTC 액세스

i5/OS NetServer, System i Navigator, 통합 파일 시스템 명령, 사용자 화면 또는 API를 통합 파일 시스템 인터페이스를 통해 QNTC 파일 시스템에 액세스할 수 있습니다.

다음 고려사항 및 제한사항에 유의하십시오.

QNTC 파일 시스템에서 권한 및 소유권

QNTC 파일 시스템은 파일이나 디렉토리의 소유권 개념을 지원하지 않습니다.

따라서, QNTC에 저장된 파일의 소유권을 변경하는 명령이나 API를 사용하면 실패하게 됩니다. QDFTOWN이라는 시스템 사용자 프로파일에서 QNTC에 있는 모든 파일과 디렉토리를 소유합니다.

NT 서버 파일과 디렉토리에 대한 권한은 Windows NT 서버에서 관리합니다. QNTC는 WRKAUT 및 CHGAUT 명령을 지원하지 않습니다.

QNTC 파일 시스템에서 대소문자 구분

QNTC 파일 시스템은 입력된 오브젝트명의 대소문자를 동일하게 보존하지만 이름의 대소문자를 구분하지는 않습니다. 그래서, 서버 파일 시스템이 대소문자를 구분한다면 QNTC 파일 시스템에서 경로명의 대소문자는 적절히 지정되어야 합니다.

오브젝트명에 대한 탐색시 이름의 문자가 대문자인지 소문자인지에 관계없이 동일한 것으로 인식합니다.

QNTC 파일 시스템에서 경로명

경로는 파일 시스템명, 서버명, 공유명, 디렉토리 및 서브디렉토리명, 오브젝트명으로 구성됩니다.

경로명의 요구사항은 다음과 같습니다.

- 경로명은 반드시 슬래시(/)로 시작하여 최고 255자까지 사용이 가능합니다. 경로명은 다음과 같은 형식으로 되어 있습니다.

```
          /QNTC/Servername/Sharename/Directory/ . . . /Object
. . /Object(QNTC는 경로명의 필수 부분입니다. )
```

- 서버명은 QNTC 경로명의 필수 부분입니다. 이 서버 이름은 TCP.IP 호스트명, NetBIOS 이름 또는 TCP/IP 주소일 수 있습니다. V6R1부터 IPv4뿐만 아니라 IPv6 주소가 지원됩니다.
- 공유명은 최대 12자 길이일 수 있습니다.
- 공유명 뒤에 오는 경로명의 각 구성요소는 최대 255자 길이일 수 있습니다.
- QNTC 안에서는 일반적으로 130개 레벨의 계층을 사용할 수 있습니다. 경로명의 모든 구성요소가 계층 레벨에 포함되어 있는 경우, 최고 132개까지의 디렉토리 레벨을 사용할 수 있습니다.
- 이름은 유니코드 CCSID로 저장됩니다.
- 디폴트로 로컬 서브네트의 각 기능적 지원 서버는 자동으로 /QNTC 아래의 디렉토리로 표시됩니다. 로컬 서브네트 외부에 위치한 액세스 가능한 시스템을 추가하려면 CRTDIR(디렉토리 작성) 명령 또는 mkdir() API를 사용하십시오.

관련 개념

15 페이지의 『경로명』

경로명(일부 시스템에서는 *pathname*이라고도 함)은 서버에 오브젝트를 찾는 방법을 알려줍니다.

관련 정보

MKDIR(디렉토리 작성) 명령

mkdir()--디렉토리 API 작성

i5/OS 용어집

QNTC 파일 시스템에서 링크

QNTC 파일 시스템은 한 오브젝트에 대해 하나의 링크만을 지원합니다. QNTC에는 기호 링크를 작성하거나 저장할 수 없습니다.

『루트』(/) 또는 QOpenSys 파일 시스템에서 기호 링크를 사용하여 QNTC의 데이터에 액세스할 수 있습니다.

관련 개념

12 페이지의 『링크』

*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

QNTC 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

대부분의 통합 파일 시스템 명령 및 화면은 QNTC 파일 시스템에서 유효합니다.

다음 명령을 제외하고 74 페이지의 『CL 명령을 사용한 액세스』에 나열된 명령은 QNTC 파일 시스템에서 작동할 수 있습니다.

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

- RTVJRNE
- RST(통합 xSeries 서버와 함께 사용할 수 있음)
- SAV(통합 xSeries 서버와 함께 사용할 수 있음)
- SNDJRNE
- STRJRN
- WRKAUT
- WRKOBJOWN
- WRKOBJPGP

동일한 제한사항이 72 페이지의 『메뉴 및 표시 화면을 사용한 액세스』에 설명된 사용자 화면에 적용됩니다.

QNTC 파일 시스템에서 통합 파일 시스템 API 사용

대부분의 통합 파일 시스템 API는 QNTC 파일 시스템에서 유효합니다.

다음 API를 제외하고 122 페이지의 『API를 사용한 조작 수행』에 나열된 API는 QNTC 파일 시스템에서 작동할 수 있습니다.

- chmod(), fchmod(), utime() 및 umask() 함수는 QNTC 내의 오브젝트에 영향을 주지 않지만 사용하는 것이 오류를 일으키지는 않습니다.
- QNTC 파일 시스템은 다음 함수를 지원하지 않습니다.
  - chown()
  - fchown()
  - fclear()
  - fclear64()
  - givedescriptor()
  - link()
  - QjoEndJournal()
  - QjoRetrieveJournalEntries()
  - QjoRetrieveJournalInformation()
  - QJORJIDI()
  - QJOSJRNE()
  - QjoStartJournal()
  - Qp0lGetPathFromFileID()
  - readlink()
  - symlink()
  - takedescriptor()
- QNTC 파일 시스템은 다음 함수 수행 시 setlimit() API를 사용하여 설정된 모든 자원 제한을 무시합니다.

- write()
- writev()
- pwrite()
- pwrite64()

관련 정보

어플리케이션 프로그램 인터페이스(API)

QNTC 환경 변수

QNTC의 네트워크 찾아보기 작동은 두 가지 환경 변수로 제어할 수 있습니다. 이러한 환경 변수에 대한 지원은 i5/OS V5R4에서 시작되었습니다. 이러한 환경 변수를 작성하려면 ADDENVVAR CL 명령을 사용하십시오.

QZLC_SERVERLIST

이 환경 변수가 ″2″로 설정되면 통합 파일 시스템의 /QNTC 디렉토리에 표시되는 모든 서버를 QNTC에서 액세스할 수 있습니다. 이것은 V5R4 이전의 디폴트 작동입니다. 이 변수가 ″2″로 설정되지 않거나 작성되지 않으면 /QNTC 디렉토리에 표시되는 일부 서버를 액세스할 수 없을 수도 있습니다.

QIBM_ZLC_NO_BROWSE

이 환경 변수가 ″1″로 설정되면 /QNTC 디렉토리는 CRTDIR CL 명령 또는 mkdir() API로 작성된 서버만을 포함합니다. 이 환경 변수가 설정되면 QNTC 파일 시스템에 대한 대다수 조작의 성능이 향상됩니다. 하지만 모든 /QNTC 디렉토리를 CL 명령을 사용하여 작성해야 합니다.

QNTC 파일 시스템에서 디렉토리 작성

/QNTC 디렉토리에 서버 디렉토리를 추가하려면 CRTDIR(디렉토리 작성) 명령 또는 mkdir() API를 사용할 수 있습니다.

디폴트로, QNTC 디렉토리는 i5/OS NetServer 정의역 및 로컬 서브네트의 모든 기능적 서버에 대해 자동으로 작성됩니다. 로컬 서브네트 또는 i5/OS NetServer 정의역 외부의 서버는 CRTDIR 명령 또는 mkdir() API를 사용하여 추가해야 합니다. 예를 들면, 다음과 같습니다.

```
CRTDIR '/QNTC/NTSRV1'
```

QNTC 파일 시스템 디렉토리 구조에 NTSRV1 서버를 추가시켜 그 서버에서 파일이나 디렉토리의 액세스가 작동할 수 있도록 합니다.

또한 TCP/IP 주소를 사용하여 디렉토리 구조에 새로운 서버를 추가할 수도 있습니다. 이 서버 이름은 IPv4 주소 또는 IPv6 주소일 수 있습니다. 예를 들면, 다음과 같습니다.

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

또는

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

QNTC 파일 시스템 디렉토리 구조에 서버를 추가합니다.

주:

- WINS용 i5/OS NetServer를 구성하여 서브네트를 넘어 자동으로 서버용 디렉토리를 작성할 수 있습니다.
- CRTDIR CL 명령 또는 mkdir() API를 사용하여 디렉토리 구조에 디렉토리를 추가하는 경우 시스템 IPL을 수행한 후 또는 기억장치 재생 명령(RCLSTG)을 실행한 후 디렉토리를 시각적으로 볼 수 없습니다. CRTDIR 명령 또는 mkdir() API는 시스템 IPL 수행 후 또는 RCLSTG 명령을 해당 디렉토리에 실행한 후에 다시 발행되어야 합니다.

API 또는 CL 명령을 사용하여 디렉토리를 추가하려는 경우 다음 예와 같이 QIBM_ZLC_NO_BROWSE 환경 변수를 추가하여 명령의 성능을 증가시킬 수 있습니다.

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

이 환경 변수는 파일 시스템이 파일 조작 수행 시 모든 네트워크 찾아보기를 바이패스하도록 합니다.

관련 정보

MKDIR(디렉토리 작성) 명령

mkdir()--디렉토리 API 작성

네트워크 인증 서비스를 위해 QNTC 파일 시스템 작동

QNTC 파일 시스템을 사용하면 Kerberos V5 인증 프로토콜을 지원하는 System i 플랫폼에서 공통 통합 파일 시스템(CIFS) 서버를 액세스할 수 있습니다.

올바로 구성된 System i 플랫폼은 LAN 관리자를 사용하여 각 서버에 인증하기 위해 암호를 입력하기 보다 이제 단일 로그온 트랜잭션으로 지원되는 CIFS 서버에 액세스할 수 있습니다.

네트워크 인증 서비스(NAS)를 QNTC와 함께 사용할 수 있게 하려면 다음 항목을 구성해야 합니다.

- 네트워크 인증 서비스(NAS)
- EIM(Enterprise Identity Mapping)

위의 항목이 구성되었으면 사용자가 QNTC 파일 시스템과 함께 NAS를 사용하도록 할 수 있습니다. 다음은 사용자에게 QNTC NAS 지원을 이용하게 하기 위해 필요한 단계입니다.

- 사용자의 i5/OS 사용자 프로파일에 있는 로컬 암호 관리 매개변수(LCLPWDMGT)가 *NO로 설정되어야 합니다. *NO로 지정하면 사용자가 시스템에 대한 암호를 가질 수 없으며 5250 세션에 사인 온할 수 없습니다. 이 서버에 대한 액세스는 System i Navigator 또는 System i Access 5250 표시장치 에뮬레이터와 같이 NAS를 작동할 수 있는 어플리케이션을 통해서만 가능합니다.

  사용자가 *YES로 지정하면 서버에서 암호를 관리하며 NAS 없이 사용자가 인증됩니다.
- Kerberos 티켓 및 System i Navigator 연결을 가지고 있어야 합니다.

- 사용하고 있는 System i 플랫폼에 대한 Kerberos 티켓은 전달가능해야 합니다. 티켓을 전달가능하게 하려면 다음 단계를 수행하십시오.
  1. NAS 영역에 대해 KDC에서 **Active Directory 사용자 및 컴퓨터** 툴에 액세스하십시오.
  2. 사용자를 선택하십시오.
  3. 서비스 프린시펄명에 해당하는 이름을 선택하십시오.
  4. **등록 정보**를 선택하십시오.
  5. **계정** 탭을 선택하십시오.
  6. **계정은 위임에 대해 신뢰할 수 있음**을 선택하십시오.

  관련 정보

  네트워크 인증 서비스

  EIM(Enterprise Identity Mapping)

i5/OS 파일 서버 파일 시스템(QFileSvr.400)

QFileSvr.400 파일 시스템은 리모트 System i 플랫폼에 상주하는 라은 파일 시스템에 대한 투명한 액세스를 제공합니다. 이는 계층 디렉토리 구조를 통해 액세스됩니다.

QFileSvr.400 파일 시스템은 사용자를 대신하여 파일 요구를 수행하는 클라이언트로 간주될 수 있습니다. QFileSvr.400은 목표 시스템의 i5/OS 파일 서버와 대화하여 실제 파일 조작을 수행합니다.

통합 파일 시스템 인터페이스를 통한 QFileSvr.400 액세스

i5/OS 파일 서버나 통합 파일 시스템 명령, 사용자 화면 및 API를 사용하여 통합 파일 인터페이스를 통해 QFileSvr.400 파일 시스템에 액세스할 수 있습니다.

통합 파일 시스템 인터페이스를 사용할 경우 다음 고려사항 및 제한사항을 알고 있어야 합니다.

주: QFileSvr.400 파일 시스템의 특성은 목표 서버에서 액세스되는 파일 시스템의 특성에 따라 판별됩니다.

QFileSvr.400 파일 시스템에서 대소문자 구분

실제로 목표 시스템의 『루트』(/) 디렉토리를 표시하는 첫 번째 레벨 디렉토리의 경우 QFileSvr.400 파일 시스템은 오브젝트명이 입력되는 형식과 같은 대소문자입니다.

그러나 QFileSvr.400에서 이름을 탐색할 때 대소문자를 구분하지 않습니다.

다른 모든 디렉토리의 경우 대소문자 구분은 액세스되는 특정 파일 시스템에 따라 달라집니다. QFileSvr.400은 i5/OS 파일 서버로 파일 요구가 송신될 때 입력된 오브젝트명과 같은 대소문자 형식을 유지합니다.

QFileSvr.400 파일 시스템에서 경로명

QFileSvr.400 파일 시스템에서 경로명은 고유 양식을 가집니다.

- 양식은 다음과 같습니다.

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

제 1 레벨 디렉토리(즉, 이전 예에 있는 RemoteLocationName)는 다음 두 속성 모두를 나타냅니다.

– 통신 연결 설정에 사용될 목표 시스템명. 대상 시스템은 다음 이름 중 하나입니다.

  - TCP/IP 호스트명(예: beowulf.newyork.corp.com)

    주: 대상 시스템이 V6R1에 있다면 호스트명은 IPv6 또는 IPv6으로 해결할 수 있는 것으로 합니다. V6R1 이전 릴리스에서 IPv4 주소만 지원됩니다.

  - SNA LU 6.2면(예: appn.newyork)

– 대상 시스템의 『루트』(/) 디렉토리

이 표시로 인해 제 1 레벨 디렉토리 작성 시 지정된 모든 속성은 무시됩니다.

이 파일 시스템을 사용하려면 첫 번째 레벨 디렉토리를 작성해야 합니다. 디렉토리를 생성하는 모든 통합 파일 시스템 인터페이스를 사용하여 이것을 할 수 있습니다.

주: 제 1 레벨 디렉토리는 다음 IPL시에 계속되지 않습니다. 다시 말해, 제 1 레벨 디렉토리는 IPL 후 다시 작성해야 합니다.

- 경로명의 각각 구성요소는 최대 255자가 될 수 있습니다. 완전한 경로명은 최대 16MB가 될 수 있습니다.

  주: 오브젝트에 상주하는 파일 시스템은 구성요소 길이 및 경로명 길이를 QFileSvr.400에서 허용하는 최대값보다 작게 제한할 수 있습니다.

- 프로그램 및 시스템 제한과 액세스 중인 파일 시스템에 의해 부과되는 제한을 제외하고 디렉토리 계층 깊이에는 제한이 없습니다.
- 이름 저장시 이름의 문자들은 유니코드 양식으로 변환됩니다.

관련 개념

18 페이지의 『이름 연속성』

『루트』(/), QOpenSys 및 사용자 정의 파일 시스템을 사용할 때 오브젝명의 문자가 동일하게 남아 있도록 하는 시스템 지원을 이용할 수 있습니다.

15 페이지의 『경로명』

경로명(일부 시스템에서는 *pathname*이라고도 함)은 서버에 오브젝트를 찾는 방법을 알려줍니다.

QFileSvr.400 파일 시스템에서 통신

QFileSvr.400 파일 시스템은 다음 방식으로 통신합니다.

- 목표 서버에서 파일 서버와의 TCP 연결은 목표 서버의 QSERVER 서브시스템이 사용 중일 때에만 설정될 수 있습니다.
- 사용되지 않는 로컬 제어 세션(예를 들어, LU 6.2 연결시 특별히 성립되는 세션)이 있는 경우에 한해 SNA LU 6.2 연결을 시도할 수 있습니다. LU 6.2 연결 시 QFileSvr.400 파일 시스템은 BLANK 모드를 사용합니다. 목표 시스템상에서 QPWFSERV라는 작업은 QSERVER 서브시스템으로 제출됩니다. 사용자 프로파일은 BLANK 모드에 대한 통신 항목에 의해 정의됩니다. LU 6.2 통신에 대한 자세한 정보는 APPC Programming  을 참조하십시오.

- 통신 프로토콜로 TCP를 사용하는 파일 서버 요청은 요청하는 작업의 문맥(context) 내에서 수행됩니다. SNA를 통신 프로토콜로 사용하는 파일 서버 요청은 i5/OS 시스템 작업인 Q400FILSVR에 의해 수행됩니다.
- 목표 서버에 연결이 아직 설정되지 않은 경우 QFileSvr.400 파일 시스템은 첫 번째 레벨 디렉토리가 TCP/IP 호스트명을 나타낸다고 가정합니다. QFileSvr.400 파일 시스템은 다음 단계를 통해 목표 서버와 연결을 설정합니다.

  1. 리모트 위치명을 IP 주소로 해결하십시오.

     주: 대상 시스템이 V6R1에 있다면 리모트 위치명은 IPv6 또는 IPv6으로 해결할 수 있는 것으로 합니다. V6R1 이전 릴리스에서 IPv4 주소만 지원됩니다.

  2. 해결된 IP 주소를 사용하여 잘 알려진 포트인 449상의 호스트 서버의 서버 맵퍼로 연결하십시오. 그런 후, 서버 맵퍼에 서비스명 『as-file』에 대한 쿼리를 보내십시오. 쿼리 결과에 따라 다음 중 하나가 발생합니다.
     - 『as-file』이 목표 서버의 서비스 표에 있는 경우, 서버 맵퍼는 i5/OS 파일 서버 디먼이 청취하고 있는 포트를 리턴합니다.
     - 서버 맵퍼가 목표 서버에서 사용 중이 아닌 경우, 『as-file』에 디폴트 포트 번호(8473)가 사용됩니다.

     그러면 QFileSvr.400 파일 시스템이 목표 서버에서 i5/OS 파일 서버 디먼과 TCP 연결을 설정하려고 시도합니다. 연결이 설정되면 QFileSvr.400은 파일 서버와 요청 및 응답을 교환합니다. QSERVER 서브시스템에서 QPWFSERVSO 사전시작 요청은 연결을 제어합니다. 사전시작 작업은 각자의 사용자 프로파일 아래서 수행됩니다.

  3. 리모트 위치명이 IP 주소로 해결되지 않는 경우, 제 1 레벨 디렉토리가 SNA LU 6.2명으로 간주됩니다. 따라서, APPC를 i5/OS 파일 서버로 연결하려는 시도가 이루어집니다.

- QFileSvr.400 파일 시스템은 주기적으로(2시간마다) 사용하지 않는 연결이 있는지(예를 들어, 연결과 연관된 열린 파일이 없는지), 2시간 동안 연결에 활동이 없었는지 판별합니다. 그러한 연결이 발견되는 경우, 연결은 종료됩니다.
- QFileSvr.400 파일 시스템은 루프를 감지할 수 없습니다. 다음 경로명은 루프의 예입니다.

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

  여기서, Remote1은 로컬 시스템입니다. 루프가 포함된 경로명이 지정되면 QFileSvr.400 파일 시스템이 짧은 기간 경과 후 오류를 리턴합니다. 시간종료가 발생했다는 오류가 표시됩니다.
- QFileSvr.400 파일 시스템은 SNA를 통하여 통신할 때는 기존의 사용 가능한 세션을 사용합니다. QFileSvr.400이 성공적으로 리모트 통신 시스템에 연결되기 위해서는 모드를 시작하고 세션을 성립해야 합니다.

QFileSvr.400 파일 시스템에서 보안 및 오브젝트 권한

두 시스템에 네트워크 인증 서비스 및 EIM(Enterprise Identity Mapping)이 구성되어 있고 사용자가 Kerberos로 인증된 경우 Kerberos를 사용하여 목표 System i 플랫폼에 상주하는 파일 시스템에 액세스하도록 인증할 수 있습니다.

- Kerberos 인증이 실패하면 사용자 ID와 암호를 사용하여 액세스를 확인합니다.

주: 대상 시스템이 사용자의 액세스를 확인한 후 티켓 허가 티켓이나 System i 티켓이 만기되는 경우, 대상 시스템과의 연결이 종료될 때까지는 만기가 유효하지 않습니다.

- 대상 System i 플랫폼에 상주하는 파일 시스템에 액세스하려면 사용자는 Kerberos가 인증에 사용되지 않을 경우, 로컬 시스템의 사용자 ID 및 암호와 일치하는 사용자 ID 및 암호를 대상 시스템에 갖고 있어야 합니다.

  주: 대상 시스템이 사용자의 액세스를 확인한 후, 로컬 또는 대상 시스템의 사용자 암호가 변경되면, 변경사항은 대상 시스템과의 연결이 종료될 때까지 반영되지 않습니다. 그러나 로컬 시스템의 사용자 프로파일이 삭제되고, 동일한 사용자 ID를 사용하여 다른 사용자 프로파일이 작성되는 경우에는 지연되지 않습니다. 이 경우 QFileSvr.400 파일 시스템은 사용자가 대상 시스템에 액세스할 수 있는지 확인합니다.

- 오브젝트 권한은 대상 시스템에 상주하는 사용자 프로파일을 기반으로 합니다. 즉, 대상 시스템의 사용자 프로파일이 오브젝트에 대한 적절한 권한을 가지고 있는 경우에만 대상 시스템의 파일 시스템에 있는 오브젝트에 액세스할 수 있습니다.

관련 정보

네트워크 인증 서비스

EIM(Enterprise Identity Mapping)

QFileSvr.400 파일 시스템에서 링크

QFileSvr.400 파일 시스템은 오브젝트에 대해 하나의 링크만을 지원합니다.

기호 링크는 QFileSvr.400에 작성되거나 저장될 수 없습니다. 그러나 QFileSvr.400의 파일은 『루트』(/), QOpenSys 또는 사용자 정의 파일 시스템으로부터 기호 링크를 사용하여 액세스할 수 있습니다.

관련 개념

12 페이지의 『링크』

*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

QFileSvr.400 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

대부분의 통합 파일 시스템 명령 및 화면은 QFileSvr.400 파일 시스템에서 유효합니다.

다음 명령을 제외하고 74 페이지의 『CL 명령을 사용한 액세스』에 나열된 명령은 QFileSvr.400 파일 시스템에서 작동할 수 있습니다.

- ADDLNK
- APYJRNCHG
- CHGAUT
- CHGJRNOBJ
- CHGOWN

- DSPAUT
- DSPJRN
- ENDJRN
- RCLLNK
- RCVJRNE
- RST
- RTVJRNE
- SAV
- SNDJRNE
- STRJRN
- WRKOBJOWN
- WRKOBJPGP

동일한 제한사항이 72 페이지의 『메뉴 및 표시 화면을 사용한 액세스』에 설명된 사용자 화면에 적용됩니다.

QFileSvr.400 파일 시스템에서 통합 파일 시스템 API 사용

대부분의 통합 파일 시스템 API는 QFileSvr.400 파일 시스템에서 유효합니다.

다음 API를 제외하고 122 페이지의 『API를 사용한 조작 수행』에 나열된 API는 QFileSvr.400 파일 시스템에서 작동할 수 있습니다.

- chown()
- fchown()
- givedescriptor()
- link()
- mkfifo()
- QjoEndJournal()
- QjoRetrieveJournalEntries()
- QjoRetrieveJournalInformation()
- QJORJIDI()
- QJOSJRNE
- QjoStartJournal
- Qp0lGetPathFromFileID()
- symlink()
- takedescriptor()

관련 정보

어플리케이션 프로그램 인터페이스(API)

네트워크 파일 시스템(NFS)

네트워크 파일 시스템(NFS)은 리모트 NFS 서버에 저장된 오브젝트 및 데이터에 대한 액세스를 제공합니다.

NFS 서버는 NFS 클라이언트가 동적으로 마운트하는 네트워크 파일 시스템을 내보낼 수 있습니다.

또한 네트워크 파일 시스템을 통해 로컬로 마운트되는 모든 파일 시스템은 리모트 서버에서 마운트되었던 디렉토리나 파일 시스템의 기능, 특성, 제한사항 및 의존성을 갖습니다. 마운트된 파일 시스템에서 작업은 로컬로 수행되지 않습니다. 서버와의 연결을 통해 요구가 전달되고 반드시 서버에 있는 파일 시스템의 유형에 있어서 요구사항 및 제한사항을 준수해야 합니다.

통합 파일 시스템 인터페이스를 통한 NFS 파일 시스템 액세스

NFS는 통합 파일 시스템 인터페이스를 통해 액세스할 수 있습니다. 고려사항 및 제한사항에 유의하십시오.

네트워크 파일 시스템의 특성

NFS를 통해 마운트된 파일 시스템의 특성은 서버로부터 마운트된 파일 시스템의 유형에 따라 달라집니다.

로컬 디렉토리나 파일 시스템으로 표시되는 것에서 수행되는 것처럼 보이는 요구들이 실제로는 NFS 연결을 통해 서버에서 수행되고 있는 것입니다.

이와 같은 클라이언트/서버 관계는 혼동을 줄 수 있습니다. 예를 들어, 클라이언트의 『루트』(/) 디렉토리 분기의 맨 위에 있는 서버로부터 QDLS 파일 시스템을 마운트했다고 가정하십시오. 마운트된 파일 시스템이 로컬 디렉토리의 확장자로 표시된다고 할지라도, 실제로 QDLS 파일 시스템처럼 기능하고 작업을 수행합니다.

NFS를 통해 마운트된 파일 시스템에 대해 이와 같은 관계를 인식하는 것은 요구를 로컬로 처리하고 서버 연결을 통해 처리하는데 있어서 매우 중요합니다. 단지 로컬 레벨에서 올바르게 처리되는 명령이라는 이유로 서버로부터 마운트된 디렉토리에서도 작동된다고 생각할 수는 없습니다. 클라이언트에 마운트된 각각의 디렉토리는 서버 파일 시스템의 특성을 갖습니다.

네트워크 파일 시스템의 서버 및 클라이언트 변화

클라이언트/서버 연결에는 네트워크 파일 시스템(NFS)이 기능하는 방법과 가지게 되는 특성에 영향을 미칠 수 있는 세 가지 주요한 가능성이 있습니다.

가능성은 다음과 같습니다.

- 사용자가 클라이언트에 있는 System i 플랫폼으로부터 파일 시스템을 마운트하는 경우.
- 사용자가 클라이언트에 있는 UNIX 플랫폼으로부터 파일 시스템을 마운트하는 경우.
- 사용자가 System i 플랫폼이나 UNIX 플랫폼이 아닌 시스템으로부터 클라이언트에 파일 시스템을 마운트하는 경우.

첫 번째 시나리오에서는 마운트된 파일 시스템이 System i 플랫폼에서 작동하는 방법과 비슷하게 클라이언트에서 작동합니다. 그러나 네트워크 파일 시스템 및 서비스되는 파일 시스템의 특징을 모두 고려해야 합니다. 예를 들어, 사용자가 서버에서 클라이언트로 QDLS 파일 시스템을 마운트하면 QDLS 파일 시스템의 특성과 제한사항을 가지게 됩니다. 인스턴스의 경우 QDLS 파일 시스템에서 경로명 구성요소는 8자와 3자의 확장자

로 제한됩니다. 그러나 마운트된 파일 시스템의 경우에도 NFS 특성 및 제한사항이 있습니다. 예를 들어, CHGAUD 명령을 사용하여 NFS 오브젝트의 감사 값을 변경할 수 없습니다.

두 번째 시나리오에서는 UNIX 서버에서 마운트된 임의의 파일 시스템이 i5/OS 서버 QOpenSys 파일 시스템과 매우 유사하게 작동한다는 사실을 알아두어야 합니다.

세 번째 시나리오에서는 오퍼레이팅 시스템과 연관된 파일 시스템의 데이터를 검토해야 합니다.

관련 참조

35 페이지의『개방 시스템 파일 시스템(QOpenSys)』

QOpenSys 파일 시스템은 POSIX 및 XPG(X/Open Portability Guide)와 같은 UNIX 기반의 개방 시스템 표준과 호환될 수 있습니다. "루트"(/) 파일 시스템과 마찬가지로 이 파일 시스템은 통합 파일 시스템에서 제공하는 스트림 파일 및 디렉토리 지원을 이용합니다.

네트워크 파일 시스템에서 링크

일반적으로, 동일한 오브젝트에 대해 다수의 하드 링크가 네트워크 파일 시스템에서 허용됩니다.

기호 링크가 전면 지원됩니다. 기호 링크는 네트워크 파일 시스템에서 다른 파일 시스템의 오브젝트와 연결시키는 데 사용될 수 있습니다. 다수의 하드 링크와 기호 링크의 가능성은 전적으로 NFS를 사용하여 마운트되는 파일 시스템에 달려 있습니다.

관련 개념

12 페이지의『링크』

*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

네트워크 파일 시스템에서 통합 파일 시스템 명령 사용

대부분의 통합 파일 시스템 명령은 네트워크 파일 시스템(NFS)에서 유효합니다.

다음 명령을 제외하고 74 페이지의『CL 명령을 사용한 액세스』에 나열된 모든 명령과 72 페이지의『메뉴 및 표시 화면을 사용한 액세스』에 표시된 모든 표시 화면은 네트워크 파일 시스템에서 작동할 수 있습니다.

- APYJRNCHG
- CHGJRNOBJ
- CHGAUD
- CHGATR
- CHGAUT
- CHGOWN
- CHGPGP
- CHKIN
- CHKOUT

- DSPJRN
- ENDJRN
- RCLLNK
- RCVJRNE
- RTVJRNE
- SNDJRNE
- STRJRN

네트워크 파일 시스템과 일반적으로 마운트되어 있는 기타 파일 시스템에 고유한 몇 가지 CL 명령이 있습니다. 그러나 다중 스레드가 가능한 프로세스에서 이러한 명령을 사용하는 것은 안전하지 않을 수도 있습니다. 다음은 이와 같은 명령에 관한 설명입니다.

표 6. 네트워크 파일 시스템 CL 명령

명령	설명
ADDMFS	마운트된 파일 시스템 추가. 로컬 클라이언트 디렉토리로 내보낸 리모트 서버 파일 시스템을 위치시킵니다.
CHGNFSEXP	네트워크 파일 시스템 변경. 내보내기는 네트워크 파일 시스템 클라이언트로 내보낸 파일 시스템의 내보내기 표에 디렉토리 트리를 추가하거나 제거합니다.
DSPMFSINF	마운트된 파일 시스템 정보 표시. 마운트된 파일 시스템에 관한 정보를 표시합니다.
ENDNFSSVR	네트워크 파일 시스템 서버 종료. 서버상의 하나 또는 모든 네트워크 파일 시스템 디먼을 종료합니다.
EXPORTFS	파일 시스템 내보내기. 네트워크 파일 시스템 클라이언트로 내보낸 파일 시스템의 내보내기 표에 디렉토리 트리를 추가하거나 제거합니다.
MOUNT	파일 시스템 마운트. 로컬 클라이언트 디렉토리에 내보낸, 리모트 서버 파일 시스템을 위치시킵니다. 이 명령은 ADDMFS 명령에 대한 별명입니다.
RLSIFSLCK	통합 파일 시스템 잠금 해제. 클라이언트에 의해 보류되거나 오브젝트에 있는 모든 네트워크 파일 시스템 바이트 범위 잠금을 해제합니다.
RMVMFS	마운트된 파일 시스템 제거. 내보낸, 리모트 서버 파일 시스템을 로컬 클라이언트 이름공간에서 제거합니다.
STRNFSSVR	네트워크 파일 시스템 서버 시작. 서버상의 하나 또는 모든 네트워크 파일 시스템 디먼을 시작합니다.
UNMOUNT	파일 시스템 마운트 해제. 내보낸, 리모트 서버 파일 시스템을 로컬 클라이언트 이름공간에서 제거합니다. 이 명령은 RMVMFS 명령에 대한 별명입니다.

주: 네트워크 파일 시스템에서 명령을 사용하기 위해서는 네트워크 파일 시스템을 먼저 마운트시켜야 합니다.

관련 정보


i5/OS 네트워크 파일 시스템 지원 PDF

네트워크 파일 시스템에서 통합 파일 시스템 API 사용

대부분의 통합 파일 시스템 API는 네트워크 파일 시스템(NFS)에서 유효합니다.

다음 API를 제외하고 122 페이지의 『API를 사용한 조작 수행』에 나열된 모든 API는 네트워크 파일 시스템에서 작동할 수 있습니다.

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

특히 네트워크 파일 시스템과 관련된 C 언어 함수에 대한 완벽한 설명은 i5/OS Network File System Support  를 참조하십시오.

주: API를 사용하기 전에 네트워크 파일 시스템이 먼저 마운트되어야 합니다.

관련 정보

어플리케이션 프로그램 인터페이스(API)

통합 파일 시스템에 액세스

시스템의 라이브러리, 오브젝트, 데이터베이스 파일, 폴더 및 문서에 대한 작업에 사용되는 메뉴, 명령, 화면과 같은 모든 사용자 인터페이스는 통합 파일 시스템이 도입되기 전과 마찬가지로 작동됩니다.

그러나 이 인터페이스는 통합 파일 시스템에서 지원되는 스트림 파일, 디렉토리 및 기타 오브젝트에 대한 작업에 사용될 수 없습니다.

통합 파일 시스템에 별도의 사용자 인터페이스 세트가 제공됩니다. 이 인터페이스는 통합 파일 시스템 디렉토리를 통해 액세스할 수 있는 파일 시스템의 오브젝트에서 사용될 수 있습니다.

메뉴와 표시장치 또는 제어 언어(CL) 명령을 사용하여 시스템에서 통합 파일 시스템의 디렉토리 및 오브젝트와 대화할 수 있습니다. 또한 API(Application Programming Interface)를 사용하여 통합 파일 시스템의 스트림 파일, 디렉토리 및 기타 지원을 활용할 수 있습니다.

Windows 데스크탑에서 서버를 관리하는 데 사용되는 System i Navigator, 그래픽 인터페이스를 통해 통합 파일 시스템과 대화할 수도 있습니다.

메뉴 및 표시 화면을 사용한 액세스

시스템에서 제공하는 메뉴 및 표시장치 세트를 사용하여 통합 파일 시스템의 파일 및 기타 오브젝트에 대한 조작을 수행할 수 있습니다.

통합 파일 시스템 메뉴를 표시하려면 다음을 수행하십시오.

1. 사용자 시스템에 사인 온하십시오.
2. 계속하려면 Enter를 누르십시오.
3. 기본 메뉴에서 **파일, 라이브러리 및 폴더** 옵션을 선택하십시오.
4. 파일, 라이브러리 및 폴더 메뉴에서 **통합 파일 시스템** 옵션을 선택하십시오.

이 때 필요에 따라 통합 파일 시스템에서 디렉토리 명령, 오브젝트 명령 또는 보안 명령에 대해 작업할 수 있습니다. 그러나 사용하게 될 CL 명령을 알고 있는 경우 옵션 메뉴를 바이패스하고 화면의 맨 아래 명령행에 해당 명령을 입력하고 **Enter**를 누를 수 있습니다.

또한 다음 단계를 수행하여 시스템에 있는 임의의 메뉴에서 통합 파일 시스템에 액세스할 수 있습니다.

1. 명령행에 GO DATA를 입력하여 파일, 라이브러리 및 폴더 메뉴를 표시하십시오.
2. **통합 파일 시스템**을 선택하십시오.

네트워크 파일 시스템 명령 메뉴를 보려면 GO CMDNFS를 입력하십시오. 사용자 정의 파일 시스템 명령 메뉴를 보려면 명령행에 GO CMDUDFS를 입력하십시오.

통합 파일 시스템 메뉴에서 다음 조작을 수행할 수 있는 화면 또는 명령을 요청할 수 있습니다.

- 디렉토리 작성, 변환 및 제거
- 현재 디렉토리명 표시 및 변경
- 오브젝트 링크 추가, 표시, 변경 및 제거
- 오브젝트 복사, 이동 및 재명명
- 오브젝트 체크 인 및 체크 아웃
- 오브젝트 저장(백업) 및 복원
- 오브젝트 소유자 및 사용자 권한 표시 및 변경
- 오브젝트 속성 표시 및 변경
- 스트림 파일 및 데이터베이스 파일 멤버간 데이터 복사
- 사용자 정의 파일 시스템 상태의 작성, 삭제, 표시
- 서버로부터의 파일 시스템 내보내기
- 클라이언트에서 파일 시스템 마운트 및 마운트 해제

어떤 파일 시스템은 이와 같은 모든 조작을 지원하지 않습니다.

관련 개념

26 페이지의『파일 시스템』

*파일 시스템*은 논리 장치(LU)를 구성하는 요소인 기억장치의 특정 세그먼트에 액세스할 수 있도록 지원합니다. 시스템에서 논리 장치는 파일, 디렉토리, 라이브러리 및 오브젝트입니다.

관련 참조

77 페이지의 『CL 명령 및 표시장치에 대한 경로명 규칙』
통합 파일 시스템 명령이나 화면을 사용하여 오브젝트에 대해 조작할 때 경로명을 제공하여 오브젝트를 식별합니다.

『CL 명령을 사용한 액세스』
통합 파일 시스템 메뉴 및 표시 화면을 통해 수행할 수 있는 모든 조작은 제어 언어(CL) 명령을 입력하여 수행될 수 있습니다. 이러한 명령은 통합 파일 시스템 인터페이스를 통해 액세스할 수 있는 파일 시스템의 파일 및 디타 오브젝트에 대해 조작할 수 있습니다.

CL 명령을 사용한 액세스

통합 파일 시스템 메뉴 및 표시 화면을 통해 수행할 수 있는 모든 조작은 제어 언어(CL) 명령을 입력하여 수행될 수 있습니다. 이러한 명령은 통합 파일 시스템 인터페이스를 통해 액세스할 수 있는 파일 시스템의 파일 및 디타 오브젝트에 대해 조작할 수 있습니다.

표 1은 통합 파일 시스템 명령을 요약한 것입니다. 특히 사용자 정의 파일 시스템, 네트워크 파일 시스템 및 일반적으로 마운트된 파일 시스템과 관련되는 CL 명령에 대한 자세한 정보는 37 페이지의 『사용자 정의 파일 시스템(UDFS)』 및 69 페이지의 『네트워크 파일 시스템(NFS)』을 참조하십시오. 하나의 명령어가 OS/2 또는 DOS 명령으로서 동일한 조작을 수행하는 경우 OS/2 및 DOS 사용자의 편의를 위해 별명(대체 명령어)이 제공됩니다.

표 7. 통합 파일 시스템 명령

명령	설명	별명
ADDLNK[3]	링크 추가. 디렉토리와 오브젝트간의 링크를 추가합니다.	
ADDMFS[3]	마운트된 파일 시스템 추가. 로컬 클라이언트 디렉토리로 내보낸 리모트 서버 파일 시스템을 위치시킵니다.	MOUNT
APYJRNCHG[2] [3]	저널링된 변경사항 적용. 저널 항목을 사용하여 저널링된 오브젝트가 저장된 이후로 발생한 변경사항을 적용하거나 지정된 시점까지의 변경사항을 적용합니다.	
CHGATR[3]	속성 변경. 단일 오브젝트, 오브젝트 그룹 또는 디렉토리, 디렉토리 목차, 모든 서브디렉토리의 목차에 변경된 속성이 있는 디렉토리의 속성을 변경합니다.	
CHGAUD[3]	감사 값 변경. 오브젝트에 대한 감사를 시작하거나 중지합니다.	
CHGAUT[3]	권한 변경. 사용자 또는 사용자 그룹에게 오브젝트에 대한 특정 권한을 부여합니다.	
CHGCURDIR[3]	현재 디렉토리 변경. 현재 디렉토리로 사용할 디렉토리를 변경합니다.	CD, CHDIR
CHGJRNOBJ[2] [3]	저널링된 오브젝트 변경. 오브젝트에 대한 저널링을 종료하고 재시작할 필요없이 오브젝트의 저널링 속성 또는 오브젝트 목록을 변경합니다.	
CHGNFSEXP	네트워크 파일 시스템 내보내기 변경. NFS 클라이언트로 내보낸된 내보내기 표로 디렉토리 트리를 추가하거나 제거합니다.	EXPORTFS
CHGOWN[3]	소유자 변경. 다른 사용자에게로 오브젝트 소유권을 이전합니다.	
CHGPGP[3]	1차 그룹 변경. 다른 사용자로 1차 그룹을 변경합니다.	
CHKIN[3]	체크 인. 이전에 체크 아웃된 오브젝트를 체크 인합니다.	
CHKOBJITG[3]	오브젝트 무결성 검사. 오브젝트의 모든 무결성 위반을 검사합니다.	
CHKOUT[3]	체크 아웃. 다른 사용자가 변경하지 못하도록 오브젝트를 체크 아웃합니다.	

표 7. 통합 파일 시스템 명령 (계속)

명령	설명	별명
CPY[3]	복사. 단일 오브젝트 또는 오브젝트 그룹을 복사합니다.	COPY
CPYFRMSTMF[3]	스트림 파일로부터 복사. 스트림 파일로부터 데이터베이스 파일 멤버로 데이터를 복사합니다.	
CPYTOSTMF[3]	스트림 파일로 복사. 데이터베이스 파일 멤버로부터 스트림 파일로 데이터를 복사합니다.	
CRTDIR[3]	디렉토리 작성. 새로운 디렉토리를 시스템에 추가합니다.	MD, MKDIR
CRTUDFS[3]	UDFS 작성. 사용자 정의 파일 시스템을 작성합니다.	
CVTDIR	디렉토리 변환. 통합 파일 시스템 디렉토리를 *TYPE1 형식에서 *TYPE2 형식으로 변환하는 데 필요한 정보를 제공합니다.	
CVTRPCSRC	RPC 소스 변환. 리모트 프로시듀어 호출(RPC) 언어로 작성된 입력 파일로부터 C 코드를 생성합니다.	RPCGEN
DLTUDFS[3]	UDFS 삭제. 사용자 정의 파일을 삭제합니다.	
DSPAUT	권한 표시. 오브젝트에 대하여 권한이 있는 사용자 리스트 및 그 오브젝트에 대한 권한을 보여줍니다.	
DSPCURDIR	현재 디렉토리 표시. 현재 디렉토리명을 표시합니다.	
DSPJRN[2] [3]	저널 표시. 저널 항목(하나 이상의 리시버에 포함된)을 외부 표시에 적합한 형태로 변환합니다.	
DSPLNK	오브젝트 링크 표시. 디렉토리 내의 오브젝트 리스트를 표시하고 오브젝트에 대한 정보를 표시하는 옵션을 제공합니다.	
DSPF	스트림 파일 표시. 스트림 파일 또는 데이터베이스 파일을 표시합니다.	
DSPMFSINF	마운트된 파일 시스템 정보 표시. 마운트된 파일 시스템에 관한 정보를 표시합니다.	STATFS
DSPUDFS	UDFS 표시. 사용자 정의 파일 시스템을 표시합니다.	
EDTF	스트림 파일 편집. 스트림 파일 또는 데이터베이스 파일을 편집합니다.	
ENDJRN[2] [3]	저널 종료. 오브젝트 또는 오브젝트 리스트에 변경사항 저널링을 종료합니다.	
ENDNFSSVR	네트워크 파일 시스템 서버 종료. 서버와 클라이언트상의 하나 또는 모든 NFS 디몬을 종료합니다.	
ENDRPCBIND	RPC 바인더 디몬 종료. 리모트 프로시듀어 호출(RPC) RPCBind 디몬을 종료합니다.	
MOV[3]	이동. 오브젝트를 다른 디렉토리로 이동합니다.	MOVE
PRTDIRINF	디렉토리 정보 인쇄. RTVDIRINF(디렉토리 정보 검색) 명령에 의해 수집된 통합 파일 시스템에서 오브젝트에 대한 디렉토리 정보를 인쇄하기 위해 사용됩니다.	
RCLLNK[3]	오브젝트 링크 재생. 사용 중인 마운트된 파일 시스템에서 문제점을 식별하고 가능한 경우 문제점을 정정합니다.	
RCVJRNE[2] [3]	저널 항목 수신. 지정된 사용자 종료 프로그램이 지속적으로 저널 항목을 수신할 수 있게 합니다.	
RLSIFSLCK[3]	통합 파일 시스템 잠금 해제. NFS 클라이언트에 의해 보류되거나 오브젝트에 있는 모든 바이트 범위 잠금을 해제합니다.	
RMVDIR[3]	디렉토리 제거. 시스템에서 디렉토리를 제거합니다.	RD, RMDIR
RMVLNK[3]	링크 제거. 오브젝트에 대한 링크를 제거합니다.	DEL, ERASE
RMVMFS[3]	마운트된 파일 시스템 제거. 내보낸, 리모트 서버 파일 시스템을 로컬 클라이언트 디렉토리에서 제거합니다.	UNMOUNT

표 7. 통합 파일 시스템 명령 (계속)

명령	설명	별명
RNM[3]	이름 변경. 디렉토리에서 오브젝트명을 변경합니다.	REN
RPCBIND	RPC 바인더 디몬 시작. 리모트 프로시듀어 호출(RPC) RPCBind 디몬을 시작합니다.	
RST[3]	복원. 오브젝트나 오브젝트 그룹을 백업 장치에서 시스템으로 복사합니다.	
RTVCURDIR	현재 디렉토리 검색. 현재 디렉토리명을 검색하고 이를 지정된 변수(CL 프로그램에서 사용됨)에 넣습니다.	
RTVDIRINF	디렉토리 정보 검색. 통합 파일 시스템에서 오브젝트 속성을 수집합니다.	
RTVJRNE[2] [3]	저널 항목 검색. 특정 저널 항목을 얻고 CL 변수에 결과를 위치시킵니다.	
SAV[3]	저장. 오브젝트나 오브젝트 그룹을 시스템에서 백업 장치로 복사합니다.	
SNDJRNE[2] [3]	저널 항목 송신. 선택적으로 저널링된 오브젝트와 연관된 사용자 저널 항목을 저널 리시버에 추가합니다.	
STRJRN[2] [3]	저널 시작. 특정 저널에 대한 변경사항(오브젝트나 오브젝트 리스트에 대해 이루어진) 저널링을 시작합니다.	
STRNFSSVR	네트워크 파일 시스템 서버 시작. 서버와 클라이언트상의 하나 또는 모든 NFS 디먼을 시작합니다.	
WRKAUT	권한에 대한 작업. 사용자 리스트 및 사용자 권한을 표시하고, 사용자 추가, 사용자 권한 변경 또는 사용자 제거의 옵션을 제공합니다.	
WRKLNK	오브젝트 링크에 대한 작업. 디렉토리 내의 오브젝트 리스트를 표시하고 오브젝트에 대하여 조치를 수행할 수 있는 옵션을 제공합니다.	
WRKOBJOWN[1]	소유자에 의한 오브젝트에 대한 작업. 사용자 프로파일의 오브젝트 리스트를 표시하고 오브젝트에 대하여 조치를 수행할 수 있는 옵션을 제공합니다.	
WRKOBJPGP[1]	1차 그룹에 의한 오브젝트에 대한 작업. 1차 그룹에서 제어하는 오브젝트 리스트를 표시하고 오브젝트에 대하여 조치를 수행할 수 있는 옵션을 제공합니다.	

주:

1. WRKOBJOWN 및 WRKOBJPGP 명령은 모든 오브젝트 유형을 표시할 수 있지만 모든 파일 시스템에서 완전하게 기능하지 않을 수도 있습니다.
2. 자세한 정보는 i5/OS Information Center의 저널 관리를 참조하십시오.
3. 다음 명령은 유니코드 사용 가능 명령입니다. 자세한 정보는 i5/OS Information Center의 제어 언어(CL)에서 유니코드 지원을 참조하십시오.

관련 개념

26 페이지의『파일 시스템』

*파일 시스템*은 논리 장치(LU)를 구성하는 요소인 기억장치의 특정 세그먼트에 액세스할 수 있도록 지원합니다. 시스템에서 논리 장치는 파일, 디렉토리, 라이브러리 및 오브젝트입니다.

관련 태스크

72 페이지의『메뉴 및 표시 화면을 사용한 액세스』

시스템에서 제공하는 메뉴 및 표시장치 세트를 사용하여 통합 파일 시스템의 파일 및 기타 오브젝트에 대한 조작을 수행할 수 있습니다.

관련 정보

제어 언어(CL)

CL 명령 및 표시장치에 대한 경로명 규칙

통합 파일 시스템 명령이나 화면을 사용하여 오브젝트에 대해 조작할 때 경로명을 제공하여 오브젝트를 식별합니다.

다음 리스트는 경로명을 지정할 때 유의해야 할 규칙을 요약한 것입니다. 이 규칙에서 *오브젝트*라는 용어는 디렉토리, 파일, 링크 또는 기타 오브젝트를 나타냅니다.

- 오브젝트명은 각 디렉토리 내에서 고유해야 합니다.
- 통합 파일 시스템 CL 명령에 전달된 경로명은 작업에 대해 코드화 문자 세트(CCSID)로 표시되어야 합니다. 작업의 CCSID가 65535인 경우, 경로명이 반드시 그 작업의 디폴트 CCSID로 표시되어야 합니다. 텍스트 스트링은 일반적으로 CCSID 37로 코드화되기 때문에, 경로를 명령에 전달하기 전에 하드 코드화된 경로명을 작업 CCSID로 변환해야 합니다.

  주: 유니코드 사용 가능 명령에서 유니코드 사용 가능 지원을 활용하여 명령을 호출하면 이러한 제약사항은 없습니다. 예를 들어, 유니코드 CCSID된 명령 및 경로명 정보와 함께 QCAPCMD API를 호출할 수 있습니다. 자세한 정보는 i5/OS Information Center의 제어 언어(CL)에서 유니코드 지원을 참조하십시오.

- 경로명은 명령행에 입력 시 작은 따옴표(')로 표시해야 합니다. 이 표시는 화면에 경로명을 입력할 때 선택사항입니다. 그러나 경로명에 인용된 스트링이 포함되어 있는 경우 ' ' 표시도 포함되어야 합니다.
- 경로명은 왼쪽에서 오른쪽으로 입력되고 최고 레벨 디렉토리에서 시작하여 명령을 통해 조작될 오브젝트명으로 끝납니다. 경로의 각 구성요소명은 슬래시(/)로 구분됩니다.

  주: 또한 일부 CL 명령은 백슬래시(₩)를 슬래시(/)로 자동 변환하여 분리자로 백슬래시(₩)를 사용할 수 있게 합니다. 그러나 일부 다른 CL 명령은 백슬래시(₩)를 다른 문자와 다르지 않게 취급합니다. 따라서 백슬래시(₩) 분리자 사용 시 주의해야 합니다.

  예를 들면, 다음과 같습니다.

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

  또는

```
'Dir1₩Dir2₩Dir3₩UsrFile'
```

- 슬래시(/) 및 백슬래시(₩)가 분리자로 사용될 때 슬래시(/) 및 백슬래시(₩) 문자와 널(null)은 경로명의 개별 구성요소에 사용될 수 없습니다. 명령어에 의해 소문자가 대문자로 변경되지 않습니다. 오브젝트가 들어 있는 파일 시스템이 대소문자를 구분하는지와 오브젝트가 작성 또는 검색 중인지에 따라 이름이 대문자로 변경될 수도 있고 그렇지 않을 수도 있습니다.
- 오브젝트명의 길이는 오브젝트가 들어 있는 파일 시스템과 명령 스트링의 최대 길이에 의해 제한됩니다. 명령어는 최대 255자의 오브젝트명을 허용하고 5000자까지의 경로명을 허용합니다.

- 경로명 시작 부분의 분리 문자(예: /)는 경로가 최상위 디렉토리(『루트』(/) 디렉토리)에서 시작함을 의미합니다. 다음은 그 예입니다.

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

- 경로명이 분리 문자(예: /)로 시작하지 않는 경우 경로는 명령을 입력하는 사용자의 현재 디렉토리에서 시작한다고 간주됩니다. 다음은 그 예입니다.

```
'MyDir/MyFile'
```

  여기서 MyDir은 사용자의 현재 디렉토리의 서브디렉토리입니다.

- 경로명의 처음 부분에 분리자(예: /)가 뒤에 오는 틸드(~) 문자는 명령을 입력하는 사용자의 홈 디렉토리에서 경로가 시작함을 의미합니다. 예를 들면, 다음과 같습니다.

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

- 경로명의 처음 부분에 사용자명 다음에 분리자(예: /)가 뒤에 오는 틸드(~) 문자는 사용자명으로 식별되는 사용자의 홈 디렉토리에서 경로가 시작함을 의미합니다. 예를 들면, 다음과 같습니다.

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

- 일부 명령에서 별표(*) 또는 물음표(?)는 이름 유형을 탐색하기 위한 경로명의 최종 구성요소에서 사용될 수 있습니다. *는 * 문자 위치에 어떠한 문자라도 올 수 있으며 그 수도 지정되지 않았음을 의미합니다. ?는 ? 문자 위치에 단일 문자가 올 수 있음을 의미합니다. 다음의 예는 그 이름이 *d*로 시작하고 *txt*로 끝나는 모든 오브젝트를 탐색합니다.

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

  다음의 예는 그 이름이 *d*로 시작하고 뒤에 단일 문자가 오고 *txt*로 끝나는 모든 오브젝트를 탐색합니다.

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

- i5/OS 특수 값과의 혼돈을 피하기 위해 경로명은 단 하나의 별표(*) 문자로 시작할 수 없습니다. 패턴의 일치를 위해 경로명을 시작할 때 다음과 같이 2개의 별표(*)를 사용하십시오.

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

  주: 이것은 별표(*) 앞에 기타 다른 문자가 없는 상대 경로명에만 적용됩니다.

- QSYS.LIB 파일 시스템의 오브젝트에 대해 조작할 때 구성요소명은 *name.object-type*의 형태여야 하며 예는 다음과 같습니다.

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

- 독립 ASP QSYS.LIB 파일 시스템의 오브젝트에 대해 조작할 때 구성요소명은 *name.object-type*의 형태여야 하며 예는 다음과 같습니다.

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

- 다음 문자가 구성요소명에 사용되는 경우 경로명을 추가 작은 따옴표(') 또는 인용 부호(") 세트 안에 넣어야 합니다.
  - 별표(*)

    주: i5/OS 특수 값과의 혼동을 피하기 위해 경로명은 단일 별표(*) 문자로 시작되어서는 안됩니다.

- 의문 부호(?)
- 작은 따옴표(')
- 인용 부호(")
- 틸드(~) 문자, 경로명에서 첫 번째 구성요소명의 첫 번째 문자로 사용되는 경우(다른 위치에 사용될 경우, 틸드는 단지 또다른 문자로 해석됨)

예를 들면, 다음과 같습니다.

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

또는

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

명령 스트링의 문자 의미가 혼동될 수 있고 명령 스트링이 잘못 입력될 수 있으므로 이 경우는 바람직하지 않습니다.

- 경로명에 콜론(:)을 사용하지 마십시오. 이는 시스템 내에서 특별한 의미를 가집니다.
- 명령 및 연관된 사용자 화면에 대한 처리 지원으로 인해, 16진 40 이하의 코드점은 명령 스트링이나 화면상에서 사용될 수 있는 문자로 인식되지 않습니다. 코드점이 사용되는 경우 다음 예와 같이 16진 표시로 입력되어야 합니다.

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

따라서, 경로명에서 16진 40이하의 코드점을 사용하지 않는 것이 좋습니다. 이 제한사항은 API가 아니라 명령 및 연관된 화면에만 적용됩니다. 또한 16진 0값은 경로명으로 허용되지 않습니다.

관련 개념

26 페이지의『파일 시스템』

*파일 시스템*은 논리 장치(LU)를 구성하는 요소인 기억장치의 특정 세그먼트에 액세스할 수 있도록 지원합니다. 시스템에서 논리 장치는 파일, 디렉토리, 라이브러리 및 오브젝트입니다.

관련 정보

제어 언어(CL)

RTVDIRINF 및 PRTDIRINF 명령의 출력에 대한 작업

RTVDIRINF(디렉토리 정보 검색) 명령은 통합 파일 시스템에서 오브젝트의 속성을 수집하기 위해 사용됩니다. 수집된 정보는 INFFILEPFX 매개변수에 의해 지정되는 정보 파일 접두부를 사용하여 명명되는 데이터베이스 파일(표)에 저장됩니다. 표는 INFLIB 매개변수에 의해 지정되는 라이브러리에 작성됩니다.

세 개의 표는 RTVDIRINF 명령의 결과로 작성됩니다. 한 개의 표는 오브젝트 속성을 저장하며, 다른 한 개의 표는 디렉토리를 위해, 나머지 표는 오브젝트 속성을 저장하는 데 사용되는 파일을 결정하기 위해 사용됩니다.

| V6R1부터 System i Navigator로 이 세 테이블을 생성할 수 있습니다. 더 자세한 정보는 92 페이지의『System
| i Navigator를 사용하여 폴더 속성 수집 및 검색하기』를 참조하십시오.

다음 표에서는 오브젝트 속성을 저장하는 표에 제공되는 필드에 대해 설명합니다. *GEN이 정보 파일 접두부(INFFILEPFX) 매개변수에 지정되는 경우 데이터베이스 파일은 이 명령으로 생성되는 고유한 접두부와 함께 작성됩니다. 접두부는 QAEZD 다음에 네 자릿수로 시작됩니다. 수집된 정보를 저장하기 위해 작성되는 파일은 이 접두부 다음에 문자 D(디렉토리 정보를 포함하는 파일의 경우) 또는 문자 O(디렉토리의 오브젝트에 대한 정보를 포함하는 파일의 경우)를 사용하여 명명됩니다. 예를 들어, 지정된 *GEN과 함께 명령이 처음으로 실행되면 파일 QAEZD0001D 및 QAEZD0001O은 정보 라이브러리(INFLIB) 매개변수로 지정되는 라이브러리에 작성됩니다. 사용자는 이 데이터베이스 명명에 사용하기 위해 파일 접두부를 지정할 수 있으며 이는 최대 9문자 길이일 수 있습니다.

표 8. QAEZDxxxxO(오브젝트 속성 저장)

필드명	필드 유형	필드 설명
QEZDIRIDX	INTEGER	디렉토리 및 오브젝트 테이블 간의 관계 ID입니다. 이 테이블을 조인(Join)하여 완전한 경로명을 얻을 수 있습니다. 오브젝트의 상위 디렉토리를 얻기 위해 이 오브젝트 테이블의 QEZDIRIDX 필드 값과 일치하는 디렉토리 테이블의 QEZDIRIDX 값을 찾습니다. **주:** ″루트″ 디렉토리(/)가 RTVDIRINF의 입력으로 지정되면 ″루트″ 디렉토리(/)가 상위 디렉토리를 가지고 있지 않지만 QEZDIRIDX 값은 1이 됩니다.
QEZOBJNAM[1]	VARGRAPHIC (1024)	오브젝트명[2]
QEZOBJLEN	INTEGER	오브젝트명(QEZOBJNAM 필드)에 포함된 바이트 수
QEZNMCCSID	INTEGER	오브젝트명(QEZOBJNAM 필드)이 표시되는 CCSID(코드화 문자 세트 ID)
QEZREGION	GRAPHIC (2)	오브젝트명(QEZOBJNAM 필드)의 국가를 나타내는 두 문자 ID. 이 ID는 배열 순서와 같이 조치의 위치에 따라 정의되는 조치에 영향을 줍니다.
QEZLANGID	GRAPHIC (3)	오브젝트명(QEZOBJNAM 필드)이 있는 언어를 나타내는 세 문자 ID
QEZMODE	INTEGER	파일 액세스 모드 및 유형. 모드에 대한 자세한 정보는 열린 파일 open() API를 참조하십시오.
QEZOBJTYPE[1]	GRAPHIC (10)	오브젝트 유형
QEZCCSID	INTEGER	데이터의 CCSID 및 오브젝트의 확장 속성
QEZALCSIZE[1]	BIGINT	이 오브젝트에 할당된 바이트 수
QEZDTASIZE	BIGINT	이 오브젝트에 있는 데이터의 바이트 단위 *크기*. 이 *크기*는 오브젝트 헤더 및 오브젝트와 연관된 확장 속성의 *크기*를 포함하지 않습니다.
QEZEAS	BIGINT	이 오브젝트와 연관된 확장 속성 수
QEZCEAS	BIGINT	이 오브젝트와 연관된 중요한 확장 속성 수
QEZEXTATRS	BIGINT	모든 확장 속성 데이터에 대한 총 바이트 수
QEZCRTTIM	TIMESTAMP	오브젝트가 작성된 날짜 및 시간
QEZACCTIM	TIMESTAMP	오브젝트 데이터에 마지막으로 액세스한 날짜 및 시간
QEZCHGTIMA[1]	TIMESTAMP	오브젝트 속성이 마지막으로 변경된 날짜 및 시간
QEZCHGTIMD	TIMESTAMP	오브젝트 데이터가 마지막으로 변경된 날짜 및 시간
QEZSTGFREE[1]	SMALLINT	오브젝트의 데이터가 오프라인으로 이동되었거나 그 온라인 기억장치 해제. 유효한 값은 다음과 같습니다. 0 - 오브젝트의 데이터가 오프라인이 아닙니다. 1 - 오브젝트의 데이터가 오프라인입니다.

표 8. QAEZDxxxxO(오브젝트 속성 저장) (계속)

필드명	필드 유형	필드 설명
QEZCHKOUT[1]	SMALLINT	오브젝트가 체크 아웃되는지에 관한 인디케이터. 유효한 값은 다음과 같습니다. 0 - 오브젝트가 체크 아웃이 아닙니다. 1 - 오브젝트가 체크 아웃입니다.
QEZCHKOWN	GRAPHIC (10)	오브젝트를 가진 사용자가 체크 아웃했습니다. 오브젝트가 체크 아웃이 아니면 이 필드는 공백입니다.
QEZCHKTIM	TIMESTAMP	오브젝트가 체크 아웃된 날짜 및 시간. 오브젝트가 체크 아웃되지 않으면 이 필드의 값으로 널(null)이 포함됩니다.
QEZLOCAL	SMALLINT	오브젝트가 로컬로 저장되거나 리모트 시스템에 저장됩니다. 오브젝트가 로컬 또는 리모트인지의 결정은 각각의 파일 시스템 규칙에 따라 다릅니다. 로컬 또는 리모트 인디케이터를 수반하지 않는 파일 시스템의 오브젝트는 리모트로 처리합니다. 유효한 값은 다음과 같습니다. 1 - 오브젝트의 데이터가 로컬로 저장됩니다. 2 - 오브젝트의 데이터가 리모트 시스템에 있습니다.
QEZOWN[1]	GRAPHIC (10)	오브젝트 또는 다음 특수 값의 소유자인 사용자 프로파일의 이름. *NOUSRPRF - 이 특수 값은 리모트 오브젝트의 UID와 일치하는 사용자 ID(UID)를 가진 로컬 iSeries™ 서버에 사용자 프로파일이 없는 것을 나타내기 위해 네트워크 파일 시스템이 사용합니다.
QEZUID	INTEGER	시스템에 있는 각 사용자는 고유한 숫자 사용자 ID 번호(UID)를 가져야 합니다.
QEZOWNPGP	GRAPHIC (10)	오브젝트 또는 다음 특수 값의 1차 그룹인 사용자 프로파일의 이름. *NONE - 오브젝트에 1차 그룹이 없습니다. *NOUSRPRF - 이 특수 값은 리모트 오브젝트의 GID와 일치하는 그룹 ID(GID)를 가진 로컬 서버에 사용자 프로파일이 없는 것을 나타내기 위해 네트워크 파일 시스템이 사용합니다.
QEZGID	INTEGER	그룹 프로파일은 고유 숫자 그룹 식별 번호(GID)로 식별됩니다.
QEZAUTLST	GRAPHIC (10)	명명된 오브젝트를 안전하게 하기 위해 사용되는 권한 부여 리스트의 이름. 값 *NONE은 권한 부여 리스트가 오브젝트의 권한을 판별할 때 사용되지 않음을 나타냅니다.
QEZASP	SMALLINT	오브젝트가 저장되는 보조 기억장치 풀
QEZJRNSTS[1]	SMALLINT	오브젝트의 현재 저널링 상태. 이 필드는 다음 값 중 하나일 수 있습니다. 0 (NOT_JOURNALED) - 오브젝트가 현재 저널 중이 아닙니다. 1 (JOURNALED) - 오브젝트가 현재 저널 중입니다.
QEZJSUBTRE	SMALLINT	이 플래그가 리턴될 때 이 오브젝트는 통합 파일 시스템 저널링 서브트리 의미론을 가진 디렉토리입니다. 0 - 오브젝트가 서브트리 의미론으로 저널되지 않습니다. 1 - 오브젝트가 서브트리 의미론으로 저널됩니다. 이 디렉토리의 서브트리에 작성되는 새로운 오브젝트는 이 디렉토리에서 저널링 속성 및 옵션을 계승합니다.

표 8. QAEZDxxxxO(오브젝트 속성 저장) (계속)

필드명	필드 유형	필드 설명
QEZJOPTENT	SMALLINT	저널링이 활성이면 선택적으로 고려되는 항목이 저널됩니다. 선택적 저널 항목의 리스트는 각 오브젝트 유형에 따라 다릅니다. 0 - 오브젝트가 선택적 항목으로 저널되지 않습니다. 1 - 오브젝트가 선택적 항목으로 저널됩니다.
QEZJAFTERI	SMALLINT	저널링이 활성이면 변경 후 오브젝트의 이미지가 저널됩니다. 0 - 오브젝트가 변경 후 이미지로 저널되지 않습니다. 1 - 오브젝트가 변경 후 이미지로 저널됩니다.
QEZJBEFORI	SMALLINT	저널링이 활성이면 변경 전 오브젝트의 이미지가 저널됩니다. 0 - 오브젝트가 변경 전 이미지로 저널되지 않습니다. 1 - 오브젝트가 변경 전 이미지로 저널됩니다.
QEZJRNID	GRAPHIC (10)	이 필드는 여러 저널링 관련 명령 및 API에서 사용될 수 있는 ID와 저널되고 있는 오브젝트를 연관시킵니다. 오브젝트가 저널링되지 않은 경우 이 필드는 공백입니다.
QEZJRNNAM	GRAPHIC (10)	저널링 상태의 값이 JOURNALED이면 이 필드는 현재 사용 중인 저널의 이름을 포함합니다. 저널링 상태의 값이 NOT_JOURNALED이면 이 필드는 이 오브젝트에 마지막으로 사용되는 저널의 이름을 포함합니다. 이 오브젝트를 저널링한 적이 없으면 이 필드의 모든 바이트는 이진 0으로 설정됩니다. 오브젝트가 저널링되지 않은 경우 이 필드는 공백입니다.
QEZJRNLIB	GRAPHIC (10)	저널링 상태의 값이 JOURNALED이면 이 필드는 현재 사용되는 저널을 포함하는 라이브러리의 이름을 포함합니다. 저널링 상태의 값이 NOT_JOURNALED이면 이 필드는 마지막으로 사용되는 저널을 포함하는 라이브러리의 이름을 포함합니다. 이 오브젝트를 저널링한 적이 없으면 이 필드의 모든 바이트는 이진 0으로 설정됩니다. 오브젝트가 저널링되지 않은 경우 이 필드는 공백입니다.
QEZJRNSTR	TIMESTAMP	오브젝트를 위해 시작되는 저널링이 있는 오브젝트에 대한 마지막 날짜 및 시간에 해당하는 에포크 이후 초 수. 이 오브젝트를 저널링한 적이 없으면 이 필드의 이 필드는 이진 0으로 설정됩니다. 오브젝트가 저널링되지 않은 경우 이 필드의 값은 널(null)입니다.
QEZAUDT	GRAPHIC (10)	오브젝트와 연관된 감사 값. 유효한 값은 다음과 같습니다. *NONE - 오브젝트에 액세스하는 사용자에 관계없이 오브젝트를 읽거나 변경할 때 이 오브젝트에 대한 감사가 발생하지 않습니다. *USRPRF - 현재 사용자가 감사되는 경우에만 이 오브젝트를 감사합니다. 이 오브젝트에 대해 감사를 완료해야 하는지 판별하기 위해 현재 사용자가 테스트됩니다. 이 오브젝트에 대해 변경 액세스만 감사되거나 읽기 및 변경 액세스 모두가 감사되는 경우 사용자 프로파일을 지정할 수 있습니다. *CHANGE - 시스템에 있는 모든 사용자가 이 오브젝트에 대한 모든 변경 액세스를 감사합니다. *ALL - 시스템에 있는 모든 사용자가 이 오브젝트에 대한 모든 액세스를 감사합니다. 모든 액세스가 읽기 또는 변경 조작으로 정의됩니다. *NOTAVL - 이 조작을 수행하는 사용자는 현재 오브젝트 감사 값을 검색할 수 없습니다.
QEZBLKSIZ	INTEGER	오브젝트의 블록 크기

표 8. *QAEZDxxxxO(오브젝트 속성 저장) (계속)*

필드명	필드 유형	필드 설명
QEZNLNK	INTEGER	오브젝트에 대한 하드 링크 수
QEZFILEID[1]	GRAPHIC (16)	오브젝트의 파일 ID. 오브젝트와 연관된 ID. 파일 ID는 오브젝트의 경로명을 검색하기 위해 Qp0lGetPathFromFileID() 와 함께 사용될 수 있습니다.
QEZFILEIDS	INTEGER	파일의 4바이트 파일 ID. 이 번호는 파일 시스템 내에서 오브젝트를 고유하게 식별합니다. 이 번호는 전체 시스템에서 오브젝트를 식별할 수 없습니다.
QEZGENID	BIGINT	파일 ID와 연관된 생성 ID
QEZFSID	BIGINT	오브젝트가 속하고 있는 파일 시스템 ID. 이 번호는 오브젝트가 속하고 있는 파일 시스템을 고유하게 식별합니다.
QEZRDEV	BIGINT	오브젝트가 특수 장치 파일을 나타내는 경우 실제 장치는 그것을 나타냅니다.
QEZDOM	GRAPHIC (10)	오브젝트의 도메인. 유효한 값은 다음과 같습니다. *SYSTEM - 오브젝트가 시스템 도메인에 있습니다. *USER - 오브젝트가 사용자 도메인에 있습니다.
QEZCRTAUD	GRAPHIC (10)	이 디렉토리에 작성되는 오브젝트와 연관된 감사 값. 유효한 값은 다음과 같습니다. *NONE - 오브젝트에 액세스하는 사용자에 관계없이 오브젝트를 읽거나 변경할 때 이 오브젝트에 대한 감사가 발생하지 않습니다. *USRPRF - 현재 사용자가 감사되는 경우에만 이 오브젝트를 감사합니다. 이 오브젝트에 대해 감사를 완료해야 하는지 판별하기 위해 현재 사용자가 테스트됩니다. 이 오브젝트에 대해 변경 액세스만 감사되거나 읽기 및 변경 액세스 모두가 감사되는 경우 사용자 프로파일을 지정할 수 있습니다. *CHANGE - 시스템에 있는 모든 사용자가 이 오브젝트에 대한 모든 변경 액세스를 감사합니다. *ALL - 시스템에 있는 모든 사용자가 이 오브젝트에 대한 모든 액세스를 감사합니다. 모든 액세스가 읽기 또는 변경 조작으로 정의됩니다. *NOTAVL - 이 조작을 수행하는 사용자는 현재 오브젝트 생성 감사 값을 검색할 수 없습니다. *SYSVAL - 이 디렉토리에서 생성된 오브젝트에 대한 오브젝트 감사 값은 시스템 감사 값(QCRTOBJAUD)으로 결정됩니다.

표 8. QAEZDxxxxO(오브젝트 속성 저장) (계속)

필드명	필드 유형	필드 설명
QEZSCN	GRAPHIC (1)	종료 프로그램이 통합 파일 시스템 스캔 관련 종료점과 함께 등록될 때 오브젝트가 스캔되는지 여부. 유효한 값은 다음과 같습니다. x'00' (SCANNING_NO) - 오브젝트는 스캔 관련 종료 프로그램에 설명된 규칙에 따라 스캔되지 않습니다. **주:** 이 속성으로 오브젝트가 복원될 때 스캔 파일 시스템 제어(QSCANFSCTL) 값 *NOPOSTRST가 지정되지 않으면 오브젝트는 복원 후 최소 한 번 스캔됩니다. x'01' (SCANNING_YES) - 오브젝트가 수정되었거나 마지막으로 오브젝트가 스캔된 이후 스캐닝 소프트웨어가 갱신된 경우 오브젝트는 스캔 관련 종료 프로그램에 설명된 규칙에 따라 스캔됩니다. x'02' (SCANNING_CHGONLY) - 마지막으로 오브젝트가 스캔된 이후 오브젝트가 수정된 경우 오브젝트는 스캔 관련 종료 프로그램에 설명된 규칙에 따라 스캔됩니다. 스캐닝 소프트웨어가 갱신된 경우 오브젝트는 스캔되지 않습니다. 스캔 파일 시스템 제어(QSCANFSCTL) 시스템 값이 지정된 *USEOCOATR을 가지고 있는 경우에만 이 속성이 실시됩니다. 그렇지 않으면 속성이 SCANNING_YES인 것처럼 처리됩니다. **주:** 이 속성으로 오브젝트가 복원될 때 스캔 파일 시스템 제어(QSCANFSCTL) 값 *NOPOSTRST가 지정되지 않으면 오브젝트는 복원 후 최소 한 번 스캔됩니다.
QEZINHSCN	GRAPHIC (1)	종료 프로그램이 통합 파일 시스템 스캔 관련 종료점과 함께 등록될 때 디렉토리에 작성된 오브젝트가 스캔되는지 여부. 유효한 값은 다음과 같습니다. x'00' - 오브젝트가 디렉토리에 작성된 후에 오브젝트는 스캔 관련 종료 프로그램에서 설명되는 규칙에 따라 스캔되지 않습니다. **주:** 이 속성으로 오브젝트가 복원될 때 스캔 파일 시스템 제어(QSCANFSCTL) 값 *NOPOSTRST가 지정되지 않으면 오브젝트는 복원 이후 최소 한 번 스캔됩니다. x'01' - 오브젝트가 디렉토리에 작성된 후에 오브젝트가 수정되었거나 마지막으로 오브젝트가 스캔된 이후 스캐닝 소프트웨어가 갱신된 경우 오브젝트는 스캔 관련 종료 프로그램에 설명된 규칙에 따라 스캔됩니다. x'02' - 오브젝트가 디렉토리에 작성된 후에, 마지막으로 오브젝트가 스캔된 이후 오브젝트가 수정된 경우에만 오브젝트는 스캔 관련 나감 프로그래에 설명된 규칙에 따라 스캔됩니다. 스캐닝 소프트웨어가 갱신된 경우 오브젝트는 스캔되지 않습니다. 스캔 파일 시스템 제어(QSCANFSCTL) 시스템 값이 지정된 *USEOCOATR을 가지고 있는 경우에만 이 속성이 실시됩니다. 그렇지 않으면 속성이 SCANNING_YES인 것처럼 처리됩니다. **주:** 이 속성으로 오브젝트가 복원될 때 스캔 파일 시스템 제어(QSCANFSCTL) 값 *NOPOSTRST가 지정되지 않으면 오브젝트는 복원 후 최소 한 번 스캔됩니다.

표 8. QAEZDxxxxO(오브젝트 속성 저장) (계속)

필드명	필드 유형	필드 설명
QEZSSTATUS	GRAPHIC (1)	이 오브젝트와 연관된 스캔 상태. 이 필드는 다음 값 중 하나일 수 있습니다. x'00' (SCAN_REQUIRED) - 스캔 관련 종료 프로그램에 의해 아직 스캔되지 않았거나 마지막으로 스캔된 이후 오브젝트 데이터 또는 CCSID가 수정되었으므로 스캔이 오브젝트에 필요합니다. 오브젝트 데이터 또는 CCSID 수정의 예는 다음과 같습니다. 직접 또는 메모리 맵핑을 통해 오브젝트에 쓰기, 오브젝트 절단, 오브젝트 지우기 및 오브젝트 CCSID 속성 변경. x'01' (SCAN_SUCCESS) - 오브젝트가 스캔 관련 종료 프로그램에 의해 스캔되었으며 마지막 스캔 요청 시에 오브젝트가 스캔에 실패하지 않았습니다. x'02' (SCAN_FAILURE) - 오브젝트가 스캔 관련 종료 프로그램에 의해 스캔되었으며 마지막 스캔 요청 시에 오브젝트가 스캔에 실패했고 조작이 완료되지 않았습니다. 오브젝트가 실패로 표시되면, 적절한 경우 오브젝트의 스캔 서명이 글로벌 스캔 키 서명 또는 독립 ASP 그룹 스캔 키 서명과 다를 때까지 다시 스캔되지 않습니다. 그러므로 오브젝트에 대한 작업으로의 후속 요청은 스캔 실패 표시와 함께 실패합니다. 실패하는 요청의 예는 오브젝트 열기, 오브젝트의 CCSID 변경, 오브젝트 복사입니다. x'05' (SCAN_PENDING_CVN) - 오브젝트가 *TYPE2 디렉토리에 없으므로 디렉토리가 변환될 때까지 스캔되지 않습니다. x'06' (SCAN_NOT_REQUIRED) - 오브젝트가 스캔되지 않는 것으로 표시되므로 오브젝트는 어떠한 스캐닝도 요구하지 않습니다.
QEZSSIGDF	GRAPHIC (1)	스캔 서명은 스캐닝 소프트웨어 지원의 레벨 표시를 지정합니다. 오브젝트가 독립 ASP 그룹에 있으면 오브젝트 스캔 서명은 연관된 독립 ASP 그룹 스캔 서명과 비교됩니다. 오브젝트가 독립 ASP 그룹에 없으면 오브젝트 스캔 서명은 글로벌 스캔 서명 값과 비교됩니다. 이 필드는 다음 값 중 하나일 수 있습니다. x'00' - 비교된 서명이 다르지 않습니다. x'01' - 비교된 서명이 다릅니다.
QEZSBINARY	GRAPHIC (1)	이것은 오브젝트가 이전에 스캔되었을 때 2진 모드로 스캔된 경우를 표시합니다. 이 필드는 다음 값 중 하나일 수 있습니다. x'00' - 오브젝트가 2진 모드로 스캔되지 않았습니다. x'01' - 오브젝트가 2진 모드로 스캔되었습니다. 오브젝트 스캔 상태가 SCAN_SUCCESS이면 오브젝트는 2진 모드로 스캔되었습니다. 오브젝트 스캔 상태가 SCAN_FAILURE이면 오브젝트는 2진 모드로 스캔되지 않았습니다.
QEZSCCSID1	INTEGER	이것은 오브젝트가 이전에 스캔되었을 때 나열된 CCSID로 스캔된 경우를 표시합니다. 오브젝트 스캔 상태가 SCAN_SUCCESS이면 오브젝트는 이 CCSID로 스캔되었습니다. 오브젝트 스캔 상태가 SCAN_FAILURE이면 오브젝트는 이 CCSID로 스캔되지 않았습니다. 0값은 이 필드가 적용되지 않는 것을 의미합니다.
QEZSCCSID2	INTEGER	이것은 오브젝트가 이전에 스캔되었을 때 나열된 CCSID로 스캔된 경우를 표시합니다. 오브젝트 스캔 상태가 SCAN_SUCCESS이면 오브젝트는 이 CCSID로 스캔되었습니다. 오브젝트 스캔 상태가 SCAN_FAILURE이면 이 필드는 0입니다. 0값은 이 필드가 적용되지 않는 것을 의미합니다.
QEZUDATE	TIMESTAMP	오브젝트가 마지막으로 사용된 날짜에 해당하는 에포크 이후 초 수. 오브젝트가 작성될 때 이 필드는 0입니다. 사용법 데이터가 i5/OS 유형 또는 오브젝트가 속한 파일 시스템을 위해 유지보수되지 않으면 이 필드는 0입니다.

표 8. QAEZDxxxxO(오브젝트 속성 저장) (계속)

필드명	필드 유형	필드 설명
QEZUDCOUNT	INTEGER	오브젝트가 사용된 일 수. 사용법은 특정 파일 시스템에 따라 그리고 개개의 오브젝트 유형에 따라 파일 시스템 내에서 다른 의미를 지원하게 합니다. 사용법은 파일의 열기 또는 닫기를 표시하거나 링크 추가, 이름 변경, 복원 또는 오브젝트 체크 아웃을 참조할 수 있습니다. 이 계수는 오브젝트가 사용되는 각 일을 증가시키며 Qp0lSetAttr() API를 호출하여 0으로 재설정됩니다.
QEZURESET	INTEGER	사용되는 요일 계수가 마지막으로 0으로 재설정된 날짜에에 해당하는 에포크의 초 수. Qp0lSetAttr() API가 호출되어 사용된 날짜 계수를 0으로 재설정할 때 이 날짜는 현재 날짜로 설정됩니다.
QEZPRMLNK	SMALLINT	오브젝트가 몇 개의 이름을 가지고 있으면 이 필드는 찾은 첫 번째 이름에만 설정됩니다.
QEZALWCKPW	SMALLINT	저장 동안 사용 중인 체크 포인트 처리 중에 스트림 파일(*STMF)이 판독기 및 출력기와 공유될 수 있는지 여부. 유효한 값은 다음과 같습니다. 0 - 오브젝트가 판독기와만 공유될 수 있습니다. 1 - 오브젝트가 판독기 및 출력기와 공유될 수 있습니다.
QEZSIG[1]	SMALLINT	오브젝트에 i5/OS 디지털 서명이 있는지 여부. 유효한 값은 다음과 같습니다. 0 - 오브젝트에 i5/OS 디지털 서명이 없습니다. 1 - 오브젝트에 i5/OS 디지털 서명이 있습니다.
QEZSYSSIG	SMALLINT	오브젝트가 시스템이 신뢰할 수 있는 소스로 서명되었는지 여부. 유효한 값은 다음과 같습니다. 0 - 오브젝트가 시스템이 신뢰할 수 있는 소스에서 가져온 서명이 없습니다. 1 - 오브젝트가 시스템이 신뢰할 수 있는 소스로 서명되었습니다. 오브젝트가 다수의 서명을 가지고 있으면 서명 중 최소 한 개는 시스템이 신뢰할 수 있는 소스에서 가져온 것입니다.
QEZMLTSIG	SMALLINT	오브젝트에 둘 이상의 i5/OS 디지털 서명이 있는지 여부. 유효한 값은 다음과 같습니다. 0 - 오브젝트가 디지털 서명을 한 개만 가지고 있습니다. 1 - 오브젝트가 디지털 서명을 두 개 이상 가지고 있습니다. QEZSYSSIG 필드가 값 1을 가지고 있으면 서명 중 최소 한 개는 시스템이 신뢰할 수 있는 소스에서 가져온 것입니다.

표 8. *QAEZDxxxxO(오브젝트 속성 저장) (계속)*

필드명	필드 유형	필드 설명
QEZDSTGOPT	SMALLINT	이 옵션은 지정되는 오브젝트에 대해 시스템이 보조 기억장치를 얼마나 할당하는지 판별하기 위해 사용해야 합니다. 이 옵션은 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템의 스트림 파일에만 지정될 수 있습니다. 이 옵션은 *TYPE1 바이트 스트림 파일에서 무시됩니다. 유효한 값은 다음과 같습니다. 0 - 보조 기억장치가 정상적으로 할당됩니다. 즉, 추가 보조 기억장치가 필요하므로 현재 공간 요구사항을 수용하기 위해 논리적 *크기* 범위로 할당되며 디스크 I/O 조작의 수를 최소화하는 반면 향후 요구사항을 예상합니다. 1 - 보조 기억장치가 오브젝트가 사용하는 공간을 최소화하기 위해 할당됩니다. 즉, 추가 보조 기억장치가 필요하므로 현재 공간 요구사항을 수용하기 위해 작은 *크기* 범위로 할당됩니다. 많은 작은 범위로 구성된 오브젝트에 액세스하면 해당 오브젝트에 대해 디스크 I/O 조작 수를 늘릴 수 있습니다. 2 - 시스템이 오브젝트에 대해 최적의 보조 기억장치 할당을 동적으로 판별합니다(디스크 I/O 조작에 비해 사용되는 공간 균형 조절). 예를 들어, 파일이 많은 작은 *크기*를 가지고 있고 종종 읽고 쓰고 있으면 향후 보조 기억장치 할당은 디스크 I/O 조작의 수를 최소화하기 위해 더 *크게* 확장됩니다. 또는 파일이 자주 절단되면 향후 보조 기억장치 할당은 사용되는 공간을 최소화하기 위해 작게 확장됩니다. 또한 정보는 이 시스템과 활동에 대해 스트림 파일 *크기*로 유지됩니다. 이 파일 *크기* 정보는 기타 오브젝트 크기와 관련이 있으므로 이 오브젝트에 대해 최적의 보조 기억장치 할당을 판별하는 데 도움을 주기 위해서도 사용됩니다.
QEZMSTGOPT	SMALLINT	이 옵션은 지정되는 오브젝트에 대해 시스템이 주 기억장치를 얼마나 할당하고 사용하는지 판별하기 위해 사용해야 합니다. 이 옵션은 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템의 스트림 파일에만 지정될 수 있습니다. 유효한 값은 다음과 같습니다. 0 - 주 기억장치가 정상적으로 할당됩니다. 즉, 가능한 한 주 기억장치가 많이 할당되고 사용됩니다. 정보가 주 기억장치에 캐시되므로 이는 디스크 I/O 조작 수를 최소화합니다. 1 - 주 기억장치가 오브젝트가 사용하는 공간을 최소화하기 위해 할당됩니다. 즉, 가능한 한 주 기억장치가 적게 할당되고 사용됩니다. 적은 정보가 주 기억장치에 캐시되므로 이는 디스크 I/O 조작 수를 늘리는 반면 주 기억장치 사용을 최소화합니다. 2 - 시스템은 기타 시스템 활동과 주 기억장치 경합에 따라 오브젝트에 대해 최적의 주 기억장치 할당을 동적으로 판별합니다. 즉, 주 기억장치 경합이 적으면 디스크 I/O 조작 수를 최소화하는 데 가능한 한 기억장치가 많이 할당되고 사용됩니다. 중요한 주 기억장치 경합이 있으면 주 기억장치 경합을 최소화하기 위해 주 기억장치가 적게 할당되고 사용됩니다. 이 옵션은 기억장치 풀의 페이징 옵션이 *CALC일 때만 영향을 미칩니다. 기억장치 풀의 페이징 옵션이 *FIXED이면 작동은 STG_NORMAL과 같습니다. 오브젝트가 파일 서버를 통해 액세스되면 이 옵션은 영향을 미치지 않습니다. 대신, 작동은 STG_NORMAL과 같습니다.
QEZDIRTYP2	SMALLINT	지정되는 디렉토리 오브젝트의 형식. 유효한 값은 다음과 같습니다. 0 - 디렉토리 형식이 *TYPE1입니다. 1 - 디렉토리 형식이 *TYPE2입니다.

표 8. QAEZDxxxxO(오브젝트 속성 저장) (계속)

필드명	필드 유형	필드 설명
QEZFILTYP2[1]	SMALLINT	스트림 파일(*STMF)의 형식. 유효한 값은 다음과 같습니다. 0 - 스트림 파일 형식이 *TYPE1입니다. 1 - 스트림 파일 형식이 *TYPE2입니다.
QEZUDFTYP2	SMALLINT	사용자 정의 파일 시스템에 작성되는 스트림 파일(*STMF)의 디폴트 파일 형식. 유효한 값은 다음과 같습니다. 0 - 스트림 파일 형식이 *TYPE1입니다. 1 - 스트림 파일 형식이 *TYPE2입니다.
QEZNONSAV	SMALLINT	오브젝트가 저장될 수 있는지 여부. 유효한 값은 다음과 같습니다. 0 - 오브젝트가 저장됩니다. 1 - 오브젝트가 저장되지 않습니다. 또한 이 오브젝트가 디렉토리이면 저장될 오브젝트로 명백히 지정되지 않는 한 디렉토리의 서브트리에 어떠한 오브젝트도 저장되지 않습니다. 서브트리는 모든 서브디렉토리 및 그 서브디렉토리 내의 오브젝트를 포함합니다.
QEZCLSTRSP	SMALLINT	오브젝트는 xSeries 서버용 가상 디스크 드라이브로 사용하기 위해 Integrated xSeries Server에 할당된 기억장치입니다. iSeries 서버의 관점에서 가상 드라이브는 통합 파일 시스템 내에서 바이트 스트림 파일로 나타납니다. 0 - 오브젝트가 가상 디스크 기억장치가 아닙니다. 1 - 오브젝트가 가상 디스크 기억장치입니다.
QEZCASE	SMALLINT	이 오브젝트를 포함하는 파일 시스템의 대소문자 구분을 표시합니다. 0 - 파일 시스템이 대소문자를 구분하지 않습니다. 1 - 파일 시스템이 대소문자를 구분합니다.
QEZOFLOW	SMALLINT	오브젝트가 상주하는 보조 기억장치 풀에서 넘치는지를 나타냅니다. 유효한 값은 다음과 같습니다. 0 - 보조 기억장치가 넘치지 않습니다. 1 - 보조 기억장치가 넘칩니다.
QEZPCREAD	SMALLINT	오브젝트가 기록되거나 삭제될 수 있는지, 확장 속성을 변경하거나 삭제했는지, 크기를 변경했는지 여부. 유효한 값은 다음과 같습니다. 0 - 오브젝트를 변경할 수 있습니다. 1 - 오브젝트를 변경할 수 없습니다.
QEZPCHID[1]	SMALLINT	오브젝트를 보통 디렉토리 리스팅을 사용하여 표시할 수 있는지 여부 0 - 오브젝트가 숨김이 아닙니다. 1 - 오브젝트가 숨김입니다.
QEZPCSYS	SMALLINT	오브젝트가 시스템 파일이며 보통 디렉토리 검색에서 제외되는지 여부 0 - 오브젝트가 시스템 파일이 아닙니다. 1 - 오브젝트가 시스템 파일입니다.

표 8. QAEZDxxxxO(오브젝트 속성 저장) (계속)

필드명	필드 유형	필드 설명
QEZPCARC	SMALLINT	오브젝트가 마지막으로 검사된 이후 변경되었는지 여부 0 - 오브젝트가 변경되지 않았습니다. 1 - 오브젝트가 변경되었습니다.
QEZSYSARC	SMALLINT	오브젝트가 변경되었고 저장해야 하는지 여부. 이는 오브젝트의 변경 시간이 갱신될 때 설정되며 오브젝트가 저장되면 해제됩니다. 0 - 오브젝트가 변경되지 않았고 저장할 필요가 없습니다. 1 - 오브젝트가 변경되었고 저장해야 합니다.
QEZJRCVNAM	GRAPHIC(10)	기존 저널 리시버는 저널 변경사항을 정상적으로 적용하기 위해 필요합니다. 적용 정보 필드가 PARTIAL_TRANSACTION으로 설정되면 저널 리시버는 부분 트랜잭션의 시작을 포함합니다. 그렇지 않으면, 저널 리시버는 저장 조작의 시작을 포함합니다. 오브젝트가 저널링되지 않은 경우 이 필드는 공백입니다.
QEZJRCVLIB	GRAPHIC(10)	저널 변경사항을 정상적으로 적용하기 위해 필요한 저널 리시버를 포함하는 라이브러리 명.오브젝트가 저널링되지 않은 경우 이 필드는 공백입니다.
QEZJRCVASP	GRAPHIC(10)	저널 변경사항을 정상적으로 적용하기 위해 필요한 저널 리시버를 포함하는 ASP의 이름. 유효한 값은 다음과 같습니다. *SYSBAS - 시스템 또는 사용자 ASP에 상주하는 저널 리시버. ASP 장치 - 저널 리시버를 포함하는 ASP 장치명.
QEZJTRNI	GRAPHIC(1)	이 필드는 확약 제어 경계에 관련이 있으므로 오브젝트의 현재 상태에 대한 정보를 설명합니다. 유효한 값은 다음과 같습니다. x'00' (NONE) - 부분 트랜잭션이 없습니다. x'01' (PARTIAL_TRANSACTION) - 오브젝트가 부분 트랜잭션으로 복원되었습니다. APYJRNCHG(저널링된 변경사항 적용) 또는 RMVJRNCHG(저널링된 변경사항 제거) 명령을 사용하여 부분 트랜잭션을 완료하거나 롤백할 때까지 이 오브젝트를 사용할 수 없습니다. x'02' (ROLLBACK_ENDED) - 오브젝트가 WRKCMTDFN(확약 정의에 대한 작업) 화면에서 "롤백 종료" 옵션을 사용하여 롤백 조작을 종료했습니다. 오브젝트를 사용할 수 없으므로 복원할 것을 권장합니다. 마지막 옵션으로 CHGJRNOBJ(저널링된 오브젝트 변경) 명령을 사용하여 오브젝트가 사용되도록 할 수 있습니다. 그러나 이 옵션을 사용하면 오브젝트가 일관성이 없는 상태가 될 수 있습니다.

주:

1. 이 필드는 PRTDIRINF 명령에서 사용되는 필드에 서브세트에 포함됩니다.
2. 이 필드에서는 오브젝트명만 저장됩니다. 경로명의 나머지는 디렉토리명의 길이가 1KB 이하이면 QEZDIRNAM1 필드에 저장되며 디렉토리명이 1KB 이상이면 QEZDIRNAM2 필드에 저장됩니다.

다음 표는 RTVDIRINF 명령에서 처리되는 디렉토리를 나열하는 표의 예입니다.

표 9. QAEZDxxxD(디렉토리 속성 저장)

필드명	필드 유형	필드 설명
QEZDIRIDX	INTEGER	디렉토리 및 오브젝트 테이블 간의 관계 ID입니다. 이 테이블을 조인(Join)하여 완전한 경로명을 얻을 수 있습니다. 오브젝트의 상위 디렉토리를 얻기 위해 이 오브젝트 테이블의 QEZDIRIDX 필드 값과 일치하는 디렉토리 테이블의 QEZDIRIDX 값을 찾습니다.
QEZDIRNAM1[1]	VARGRAPHIC (1024)	상위 디렉토리 경로. 경로 길이가 1KB(1024바이트) 이하인 경우에만 사용됩니다.
QEZDIRNAM2[1]	DBCLOB (16M)	상위 디렉토리 경로. 경로 길이가 1KB(1024바이트) 이상인 경우에만 사용됩니다. 최대 16MB까지 경로를 저장할 수 있습니다.
QEZDRCCSID	INTEGER	디렉토리 CCSID
QEZDREGION	GRAPHIC (2)	디렉토리 경로 영역 ID
QEZLANGID	GRAPHIC (3)	디렉토리 경로 언어 ID
QEZDIRLEN[1]	INTEGER	디렉토리의 경로명 길이
QEZDIRFID	GRAPHIC (16)	디렉토리의 파일 ID. 오브젝트와 연관된 ID. 파일 ID는 오브젝트의 경로명을 검색하기 위해 Qp0lGetPathFromFileID() 와 함께 사용될 수 있습니다.
QEZDFID	INTEGER	디렉토리의 파일 ID
QEZDIRFSID	BIGINT	디렉토리의 파일 시스템 ID
QEZDIRGID	BIGINT	생성 ID
QEZPARDIR	INTEGER	상위 디렉토리 색인
주: 1. 이 필드는 PRTDIRINF 명령에서 사용되는 필드에 서브세트에 포함됩니다.		

다음 표는 RTVDIRINF 명령이 실행될 때 작성한 파일에 대해 저장하는 정보를 표시합니다. 이 정보가 들어 있는 파일이 존재하지 않으면 RTVDIRINF 명령이 파일을 작성합니다. 후속적으로 명령이 실행되는 경우 기존 파일에 정보가 추가됩니다. PRTDIRINF 명령은 이 정보를 사용하여 RTVDIRINF 명령의 다른 인스턴스에서 검색된 정보를 저장하는 데 사용된 데이터베이스 파일을 판별합니다.

표 10. QUSRSYS/QAEZDBFILE(작성된 파일 저장)

필드명	필드 유형	필드 설명
QEZDIRSRC	VARGRAPHIC (5000)	DIR 매개변수(RTVDIRINF)에 지정된 경로
QEZPRCCSID	INTEGER	경로 CCSID
QEZPREGION	GRAPHIC (2)	경로 영역 ID
QEZPLANGID	GRAPHIC (3)	경로 언어 ID
QEZOBJFILE[1]	VARGRAPHIC (20)	오브젝트 속성을 저장하여 생성된 파일명
QEZDIRFILE[1]	VARGRAPHIC (20)	디렉토리 색인을 저장하여 생성된 파일명
QEZLIB[1]	VARGRAPHIC (20)	모든 생성된 파일이 상주하는 라이브러리
QEZSTRTIME	TIMESTAMP	RTVDIRINF가 제출된 날짜/시간
QEZENDTIME	TIMESTAMP	RTVDIRINF가 완료된 날짜/시간
주: 1. 이 필드는 PRTDIRINF 명령에서 사용되는 필드에 서브세트에 포함됩니다.		

관련 정보

RTVDIRINF(디렉토리 정보 검색) 명령

Qp0lGetPathFromFileID()--파일 ID로 오브젝트 경로명 얻기 API

Qp0lSetAttr()--속성 설정 API

APYJRNCHG(저널링된 변경사항 적용) 명령

RMVJRNCHG(저널링된 변경사항 제거) 명령

CHGJRNOBJ(저널링된 오브젝트 변경) 명령

PRTDIRINF(디렉토리 정보 인쇄) 명령

RTVDIRINF의 데이터에 액세스:

표에 있는 데이터에 액세스하는 몇 가지 옵션이 있습니다.

다음은 RTVDIRINF(디렉토리 정보 검색) 명령을 통해 작성된 데이터에 액세스할 수 있는 방법입니다.

- PRTDIRINF(디렉토리 정보 인쇄) 명령 사용

  이 명령은 통합 파일 시스템에서 디렉토리 정보 및 오브젝트에 관한 디렉토리 정보를 인쇄하는 데 사용됩니다. 인쇄할 정보는 RTVDIRINF 명령에서 사용자가 지정한 데이터베이스 파일에 저장됩니다.
- i5/OS 오퍼레이팅 시스템에서 DB2® 테이블을 쿼리하기 위해 IBM®이 제공하는 프로그램 또는 명령을 이용합니다.

  보다 일반적인 툴의 일부는 STRSQL(대화식 세션 시작) 명령 및 System i Navigator입니다.

  예를 들어, 10KB보다 큰 할당 크기를 가지는 특정 경로(이전에 RTVDIRINF 명령을 통해 수집된)에서 오브젝트를 선택하려는 경우 다음과 같이 쿼리를 수행할 수 있습니다.

  ```
  SELECT QEZOBJNAM, QEZALCSIZE FROM library_name/QAEZDxxxxO WHERE
  QEZALCSIZE > 10240
  ```
- 유효한 DB 메소드를 사용하여 데이터베이스 표에 액세스하고 사용자 소유 프로그램을 작성할 수 있습니다.
- 명령 호출을 통해 다양한 쿼리를 실행하는 방법 이외에도 System i Navigator를 사용하여 쉽게 System i Navigator의 디렉토리 정보 데이터(폴더 속성 데이터)를 검색, 표시 및 분석할 수 있습니다. 자세한 정보는 System i Navigator를 사용하여 폴더 속성 수집 및 검색하기를 참조하십시오.

관련 태스크

92 페이지의『System i Navigator를 사용하여 폴더 속성 수집 및 검색하기』

System i Navigator를 사용하여 통합 파일 시스템 내 오브젝트의 속성을 수집 및 분석할 수 있습니다. 이 사용하기 쉬운 그래픽 인터페이스는 디렉토리 정보 검색(RTVDIRINF) 명령과 동일한 기능을 제공합니다. RTVDIRINF 명령으로 수집된 것 뿐만 아니라 이 인터페이스로 수집된 데이터를 검토 및 쿼리할 수 있습니다.

관련 정보

PRTDIRINF(디렉토리 정보 인쇄) 명령

STRSQL(SQL 대화식 세션 시작) 명령

삽입된 SQL 프로그램밍

SQL 호출 레벨 인터페이스

RTVDIRINF의 데이터 사용:

다음은 데이터가 중요한 이유 또는 세 개의 표 각각에서 산출된 데이터를 사용할 수 있는 방법을 표시하는 예입니다.

- 80 페이지의 표 8의 경우, 이 표 내의 필드에 기반하는 통계 또는 보고서를 작성하기 위해 쿼리를 만들 수 있습니다. PRTDIRINF는 모든 필드에 기반하는 보고서를 포함하지 않습니다. 대신, 서브세트가 사용됩니다.
- 90 페이지의 표 9의 데이터는 RTVDIRINF 명령의 DIR 매개변수에 지정된 경로 내의 모든 디렉토리를 포함합니다. 경로명에 관한 특정 속성을 알아보려면(예: CCSID, 언어 ID 또는 길이) 이 데이터가 유용합니다. 또한 이 표에 저장된 각 디렉토리는 고유 값 또는 이를 식별할 수 있는 색인을 가지고 있습니다. 80 페이지의 표 8에서는 같은 필드(QEZDIRIDX)를 찾을 수 있으며, 이는 오브젝트가 어느 디렉토리에 속하는지 알려줍니다. 어느 오브젝트가 어느 디렉토리에 속하는지 알아보려면 결합을 사용하여 쿼리를 만들 수 있습니다. 예를 들어, 다음 쿼리문은 ″/MYDIR″ 디렉토리에 있는 모든 오브젝트의 이름을 선택합니다.

```
SELECT QAEZDxxxxO.QEZOBJNAM FROM library_name/QAEZDxxxxD,
library_name/QAEZDxxxxO WHERE QAEZDxxxxD.QEZDIRNAM1 = '/MYDIR' AND
QAEZDxxxxD.QEZDIRIDX = QAEZDxxxxO.QEZDIRIDX
```

- 90 페이지의 표 10은 대개 PRTDIRINF 명령에서 RTVDIRINF 실행에 대한 특정 데이터를 얻기 위해 사용됩니다. 이 예는 다음과 같습니다. 작성되는 표의 이름, 표가 상주하는 라이브러리 및 처리의 시작과 종료 시간. 이 표를 사용하여 RTVDIRINF가 발생된 시기 또는 이를 쿼리하기 위해 탐색해야 하는 표를 알아볼 수 있습니다.

System i Navigator를 사용하여 폴더 속성 수집 및 검색하기:

System i Navigator를 사용하여 통합 파일 시스템 내 오브젝트의 속성을 수집 및 분석할 수 있습니다. 이 사용하기 쉬운 그래픽 인터페이스는 디렉토리 정보 검색(RTVDIRINF) 명령과 동일한 기능을 제공합니다. RTVDIRINF 명령으로 수집된 것 뿐만 아니라 이 인터페이스로 수집된 데이터를 검토 및 쿼리할 수 있습니다.

주: 이 방법은 System i Navigator V6R1 또는 그 이후 버전에서만 사용 가능합니다.

통합 파일 시스템 내 특정 오브젝트의 속성에 대한 보고서를 작성하려면 다음 단계를 따르십시오.

1. System i Navigator에서 **내 연결** → *시스템* → **파일 시스템** → **통합 파일 시스템**을 확장하십시오.
2. 관심이 있는 오브젝트를 포함하고 있는 폴더에서 마우스 오른쪽 단추를 누르고 **폴더 속성 정보** → **속성 수집**을 선택하십시오.
3. 속성 수집 창에서 기본 설정을 지정하십시오. 하위 폴더의 속성도 수집하려면 **이 폴더에 포함된 하위 폴더의 내용 포함**을 선택하십시오. 선택적으로 **파일 접두부** 및 **라이브러리**를 지정할 수 있습니다. **확인**을 눌러서 오브젝트 속성 수집을 시작하십시오.

이 데이터 수집 프로세스는 시간이 걸릴 수 있습니다. 수집된 속성 표시 창에 표시되기까지 수 초를 기다려야 합니다. **새로 고침**을 눌러 데이터 수집 프로세스의 최신 상태를 볼 수 있습니다. 상태는 **완료**, **실패** 또는 **진행 중**입니다. **진행 중** 상태가 표시되지 않을 때까지 기다린 후 다음 단계로 진행합니다.

4. 수집된 속성 표시 창의 분석하려고 하는 항목에서 마우스 오른쪽 단추를 누르고 **정보 분석**을 선택하십시오.

   주: System i Navigator에서 속성 수집 작업을 수행하거나 이전에 디렉토리 정보 검색(RTVDIRINF) 명령을 실행했다면 해당 폴더에서 마우스 오른쪽 단추를 누르고 **폴더 속성 정보 → 수집된 속성 표시**를 선택하여 직접 수집된 속성 표시 창으로 갈 수 있습니다.

5. 폴더 정보 분석 창의 **열**, **필터** 및 **순서** 탭에서 보고자 하는 속성을 사용자 정의할 수 있습니다. **확인**을 눌러 폴더 속성 정보 보고서를 생성하십시오.

   예는 다음과 같습니다. *크기*가 10MB *초과*하는 파일과 *그것의 소유자*를 *파일 크기*로 먼저 정렬하고 다음 *소유자*로 정렬하여 표시하는 상황을 가정합니다.

   - **열** 탭에서 *소유자*를 선택하고 **추가**를 누르십시오. *상위 폴더 경로*를 선택하고 **추가**를 누르십시오. *오브젝트명*을 선택하고 **추가**를 누르십시오. *할당 크기*를 선택하고 **추가**를 누르십시오.
   - **필터** 탭의 **필드**에서 *할당 크기*를, **조건**에서 *다음을 초과하는 크기*를 선택하고 **크기**에 10을 입력한 후 **단위**에서 *MB*를 선택하십시오. 필터를 작성하려면 **추가**를 누르십시오.
   - **순서** 탭의 **첫 번째 정렬**에서 *할당 크기* 및 *내림차순*을 선택하고 **두 번째 정렬**에서 *소유자* 및 *내림차순*을 선택하십시오.
   - **확인**을 클릭하십시오. 폴더 속성 정보 보고서는 사용자가 지정한 방식으로 가공된 정보를 표시합니다.

   관련 참조

   91 페이지의 『RTVDIRINF의 데이터에 액세스』

   표에 있는 데이터에 액세스하는 몇 가지 옵션이 있습니다.

API를 사용한 액세스

API(Application Programming Interface)를 사용하여 통합 파일 시스템에 액세스할 수 있습니다.

관련 참조

122 페이지의 『API를 사용한 조작 수행』

통합 파일 시스템 오브젝트에서 조작을 수행하는 API(Application Programming Interface)의 대부분은 C 언어 함수 형태입니다.

PC를 사용한 액세스

PC가 System i 제품에 연결되어 있는 경우 PC에 저장되어 있는 것처럼 통합 파일 시스템의 디렉토리 및 오브젝트와 대화할 수 있습니다.

Windows 탐색기의 끌어서 놓기 기능을 사용하여 디렉토리 간에 오브젝트를 복사할 수 있습니다. 필요한 경우, 시스템 드라이브의 오브젝트를 선택하여 PC 드라이브로 해당 오브젝트를 끌면, 실제로 오브젝트를 시스템에서 PC로 복사할 수 있습니다.

Windows 인터페이스를 사용하여 System i 제품 및 PC 사이에 복사된 모든 오브젝트는 EBCDIC 및 ASCII 사이에 자동으로 변환될 수 있습니다. 이 변환을 자동으로 수행하도록 System i Access 제품군을 구성할 수 있으며 이 변환이 특정 확장자를 가진 파일에서 수행되도록 지정할 수도 있습니다.

오브젝트 유형에 따라 PC 인터페이스 및 PC 어플리케이션을 사용하여 오브젝트에 대해 작업할 수 있습니다. 예를 들어, 텍스트가 들어 있는 스트림 파일은 PC 편집기를 사용하여 편집할 수 있습니다.

PC를 사용하여 System i 제품에 연결된 경우 통합 파일 시스템은 PC에서 시스템의 디렉토리 및 오브젝트를 사용할 수 있도록 합니다. PC는 Windows 오퍼레이팅 시스템, FTP 클라이언트 또는 System i Navigator(System i Access 제품군의 일부)에 내장된 파일 공유 클라이언트를 사용하여 통합 파일 시스템의 파일에 대한 작업을 수행할 수 있습니다. PC는 Windows 파일 공유 클라이언트를 사용하여 시스템에서 실행되는 i5/OS NetServer에 액세스합니다.

관련 개념

『System i Navigator를 사용하여 액세스』
System i Navigator는 Windows 데스크탑에서 시스템을 관리하기 위한 그래픽 인터페이스입니다. System i Navigator는 시스템을 보다 수월하게 운영 및 관리하여 생산성을 높일 수 있도록 합니다.

관련 태스크

95 페이지의 『i5/OS NetServer를 사용하여 액세스』
Windows Network Neighborhood (i5/OS NetServer)용 i5/OS 지원은 Windows 클라이언트가 i5/OS 공유 디렉토리 경로 및 공유 출력 대기행렬에 액세스할 수 있도록 하는 기능입니다. i5/OS NetServer를 사용하여 Windows 소프트웨어를 실행하는 PC가 System i 플랫폼에서 관리하는 데이터 및 프린터에 끊임없이 액세스할 수 있습니다.

관련 참조

96 페이지의 『파일 전송 프로토콜을 사용한 액세스』
파일 전송 프로토콜(FTP) 클라이언트를 사용하면 System i 플랫폼에 있는 파일을 전송할 수 있습니다.

System i Navigator를 사용하여 액세스

System i Navigator는 Windows 데스크탑에서 시스템을 관리하기 위한 그래픽 인터페이스입니다. System i Navigator는 시스템을 보다 수월하게 운영 및 관리하여 생산성을 높일 수 있도록 합니다.

인스턴스의 경우 한 시스템에서 다른 시스템으로 사용자 프로파일을 끌면 사용자 프로파일을 다른 시스템에 복사할 수 있습니다. 마법사는 보안과 TCP/IP 서비스 및 어플리케이션 설정을 안내합니다.

| System i Navigator를 사용하여 많은 타스크를 수행할 수 있습니다. 다음에는 시작하는 데 도움이 되는 몇
| 가지 일반적인 통합 파일 시스템 타스크입니다.

파일 및 폴더에 대한 작업

- 140 페이지의 『폴더 작성』
- 140 페이지의 『파일 또는 폴더 제거』
- 142 페이지의 『권한 설정』

- 143 페이지의『텍스트 파일 변환 설정』
- 143 페이지의『다른 시스템으로 파일 또는 폴더 송신』
- 144 페이지의『파일 또는 폴더 전송 옵션 변경』
- 147 페이지의『오브젝트가 스캔되어야 하는지 여부 설정』
- 148 페이지의『오브젝트 체크 인』
- 149 페이지의『오브젝트 체크 아웃』
- 92 페이지의『System i Navigator를 사용하여 폴더 속성 수집 및 검색하기』

파일 공유에 대한 작업

- 144 페이지의『파일 공유 작성』
- 144 페이지의『파일 공유 변경』

사용자 정의 파일 시스템에 대한 작업

- 145 페이지의『새로운 사용자 정의 파일 시스템 작성』
- 145 페이지의『사용자 정의 파일 시스템 마운트』
- 146 페이지의『사용자 정의 파일 시스템 마운트 해제』
- 146 페이지의『동적으로 마운트된 파일 시스템에 대한 작업』

오브젝트 저널링

- 111 페이지의『저널링 시작』
- 112 페이지의『저널링 종료』

i5/OS NetServer를 사용하여 액세스

Windows Network Neighborhood (i5/OS NetServer)용 i5/OS 지원은 Windows 클라이언트가 i5/OS 공유 디렉토리 경로 및 공유 출력 대기행렬에 액세스할 수 있도록 하는 기능입니다. i5/OS NetServer를 사용하여 Windows 소프트웨어를 실행하는 PC가 System i 플랫폼에서 관리하는 데이터 및 프린터에 끊임없이 액세스할 수 있습니다.

네트워크의 PC 클라이언트는 해당 오퍼레이팅 시스템에 포함된 파일 및 인쇄 공유 기능을 사용합니다. 따라서 i5/OS NetServer를 사용하기 위해 PC에 추가 소프트웨어를 설치할 필요가 없습니다.

Samba 클라이언트 소프트웨어가 설치된 Linux 클라이언트도 i5/OS NetServer를 통해 데이터 및 프린터에 끊임없이 액세스할 수 있습니다. i5/OS에서 내보낸 NFS 파일 시스템을 마운트하는 것과 유사한 방식으로 i5/OS NetServer 디렉토리는 Samba 파일 시스템(smbfs)으로 Linux 클라이언트에서 마운트할 수 있습니다.

i5/OS NetServer 파일 공유는 System i 네트워크의 클라이언트와 공유하는 i5/OS NetServer 디렉토리 경로입니다. 파일 공유는 시스템의 임의의 통합 파일 시스템 디렉토리로 구성될 수 있습니다. i5/OS NetServer를 사용하여 파일 공유에 대한 작업을 하기 전에 i5/OS NetServer 파일 공유를 작성하고 필요한 경우 System i Navigator를 사용하여 i5/OS NetServer 파일 공유를 변경해야 합니다.

i5/OS NetServer를 사용하여 통합 파일 시스템 파일 공유에 액세스하려면 다음을 수행하십시오.

1. Windows PC에서 **시작**을 마우스 오른쪽 버튼으로 클릭한 후 **탐색**을 선택하여 Windows 탐색기를 여십시오.
2. 툴 메뉴를 열고 **네트워크 드라이브 연결**을 선택하십시오.
3. 사용하지 않는 드라이브 문자를 파일 공유용으로 선택하십시오(예: I:₩ 드라이브).
4. i5/OS NetServer 파일 공유명을 입력하십시오. 예를 들어, ₩₩QSYSTEM1₩*Sharename*과 같은 구문을 입력할 수 있습니다.

   주: QSYSTEM1은 i5/OS NetServer의 시스템명이며 *Sharename*은 사용할 파일 공유명입니다.
5. **확인**을 클릭하십시오.

주: i5/OS NetServer를 사용하여 액세스할 때 시스템명이 System i Access 제품군에서 사용되는 이름과 다를 수 있습니다. 예를 들어, i5/OS NetServer명은 QAS400X이며 파일에 대해 작업하기 위한 경로는 ₩₩QAS400X₩QDLS₩MYFOLDER.FLR₩MYFILE.DOC일 수 있습니다. 그러나, System i Access 제품군명은 AS400X이며 파일에 대해 작업하기 위한 경로는 ₩₩AS400X₩QDLS₩MYFOLDER.FLR₩MYFILE.DOC일 수 있습니다.

i5/OS NetServer를 사용하여 네트워크와 공유할 디렉토리를 선택합니다. 해당 디렉토리가 시스템명 아래의 첫 번째 레벨로서 나타납니다. 예를 들어, /home/fred 디렉토리를 fredsdir이라는 이름으로 공유하는 경우 PC에서는 ₩₩QAS400X₩FREDSDIR, Linux 클라이언트에서는 //qas400x/fredsdir이라는 이름으로 이 디렉토리에 액세스할 수 있습니다.

″루트″ (/) 파일 시스템은 다른 i5/OS 파일 시스템보다 탁월한 PC 파일 서비스 성능을 제공합니다. 파일을 ″루트″(/) 파일 시스템으로 이동할 수 있습니다. 더 자세한 정보는 141 페이지의 『다른 파일 시스템으로 파일 또는 폴더 이동』을 참조하십시오.

관련 정보

i5/OS NetServer

i5/OS NetServer 파일 공유

파일 전송 프로토콜을 사용한 액세스

파일 전송 프로토콜(FTP) 클라이언트를 사용하면 System i 플랫폼에 있는 파일을 전송할 수 있습니다.

또한 문서 라이브러리 서비스(QDLS) 파일 시스템의 폴더 및 문서를 전송할 수 있습니다. FTP 클라이언트는 클라이언트 파일에서 부속 명령을 읽고 이들 부속 명령에 대한 응답을 파일로 기록하는 무인 일괄처리 모드에서 대화식으로 실행할 수 있습니다. 또한 시스템에서 파일을 조작하기 위한 다른 기능도 포함되어 있습니다.

다음과 같은 파일 시스템간에 파일을 전송할 경우 FTP를 사용할 수 있습니다.

- 『루트』 (/) 파일 시스템
- 개방 시스템 파일 시스템(QOpenSys)
- 라이브러리 파일 시스템(QSYS.LIB)
- 독립 ASP QSYS.LIB 파일 시스템

- 문서 라이브러리 서비스 파일 시스템(QDLS)
- 광 파일 시스템(QOPT)
- 네트워크 파일 시스템(NFS)
- i5/OS NetClient 파일 시스템(QNTC)
- QFileSvr.400 파일 시스템

그러나 다음과 같은 제한사항에 유의하십시오.

- 통합 파일 시스템은 FTP 지원을 파일 데이터 전송으로만 제한합니다. 속성 데이터를 전송하는 경우에는 FTP를 사용할 수 없습니다.
- QSYS.LIB 및 독립 ASP QSYS.LIB 파일 시스템은 FTP 지원을 실제 파일 멤버, 소스 실제 파일 멤버 및 저장 파일로만 제한합니다. 프로그램(*PGM)과 같은 기타 오브젝트 유형을 전송하는 경우에는 FTP를 사용할 수 없습니다. 그러나 기타 오브젝트 유형은 저장 파일에 저장할 수 있으며, 저장 파일을 전송한 후 오브젝트를 복원할 수 있습니다.

관련 정보

파일 전송 프로토콜

파일 전송 프로토콜을 사용한 파일 전송

통합 파일 시스템 변환

i5/OS 오퍼레이팅 시스템은 새 디렉토리 형식 또는 유니코드 표준을 지원하는 통합 파일 시스템의 파일 시스템에 대해 자동 변환을 실행합니다.

디렉토리를 *TYPE1에서 *TYPE2로 변환

통합 파일 시스템의 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템(UDFS)은 *TYPE2 디렉토리 형식을 지원합니다.

*TYPE2 디렉토리 형식은 기존의 *TYPE1 디렉토리 형식을 개선한 것입니다. *TYPE2 디렉토리의 내부 구조는 *TYPE1 디렉토리와 다르며 성능과 신뢰성이 향상되었습니다.

i5/OS V5R3 이상의 릴리스가 설치된 후에 아직 변환되지 않은 파일 시스템에 대해 *TYPE2 디렉토리로의 변환이 자동으로 시작됩니다. 이 변환은 시스템 활동에 현저하게 영향을 주어서는 안됩니다.

*TYPE1에서 *TYPE2로 변환의 개요

통합 파일 시스템의 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템(UDFS)은 *TYPE2 디렉토리 형식을 지원합니다.

*TYPE2 디렉토리 형식은 기존의 *TYPE1 디렉토리 형식을 개선한 것입니다. *TYPE2 디렉토리의 내부 구조는 *TYPE1 디렉토리와 다르며 성능과 신뢰성이 향상되었습니다. 향상되는 성능 및 신뢰성 외에도 통합 파일 시스템 스캐닝 지원과 같은 새로운 기능은 *TYPE2 디렉토리의 오브젝트에만 사용할 수 있습니다. 더 자세한 정보는 20 페이지의 『스캐닝 지원』을 참조하십시오.

i5/OS V5R3M0 이상의 릴리스가 설치된 후에 아직 변환되지 않은 파일 시스템에 대해 *TYPE2 디렉토리로의 변환이 자동으로 시작됩니다. 이 변환은 낮은 우선순위 백그라운드 작업에서 실행되므로 시스템 활동에 현저하게 영향을 주어서는 안됩니다.

변환 기능이 아직 완료되지 않은 경우 및 시스템이 정상 또는 비정상 IPL을 가지고 있는 경우 IPL이 완료되면 변환 기능이 재개됩니다. 모든 적합한 파일 시스템이 완전하게 변환될 때까지 변환은 초기 프로그램 로드(IPL)마다 재시작합니다.

이 자동 변환에 적합한 파일 시스템은 ″루트″(/), QOpenSys 및 ASP용 사용자 정의 파일 시스템 1 ~ 32입니다.

주: V5R3 이상 릴리스의 오퍼레이팅 시스템을 설치하기 전에 파일 시스템을 변환하면 *TYPE2 디렉토리로의 자동 변환을 피할 수 있습니다.

관련 개념

10 페이지의 『*TYPE2 디렉토리』
통합 파일 시스템의 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템(UDFS)은 *TYPE2 디렉토리 형식을 지원합니다. *TYPE2 디렉토리 형식은 기존의 *TYPE1 디렉토리 형식을 개선한 것입니다.

관련 참조

『변환 상태 결정』
i5/OS V5R3M0 이상의 릴리스가 설치된 후에 아직 변환되지 않은 파일 시스템에 대해 *TYPE2 디렉토리로의 변환이 자동으로 시작됩니다. 이 변환 처리는 QFILESYS1 시스템 작업의 2차 스레드로 실행됩니다.

102 페이지의 『추가 정보: 독립 ASP』
독립 ASP의 사용자 정의 파일 시스템이 아직 *TYPE2 디렉토리 형식으로 변환되지 않은 경우 독립 ASP가 V5R2 이상의 오퍼레이팅 시스템이 설치된 시스템으로 처음 연결변환될 때 변환됩니다.

디렉토리 변환 시 고려사항

다음은 디렉토리 변환 프로세스 동안 고려해야 할 몇 가지 사항입니다.

변환 상태 결정:

i5/OS V5R3M0 이상의 릴리스가 설치된 후에 아직 변환되지 않은 파일 시스템에 대해 *TYPE2 디렉토리로의 변환이 자동으로 시작됩니다. 이 변환 처리는 QFILESYS1 시스템 작업의 2차 스레드로 실행됩니다.

변환 처리 상태를 판별하려면 다음과 같이 CVTDIR(디렉토리 변환) 명령을 사용할 수 있습니다.

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

CVTDIR 명령의 이 호출은 ″루트″(/), QOpenSys 및 UDFS 파일 시스템에 대한 현재 디렉토리 형식 및 파일 시스템이 현재 변환 중인지 나열합니다. 추가로 이 호출은 변환 기능의 현재 우선순위, 시스템에 의해 현재 변환되고 있는 파일 시스템, 그 파일 시스템을 위해 처리된 링크 수 및 그 파일 시스템을 위해 처리된 디렉토리의 비율을 나열합니다. 변환 기능이 시스템 활동에 현저하게 영향을 주지 않으므로 시스템은 매우 낮은 우선

순위(99)로 변환 기능을 시작합니다. 그러나 CVTDIR 명령의 OPTION 매개변수에 대한 *CHGPTY 값을 사용하여 변환 기능의 우선순위를 변경할 수 있습니다. 이 매개변수 스펙에 대한 추가 정보는 CVTDIR을 참조하십시오.

QFILESYS1 작업이 변환을 처리하고 있으므로 변환에 대한 모든 문제점을 나타내는 메세지에 대해 QFILESYS1 작업 기록부를 표시할 수 있습니다. 또한 여러 진행 메세지가 파일 시스템 변환에 대해 송신됩니다. 이 메세지는 변환되고 있는 파일 시스템, 파일 시스템에서 처리된 링크 수, 파일 시스템에서 처리된 디렉토리 비율 등의 정보를 포함합니다. 모든 오류 및 진행 메세지의 대부분이 QSYSOPR 메세지 대기행렬로도 송신됩니다. 그러므로 이것은 앞으로의 참조를 위해 QHST 기록부 또는 QFILESYS1 작업 로그가 이 메세지를 포함하여 보존되도록 하는 좋은 연습이 될 것입니다. 파일 시스템이 완전하게 변환되고 통합 파일 시스템이 예상만큼 작업되고 있으면 이 기록 정보를 삭제할 수 있습니다.

관련 정보

CVTDIR(디렉토리 변환) 명령

사용자 프로파일 작성:

변환 기능은 변환 기능이 실행될 때 사용될 사용자 프로파일을 작성합니다. 이 사용자 프로파일은 이름 QP0FCWA를 가집니다. 이 이름은 원래 소유자가 자신의 디렉토리를 소유할 수 없을 경우 파일 시스템의 디렉토리를 소유하기 위해 변환 기능에서 사용합니다.

사용자 프로파일은 변환이 완료된 후 가능하면 삭제됩니다. 디렉토리의 소유권이 이 사용자 프로파일에 지정되는 경우 메세지 CPIA08B가 QFILESYS1 작업 기록부 및 QSYSOPR 메세지 대기행렬로 송신됩니다.

관련 참조

100 페이지의『디렉토리 소유자 변경』

*TYPE1 디렉토리를 소유한 사용자 프로파일이 작성된 *TYPE2 디렉토리를 소유할 수 없는 경우 *TYPE2 디렉토리의 소유자가 대체 사용자 프로파일로 설정됩니다.

오브젝트 이름 변경:

*TYPE2 디렉토리의 링크명은 유효한 UTF-16 이름이어야 합니다.

* TYPE2 디렉토리의 명명 규칙은 UCS2 레벨 1 이름을 가지는 *TYPE1 디렉토리와 다릅니다. 이러한 이유로 디렉토리 변환 중에 유효하지 않거나 중복되는 이름이 발견될 수 있습니다. 이름이 유효하지 않거나 중복되는 경우, 해당 이름이 고유하고 유효한 UTF-16 이름으로 변경되며 메세지 CPIA08A가 QFILESYS1 작업 기록부 및 QSYSOPR 메세지 대기행렬로 송신되어 원래 이름과 새 이름을 나열합니다. 이름에 포함된 결합 문자나 유효하지 않은 대용 문자쌍으로 인해 오브젝트명이 변경될 수 있습니다.

UTF-16에 대한 자세한 정보는 유니코드(Unicode) 홈 페이지(www.unicode.org )를 참조하십시오.

결합 문자:

일부 문자는 둘 이상의 유니코드(Unicode) 문자로 구성될 수 있습니다.

예를 들어, 악센트가 있는 문자(예: é 또는 à) 또는 움라우트가 있는 문자(예: ä 또는 ö)는 디렉토리에 저장되기 전에 일반 형식으로 변경되거나 *표준화되어야* 모든 오브젝트가 고유한 이름을 가질 수 있습니다. 결합 문자를 표준화한다는 것은 문자를 알려지고 예측 가능한 형식으로 만드는 것입니다. *TYPE2 디렉토리에 대해 선택된 형식은 *표준 구성 형태*입니다. *TYPE1 디렉토리에 동일한 결합 문자가 포함된 두 개의 오브젝트가 있는 경우 둘 다 같은 이름으로 표준화됩니다. 한 오브젝트에 구성된 결합 문자가 포함되어 있고 다른 오브젝트에 분해된 결합 문자가 포함되어 있는 경우에도 이로 인해 충돌이 발생합니다. 따라서 *TYPE2 디렉토리로 링크하기 전에 둘 중 하나의 이름을 변경해야 합니다.

대용 문자:

일부 문자에는 유효한 유니코드(Unicode) 표현이 없습니다.

이러한 문자는 특수한 값을 가집니다. 이들 문자는 특정 범위의 두 개의 유니코드(Unicode) 문자로 구성되는데, 첫 번째 유니코드(Unicode) 문자는 첫 번째 범위(예: 0xD800-0xD8FF)에 속하며 두 번째 유니코드(Unicode) 문자는 두 번째 범위(예: 0xDC00-0xDCFF)에 속합니다. 이를 대체쌍이라고 합니다.

유니코드(Unicode) 문자 중 하나가 누락되거나 문자의 순서가 잘못 되면(일부 문자만) 이름은 유효하지 않습니다. 이러한 유형의 이름은 *TYPE1 디렉토리에서는 허용되지만 *TYPE2 디렉토리에서는 허용되지 않습니다. 유효하지 않은 이름이 포함된 이름이 발견되면 변환 기능을 계속 수행하기 위해 오브젝트가 *TYPE2 디렉토리로 링크되기 전에 해당 이름이 변경됩니다.

사용자 프로파일 고려사항:

변환이 실행되는 동안, *TYPE1 디렉토리를 소유한 동일한 사용자 프로파일이 대응하는 *TYPE2 디렉토리를 계속 소유할 수 있도록 모든 시도가 이루어집니다.

*TYPE1 및 *TYPE2 디렉토리는 잠시 동안 동시에 존재하게 되므로 이 시도는 사용자 프로파일이 소유한 기억장치의 양과 사용자 프로파일의 항목 수에 영향을 줍니다.

사용자 프로파일에 대한 최대 기억장치 변경:

디렉토리 변환 처리 중에 많은 수의 디렉토리가 동시에 두 형식으로 존재하며 같은 사용자 프로파일에서 이를 소유합니다.

변환 처리 중 사용자 프로파일에 대한 최대 기억장치 제한에 도달하면 사용자 프로파일에 대한 제한값이 증가합니다. 메세지 CPIA08C가 QFILESYS1 작업 기록부 및 QSYSOPR 메세지 대기행렬로 송신됩니다.

디렉토리 소유자 변경:

*TYPE1 디렉토리를 소유한 사용자 프로파일이 작성된 *TYPE2 디렉토리를 소유할 수 없는 경우 *TYPE2 디렉토리의 소유자가 대체 사용자 프로파일로 설정됩니다.

메세지 CPIA08B가 QFILESYS1 작업 기록부 및 QSYSOPR 메세지 대기행렬로 송신되고 변환이 계속됩니다.

*TYPE1 디렉토리를 소유한 사용자 프로파일이 작성된 *TYPE2 디렉토리를 소유할 수 없는 경우 *TYPE2 디렉토리의 소유자가 대체 사용자 프로파일로 설정됩니다. 메세지 CPIA08B가 QFILESYS1 작업 기록부 및 QSYSOPR 메세지 대기행렬로 송신되고 변환이 계속됩니다.

관련 참조

99 페이지의『사용자 프로파일 작성』

변환 기능은 변환 기능이 실행될 때 사용될 사용자 프로파일을 작성합니다. 이 사용자 프로파일은 이름 QP0FCWA를 가집니다. 이 이름은 원래 소유자가 자신의 디렉토리를 소유할 수 없을 경우 파일 시스템의 디렉토리를 소유하기 위해 변환 기능에서 사용합니다.

보조 기억장치 요구사항:

시스템이 파일 시스템의 디렉토리를 *TYPE2 형식으로 변환하는 동안 보조 기억장치 요구사항을 고려해야 합니다.

다음은 보조 기억장치 요구사항과 관련된 몇 가지 고려사항입니다.

- *TYPE2 형식으로 변환된 후 디렉토리의 최종 크기
- 변환 기능이 실행되는 동안 필요한 추가 기억장치

*TYPE2 디렉토리의 최종 크기가 *TYPE1 디렉토리보다 작은 경우가 많습니다. 일반적으로 350개 미만의 오브젝트가 포함된 *TYPE2 디렉토리는 같은 수의 오브젝트가 포함된 *TYPE1 디렉토리보다 필요한 보조 기억장치가 적습니다. 350개 이상의 오브젝트가 포함된 *TYPE2 디렉토리는 *TYPE1 디렉토리보다 평균적으로 10% 더 큽니다.

변환 기능이 실행되는 동안 추가 기억장치가 필요합니다. 변환 기능이 수행될 때 디렉토리가 *TYPE1 버전 및 *TYPE2 버전으로 동시에 존재할 수 있어야 합니다.

주: i5/OS V5R3M0 오퍼레이팅 시스템 또는 이후 릴리스를 설치하기 전에 OS/400 V5R2(CVTDIR) 명령에서 *ESTIMATE 옵션을 실행하도록 고려할 수 있습니다. 이것은 변환 중에 필요한 보조 기억장치 양의 예측값을 제공하기 때문입니다.

관련 정보

CVTDIR(디렉토리 변환) 명령

추가 정보: 기호 링크:

기호 링크는 다른 오브젝트로의 경로를 포함하는 통합 파일 시스템 내의 오브젝트입니다.

오브젝트명이 변경 가능한 경우 변환 중에 몇몇 인스턴스가 나타납니다. 변환 중에 기호 링크 내의 경로 요소명 중 하나가 변경되면 기호 링크의 내용은 더 이상 해당 오브젝트를 가리키지 않습니다.

관련 개념

12 페이지의 『링크』
*링크*는 디렉토리와 오브젝트 사이의 명명된 연결입니다. 사용자나 프로그램은 오브젝트에 대한 링크명을 지정하여 시스템에 오브젝트 위치를 알려 줄 수 있습니다. 링크는 경로명이나 경로명의 부분으로 사용할 수 있습니다.

관련 참조

99 페이지의 『오브젝트 이름 변경』
*TYPE2 디렉토리의 링크명은 유효한 UTF-16 이름이어야 합니다.

관련 정보

symlink()--기호 링크 작성

추가 정보: 독립 ASP:

독립 ASP의 사용자 정의 파일 시스템이 아직 *TYPE2 디렉토리 형식으로 변환되지 않은 경우 독립 ASP가 V5R2 이상의 오퍼레이팅 시스템이 설치된 시스템으로 처음 연결변환될 때 변환됩니다.

추가 정보: 저장 및 복원:

*TYPE1으로 존재하는 디렉토리는 *TYPE2로 변환된 파일 시스템에서 저장하거나 복원할 수 있습니다.

마찬가지로 디렉토리가 *TYPE2 디렉토리로 존재하는 경우 *TYPE1 제한을 초과하지 않으면 *TYPE2로 존재하는 디렉토리를 *TYPE1 형식의 파일 시스템에서 저장하거나 복원할 수 있습니다.

추가 정보: 통합 파일 시스템 오브젝트 재생:

시스템이 *TYPE2 디렉토리 형식을 지원하도록 ″루트″(/), QOpenSys 및 사용자 ASP UDFS 파일 시스템을 변환하는 동안 RCLSTG(기억장치 재생) 및 RCLLNK(오브젝트 링크 재생) 명령은 독립 ASP의 디렉토리를 포함한 모든 통합 파일 시스템 디렉토리에서 실행될 수 없습니다.

OMIT(*DIR) 매개변수 값은 RCLSTG 명령에서 통합 파일 시스템 디렉토리를 생략하고 통합 파일 시스템과 관련되지 않은 오브젝트가 재생되도록 하기 위해 사용할 수 있습니다.

관련 개념

113 페이지의 『″루트″ (/), QOpenSys 및 사용자 정의 파일 시스템의 재생 조작』
RCLLNK(오브젝트 링크 재생) 및 RCLSTG(기억장치 재생) 명령을 사용하여 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템 재생을 완료할 수 있습니다.

관련 정보

RCLSTG(기억장치 재생) 명령

RCLLNK(오브젝트 링크 재생) 명령

통합 파일 시스템 스캐닝:

″루트″(/), QOpenSys 및 사용자 ASP UDFS 파일 시스템의 오브젝트는 파일 시스템이 *TYPE2 디렉토리 형식으로 완전하게 변환될 때까지 통합 파일 시스템 스캔 관련 종료점을 사용하여 스캔되지 않습니다.

스캔 관련 속성은 파일 시스템이 완전하게 변환되지 않더라도 오브젝트가 스캔되는지 여부를 지정하도록 *TYPE1 및 *TYPE2 디렉토리에서 오브젝트를 설정할 수 있습니다.

시스템이 *TYPE1 디렉토리 형식에서 *TYPE2 디렉토리 형식으로 오브젝트를 변환하는 동안 변환되는 오브젝트가 복원되고 있는 것처럼 스캔 제어 시스템 값 오브젝트가 복원된 후 다음 액세스 시 스캔이 고려됩니다. 예를 들어, 변환이 진행 중인 동안 오브젝트가 복원된 후 다음 액세스 시 스캔 값이 지정되고 오브젝트가 *TYPE1 디렉토리에 있으며 오브젝트가 스캔되지 않음 속성이 지정되면 파일 시스템이 완전하게 변환된 이후에 최소 한 번 스캔됩니다.

관련 개념

20 페이지의『스캐닝 지원』

i5/OS 오퍼레이팅 시스템을 사용하여 통합 파일 시스템 오브젝트를 스캔할 수 있습니다.

22 페이지의『관련 시스템 값』

시스템에 대해 원하는 스캐닝 환경을 설정하기 위해 QSCANFS 및 QSCANFSCTL 시스템 값을 사용할 수 있습니다.

추가 문자 지원을 위한 이름 변환

파일 시스템은 유니코드로 이름을 저장합니다. 대소문자를 구분하지 않는 파일 시스템은 특정 유니코드 표준의 문자 또는 대소문자 구분 규칙 변경으로 인해 영향을 받을 수 있습니다. i5/OS V6R1부터 ""루트""(/)와 같이 대소문자 구분을 하지 않는 파일 시스템 및 CASE(*MONO)로 작성된 사용자 정의 파일 시스템(UDFS)은 유니코드 표준 4.0을 지원합니다.

i5/OS가 설치된 직후에 유니코드 표준을 지원하도록 변환되지 않은 모든 파일 시스템을 대상으로 디렉토리를 유니코드 표준 4.0으로 변환하는 작업이 자동 시작됩니다. 이 변환은 낮은 우선순위 백그라운드 작업으로 실행되기 때문에 시스템 활동에 큰 영향을 주지 않습니다.

관련 정보

http://www.unicode.org

자동 이름 변환 개요

""루트""(/)와 같이 대소문자 구분을 하지 않는 파일 시스템 및 CASE(*MONO)로 생성된 UDFS는 유니코드 표준 4.0으로 저장된 이름을 지원합니다. 시스템은 이름에 포함된 추가 문자를 지원하기 위해 자동 이름 변환을 실행합니다.

통합 파일 시스템에서 이름 검색은 처리되고 있는 오브젝트를 식별하기 위해 통합 파일 시스템 CL 명령 또는 API에 이름이 전달된 경우에 발생합니다. 새로운 문자 지원을 위해 새로운 문자가 정의되고 대소문자 규칙이 갱신된 경우 이 변경사항으로 영향을 받는 문자를 포함하고 있는 이름은 더 이상 검색되지 않을 수 있습니다.

통합 파일 시스템은 유니코드로 모든 이름을 저장합니다. i5/OS V6R1 이전 통합 파일 시스템은 유니코드 표준 2.0을 지원합니다. V6R1부터 통합 파일 시스템은 유니코드 표준 4.0을 지원합니다. 이름 변환 유틸리티는

유니코드 표준 4.0 지원을 위해 대소문자을 구분하지 않는 파일 시스템 내 디렉토리를 자동 갱신합니다. 대소문자를 구분하지 않고 이 변환의 대상이 되는 파일 시스템은 ""루트""(/) 및 CASE(*MONO)로 작성된 사용자 정의 파일 시스템(보조 기억장치 풀에 있는)입니다.

""루트""(/) 및 기본 사용자 보조 기억장치 풀에 있는 UDFS(1-32)에 있는 이름의 변환은 i5/OS 오퍼레이팅 시스템이 설치되어 QFILESYS1 시스템 작업의 낮은 우선순위 스레드에서 실행된 후 자동 시작됩니다. 변환 기능이 완료되지 않은 상태에서 시스템이 초기 프로그램 로드(IPL)을 수행한 경우 변환 기능은 IPL이 완료된 후 다시 시작됩니다. 모든 적합한 파일 시스템이 완전하게 변환될 때까지 변환은 IPL마다 재시작합니다.

독립 ASP의 연결변환 작업 중 독립 디스크 풀 그룹 #####용 QFSYS##### 작업은 자동적으로 시작되고 독립 ASP #####의 UDFS 내 이름 변환은 이 시스템 작업의 낮은 우선순위 스레드에서 실행됩니다. 독립 ASP 내 UDFS의 이름 변환은 기본 사용자 ASP 내에 있는 ""루트""(/) 및 UDFS의 이름 변경이 완료되기 전까지 디렉토리 변환을 시작하지 않습니다. 변환 기능이 완료되지 않은 상태에서 독립 ASP가 단절변환된 경우 또는 시스템이 IPL을 수행한 경우 변환 기능은 IPL이 완료되고 독립 ASP가 연결변환한 후 재개됩니다.

주: 하나 이상의 독립 ASP가 동시에 연결변환하는 경우 매 순간 하나의 독립 ASP 이름 변환만이 활성화됩니다. 일단 한 변환이 완료되면 다음 변환이 활성화됩니다.

이전 릴리스에서는 이러한 이름 변환을 계획하기 위해 Analyze Object Conversion (ANZOBJCVN) 명령을 사용할 수 있습니다. ANZOBJCVN 명령은 새 릴리스의 새 유니코드 문자 및 대소문자 구분 규칙에 영향을 받을 수 있는 문자를 1개 이상 포함하고 있는 통합 파일 시스템 오브젝트명을 포함한 시스템에 있는 오브젝트에 대한 변환 정보를 수집 또는 보고합니다. 영향을 받는 오브젝트는 새 릴리스로 업그레이드된 후 수행되는 자동 변환에서 새 유니코드 문자로 이름 변경될 수 있습니다. 변경된 오브젝트 이름이 어플리케이션에 영향을 미칠 수 있기 때문에 새 릴리스로 업그레이드하기 전에 영향을 받는 오브젝트를 이름 변환하고자 할 수도 있습니다.

관련 개념

18 페이지의 『이름 연속성』

『루트』(/), QOpenSys 및 사용자 정의 파일 시스템을 사용할 때 오브젝명의 문자가 동일하게 남아 있도록 하는 시스템 지원을 이용할 수 있습니다.

관련 정보

http://www.unicode.org

권장됨: 오브젝트 변환을 위한 PTF 설치 및 시스템 분석

통합 파일 시스템 보안 계획

이름 변경 시 고려사항

다음은 자동 이름 변환 프로세스 동안 고려해야 할 몇 가지 사항입니다.

변환 상태 결정:

어떤 파일 시스템이 변환되었는지를 표시하는 일반 메세지 및 다양한 진행 상황 메세지는 변환을 수행하는 작업의 작업 로그로 전송됩니다.

| ""루트""(/) 및 ASP 1-32의 UDFS에서 이 메세지는 QFIESYS1 작업 로그로 전송됩니다. 독립 ASP의 UDFS
| 에서 이 메세지는 독립 디스크 풀 그룹 #####의 QFSYS##### 작업의 작업 로그로 전송됩니다. 이 메세지는
| 변환되고 있는 파일 시스템 및 파일 시스템에서 처리된 링크 수 등의 정보를 포함합니다. 일반 메세지 중 다수
| 가 QSYSOPR 메세지 대기행렬로도 송신됩니다. 오류 메세지는 QHST 이력 로그로 전송됩니다.

| 그러므로 이것은 앞으로의 참조를 위해 QHST 로그가 이 메세지를 포함하여 보존되도록 하는 좋은 연습이 될
| 것입니다. 파일 시스템이 완전하게 변환되고 통합 파일 시스템이 예상만큼 작업되고 있으면 이 기록 정보를 삭
| 제할 수 있습니다.

| **오브젝트 이름 변경:**

| 대소문자를 구분 안하는 파일 시스템에서 시스템에서 이름을 탐색하는 동안 대소문자를 구분하지 않습니다. 대
| 소문자 구분 규칙의 변화는 대소문자를 구분하지 않는 경우 동일한 문자로 간주되는 문자에 영향을 줍니다.

| 자동 이름 변환 중 유니코드 표준 2.0에서 동일하게 간주되지 않는 이름은 유니코드 표준 4.0에서는 동일하게
| 간주될 수 있습니다. 그래서 이 오브젝트 중 하나의 이름은 변경되어야 합니다. 고유한 이름으로 변경할 때 기
| 존 이름 및 새 이름을 포함하고 있는 CPDAOBC 메세지가 QHST로 전송됩니다.

| 다른 버전에 대한 자세한 정보는 유니코드 홈 페이지(www.unicode.org)를 참조하십시오.

| **사용자 프로파일 고려사항:**

| 변환을 실행하는 중 시스템은 해당 사용자 프로파일이 소유하는 기억장치의 양에 영향을 주는 추가 기억장치
| 가 필요합니다.

| 이름 변환 처리 중 사용자 프로파일의 최대 기억장치 제한에 도달하면 시스템은 해당 사용자 프로파일에서 그
| 제한값을 증가시키고 QSYSOPR 메세지 대기열에 CPIAO8C 메세지를 송신합니다.

| **추가 정보: 기호 링크:**

| 기호 링크는 다른 오브젝트의 경로명을 포함하는 통합 파일 시스템 내의 오브젝트입니다. 변환 중에 기호 링크
| 내의 경로 요소명 중 하나가 변경되면 기호 링크의 내용은 더 이상 다른 오브젝트를 가리키지 않습니다.

| **추가 정보: 독립 ASP:**

| 독립 ASP의 사용자 정의 파일 시스템이 현재 릴리스의 유니코드 표준 버전을 사용하도록 변환되지 않은 경우
| 독립 ASP가 연결변환될 때 이 변환을 수행하는 작업이 시작됩니다.

| 시스템은 독립 디스크 풀 그룹 #####에 대해 QFSYS##### 작업을 시작합니다. 자동 이름 변환은 낮은 우선
| 순위 스레드에서 실행되며 해당 그룹 내 각 독립 보조 기억장치 풀마다 하나의 스레드가 시작됩니다. 시스템은
| 자동 이름 변환이 완료되지 않은 상태에서 독립 디스크 풀이 연결변환될 때마다 이 작업을 시작합니다.

| **추가 정보: 저장 및 복원:**

| 유니코드 표준 2.0을 사용하는 디렉토리를 저장하여 유니코드 표준 4.0을 사용하는 파일 시스템에서 복원할
| 수 있으며 그 역 과정도 가능합니다.

| 오브젝트가 복원될 때 그것은 하나의 디렉토리로 링크됩니다. 오브젝트가 링크되고 있는 디렉토리의 유니코드 표준이 복원되는 오브젝트의 이름에 적용됩니다. 그래서 해당 디렉토리에 사용되는 유니코드 표준에서 오브젝트의 이름이 고유하지 않다면 그 오브젝트는 복원될 수 없습니다. 이 경우 시스템은 원인 코드 1과 함께 CPD37B9 메세지를 작업 로그로 전송합니다.

추가 정보: 통합 파일 시스템 오브젝트 재생:

자동 이름 변환 중 오류가 발생한다면 스토리지 재생(RCLSTG) 및 오브젝트 링크 재생(RCLLNK) 명령은 해당 디렉토리 및 그것의 하위 디렉토리의 변환을 완료할 수 있습니다. 이것은 해당 파일 시스템에서 자동 이름 변환 처리가 완료된 후에만 발생합니다.

또한, RCLSTG 명령은 모든 유실된 디렉토리를 유니코드 표준 4.0을 사용하도록 변환합니다.

관련 개념

113 페이지의 『″루트″ (/), QOpenSys 및 사용자 정의 파일 시스템의 재생 조작』

RCLLNK(오브젝트 링크 재생) 및 RCLSTG(기억장치 재생) 명령을 사용하여 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템 재생을 완료할 수 있습니다.

관련 정보

RCLSTG(기억장치 재생) 명령

RCLLNK(오브젝트 링크 재생) 명령

오브젝트 저널링

저널링의 1차 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 복구할 수 있도록 하는 것입니다. 또한 저널링의 주요 사용은 고가용성 또는 작업부하 균형 조절을 위해 다른 시스템에 오브젝트 변경사항을 복제할 때 도움을 주기 위한 것입니다.

이 정보는 저널 관리에 대한 간략한 개요를 제공하고 통합 파일 시스템 오브젝트 저널링을 위한 고려사항을 제공하며 통합 파일 시스템 오브젝트의 저널링 지원에 대한 설명도 제공합니다.

관련 정보

저널 관리

저널링 개요

이 주제에서는 통합 파일 시스템 오브젝트에 대한 저널링 지원을 소개합니다.

관련 정보

저널 관리

저널 관리

저널 관리의 기본 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 복구할 수 있도록 하는 것입니다.

다음에 대해 저널 관리를 사용할 수도 있습니다.

- 시스템에서 오브젝트에 대해 발생하는 활동에 대한 감사 추적
- 저널링할 수 없는 오브젝트를 제외한 다른 오브젝트에 대해 발생한 활동 기록
- 활동 중 저장 매체에서 복원시 보다 빠른 복구
- 고가용성 또는 작업부하 균형 조절을 위해 다른 시스템에 오브젝트 변경사항 복제 지원
- 어플리케이션 프로그램 테스트 지원

저널을 사용하여 저널 관리로 보호하려는 오브젝트를 정의할 수 있습니다. 통합 파일 시스템에서 스트림 파일, 디렉토리 및 기호 링크를 저널링할 수 있습니다. ″루트″(/), QOpenSys, UDFS 파일 시스템의 오브젝트만 지원됩니다.

관련 개념

『저널링해야 할 오브젝트』

통합 파일 시스템 오브젝트를 저널링할지 여부를 결정할 때 고려해야 할 몇 가지 질문이 있습니다.

저널링해야 할 오브젝트

통합 파일 시스템 오브젝트를 저널링할지 여부를 결정할 때 고려해야 할 몇 가지 질문이 있습니다.

저널링해야 하는 오브젝트를 판별하려면 다음 질문을 고려해야 합니다.

- 오브젝트는 얼마나 많이 변경됩니까? 저장 조작 사이에 변경사항이 많은 오브젝트는 저널링하는 것이 좋습니다.
- 오브젝트의 변경사항을 재구성하는 것이 어렵습니까? 기록된 레코드가 없는 오브젝트에 변경사항이 많습니까? 예를 들면, 전화 주문 항목에 사용된 오브젝트는 메일로 주문 양식이 도착하는 주문에 사용된 오브젝트보다 재구성하기 어렵습니다.
- 오브젝트의 정보는 얼마나 중요합니까? 오브젝트가 마지막 저장 조작으로 다시 복원되어야 하는 경우, 변경사항 재구성이 지연되면 업무상 어떤 영향이 있습니까?
- 오브젝트는 시스템의 기타 오브젝트와 얼마나 관련이 있습니까? 특정 오브젝트의 데이터가 자주 변경되지 않더라도 해당 오브젝트의 데이터가 시스템에 있는 더욱 동적인 다른 오브젝트에 비해 중요할 수 있습니다. 예를 들면, 고객의 마스터 파일에 의존하는 오브젝트가 많습니다. 주문을 재구성하는 경우, 고객 마스터 파일에는 이전에 저장 조작한 이후에 발생한 신용 한도 변경사항 또는 새로운 고객이 포함되어야 합니다.

저널링된 통합 파일 시스템 오브젝트

일부 통합 파일 시스템 오브젝트 유형은 i5/OS 저널링 지원을 사용하여 저널링할 수 있습니다.

지원되는 오브젝트 유형은 스트림 파일, 디렉토리 및 기호 링크입니다. ″루트″(/), QOpenSys, UDFS 파일 시스템만 이런 오브젝트 유형을 지원합니다. 일반적인 시스템 인터페이스(CL 명령 또는 API)나 System i Navigator를 사용하여 통합 파일 시스템 오브젝트를 저널링할 수 있습니다. 저널링 상태 정보를 표시하고 System i Navigator를 통해 저널링을 시작하고 종료할 수 있습니다.

주: 메모리 맵핑된 스트림 파일인 가상 볼륨 파일 및 통합 xSeries 서버(IXS) 네트워크 저장 공간으로 사용되는 스트림 파일은 저널링될 수 없습니다. 블록 특수 파일(*BLKSF) 오브젝트를 포함할 수 있는 디렉토리는 저널링될 수 없습니다. 이 예는 /dev/QASP01, /dev/QASP22 및 /dev/IASPNAME입니다.

다음 리스트는 통합 파일 시스템의 저널링 지원을 요약한 것입니다.

- 총칭 명령과 API 모두를 사용하여 지원된 오브젝트 유형에 대해 저널 조작을 수행할 수 있습니다. 이러한 인터페이스들은 일반적으로 경로명, 파일 ID 또는 두 가지 양식으로 오브젝트 식별을 허용합니다.
- 통합 파일 시스템 오브젝트의 전체 서브트리에 대해 저널 시작, 저널 종료, 저널된 오브젝트 변경 및 저널된 변경사항 적용을 포함하여 몇 가지 저널 조작 명령을 수행할 수 있습니다. 선택적으로, 오브젝트명에 와일드 카드 패턴을 사용하는 포함 및 제외 리스트를 사용할 수 있습니다. 예를 들어, 저널 시작 명령을 사용하여 `"*.data"` 패턴과 일치하는 `"/MyCompany"` 트리의 모든 오브젝트에 대해 저널을 시작하지만 `"A*.data"` 및 `"B*.data"` 패턴과 일치하는 모든 오브젝트를 제외하도록 지정할 수 있습니다.
- 디렉토리에 대한 저널링 지원에는 디렉토리 내에서 링크 추가, 링크 삭제, 오브젝트 작성, 오브젝트 이름 변경, 오브젝트 이동 등의 디렉토리 조작이 포함됩니다.

  저널링된 디렉토리는 서브트리의 신규 오브젝트가 디렉토리의 현재 저널링 상태를 계승하도록 설정할 수 있는 속성을 지원합니다. 저널링된 디렉토리에 대해 이 속성이 작동되면, 시스템은 작성되거나 디렉토리로 링크(하드 링크 추가, 오브젝트 이름 변경 또는 이동)된 모든 스트림 파일, 디렉토리, 기호 링크에 대해 자동으로 저널링을 시작합니다.

  주: 상속의 저널링 속성 고려사항:

  - 현재 같은 디렉토리에 상주하는 오브젝트명을 변경하면, 디렉토리가 상속의 현재 저널링 상태 속성을 갖는다 하더라도 오브젝트에 대해 저널링을 시작할 수 없습니다.
  - 디렉토리가 상속의 저널링 속성을 가진 디렉토리로 이동되면 이동되는 디렉토리는 적절하게 오브젝트에 대해 저널링을 시작합니다. 이동되는 디렉토리 내의 오브젝트는 영향을 받지 않습니다.
  - 상속의 저널링 속성을 가진 디렉토리로 오브젝트가 복원되는 경우 오브젝트가 저널되지 않았으면 그 오브젝트에 대해 저널링이 시작되지 않습니다.
  - APYJRNCHG(저널링된 변경사항 적용) 명령을 사용하면 모든 디렉토리에 대한 상속 저널링 속성의 현재 값은 사용되지 않습니다. 대신, 적용의 일부로 작성된 모든 오브젝트는 저널링을 시작하거나 적용되고 있는 런타임 활동 중에 발생한 것에 기반하지 않습니다.
- 오브젝트명 및 전체 경로명이 몇 개의 통합 파일 시스템 오브젝트 저널 항목 내에 포함됩니다. 오브젝트명과 경로명은 자국어 지원(NLS)이 가능합니다.
- 시스템이 비정상적으로 종료되면 저널링된 통합 파일 시스템 오브젝트에 시스템 초기 프로그램 로드(IPL) 복구가 제공됩니다.
- 다양한 쓰기 인터페이스에서 지원되는 최대 쓰기 제한은 2GB - 1입니다. RCVSIZOPT(*MAXOPT2 또는 *MAXOPT3)를 지정한 경우 4 000 000 000바이트입니다. 지정하지 않은 경우 최대 저널 항목 크기는 15 761 440바이트입니다. 스트림 파일을 저널링하고 15 761 440바이트를 초과하는 쓰기가 발생한 경우 *MAXOPT2 또는 *MAXOPT3 지원을 사용하여 오류가 발생하지 않도록 할 수 있습니다.

다양한 저널 항목 배치에 대한 자세한 정보는 QSYSINC/H(QP0LJRNL) 멤버에 제공된 C 언어 포함 파일 qp0ljrnl.h에 있으며 여기에는 통합 파일 시스템 저널 항목별 데이터 내용과 형식 세부사항이 들어 있습니다.

관련 개념

16 페이지의 『스트림 파일』
*스트림 파일*은 단순하게 바이트가 나열된 구조로 랜덤 액세스가 가능합니다.

5 페이지의 『디렉토리』
*디렉토리*는 사용자가 지정한 이름으로 오브젝트를 찾는 데 사용되는 특수 오브젝트입니다. 각 디렉토리에는 이에 연결된 오브젝트 리스트가 들어 있습니다. 해당 리스트에 다른 디렉토리가 포함될 수 있습니다.

14 페이지의 『기호 링크』
소프트 링크라고도 하는 *기호 링크*는 파일에 포함된 경로명입니다.

관련 태스크

111 페이지의 『저널링 시작』
저널링을 시작하려면 System i Navigator를 통한 오브젝트에 다음 단계를 수행하십시오.

112 페이지의 『저널링 종료』
오브젝트에서 저널링을 시작한 이후 어떤 이유로든 이 오브젝트의 저널링을 종료하려는 경우 이 주제에 설명된 단계를 사용할 수 있습니다.

112 페이지의 『저널링 변경』
오브젝트에서 저널링을 시작한 이후 어떤 이유로든 저널링을 종료하고 재시작하지 않고 오브젝트에서 저널링 속성을 변경하려고 하는 경우에 저널링된 오브젝트를 변경할 수 있게 하는 CHGJRNOBJ(저널링된 오브젝트 변경) 명령을 사용할 수 있습니다.

관련 정보

저널 관리

저널 항목 정보 파인더

저널링된 조작

다음 조작은 조작이 사용되고 있는 오브젝트나 링크의 유형이 저널링될 수 있는 유형인 경우에만 저널링됩니다.

- 오브젝트 작성
- 기존 오브젝트에 링크 추가
- 링크 해제
- 링크 이름 변경
- 파일 ID 이름 변경
- 디렉토리 간에 링크 이동

다음의 저널링된 조작은 스트림 파일에만 해당됩니다.

- 데이터 쓰기 또는 지우기
- 파일 절단/확장
- 파일 데이터 강제
- 기억장치를 해제하여 저장

다음의 저널링된 조작은 저널링된 모든 오브젝트 유형에 적용됩니다.

- 속성 변경(권한 및 소유권과 같은 보안 변경사항 포함)
- 열기
- 닫기
- 저널링 시작
- CHGJRNOBJ(저널링된 오브젝트 변경) 명령
- 저널링 종료
- APYJRNCHG(저널링된 변경사항 적용) 명령 시작
- APYJRNCHG(저널링된 변경사항 적용) 명령 종료
- 저장
- 복원

관련 정보

저널 관리

저널 항목 정보 파인더

저널 항목에 대한 특수 고려사항

다수의 저널링된 통합 파일 시스템 조작이 내부적으로 확약 제어를 사용하여 조작 중에 수행된 복수 기능으로부터 단일 트랜잭션을 형성합니다.

이 저널링된 조작은 확약 제어 주기에 확약 저널 항목(저널 코드 C, 유형 CM)이 없으면, 완전한 것으로 간주될 수 없습니다. 확약 제어 주기에 롤백 저널 항목(저널 코드 C, 유형 RB)이 포함된 저널링된 조작은 실패한 조작이며, 이 조작 내의 저널 항목이 복제되거나 반복되어서는 안됩니다.

이런 방법으로 확약 제어를 사용하는 저널링된 통합 파일 시스템 항목(저널 코드 B)은 다음과 같습니다.

- AA — 감사 값 변경
- B0 — 작성 시작
- B1 — 요약 작성
- B2 — 링크 추가
- B3 — 이름 변경/이동
- B4 — 링크 해제(상위 디렉토리)
- B5 — 링크 해제(링크)
- B7 — 작성된 오브젝트 권한 정보
- FA — 속성 변경
- JT - 저널 시작(상속 저널링 속성이 '예'인 디렉토리의 조작으로 저널링이 시작되는 경우에만)
- OA — 권한 변경
- OG — 오브젝트 1차 그룹 변경

• OO — 오브젝트 소유자 변경

몇 개의 통합 파일 시스템 저널 항목에 항목이 요약 항목인지 여부를 표시하는 특정 데이터 필드가 있습니다. 요약 항목 유형을 송신하는 조작은 두 가지의 동일한 유형의 항목을 저널로 송신합니다. 첫 번째 항목에는 입력 항목별 데이터 서브세트가 들어 있습니다. 두 번째 항목에는 전체 입력 항목별 데이터가 들어 있어 요약 항목임을 표시합니다. 오브젝트 복제 또는 조작을 반복하는 프로그램은 일반적으로 요약 항목에만 관계 있습니다.

저널링된 디렉토리에서 조작을 작성하는 경우, B1 저널 항목(요약 작성)을 요약 항목으로 간주합니다.

일부 저널링된 조작은 조작과 역으로 관련된 저널 항목을 송신해야 합니다. 예를 들면, B4 저널 항목(링크 해제)이 들어 있는 확약 제어 주기에도 B2 저널 항목(링크 추가)이 포함될 수 있습니다. 이 시나리오 유형은 롤백 저널 항목(C -- RB)이 초래되는 조작에서만 발생합니다.

이 시나리오가 발생하는 이유는 다음 두 가지입니다.

1. 조작이 막 실패하였고 오류 경로 클린업을 위해 내부적으로 항목이 필요합니다.
2. 시스템 중지로 인해 조작이 인터럽트되었으며, 후속 IPL 동안 인터럽트된 조작을 롤백하기 위해 항목 송신에 필요한 복구가 수행되었습니다.

관련 정보

저널 항목 정보 파인더

여러 하드 링크 및 저널링 고려사항

저널링된 통한 파일 시스템 오브젝트와의 여러 하드 링크를 가지고 있으면 연관된 저널 정보뿐만 아니라 연계가 보존되므로 모든 링크는 함께 저장되고 복원되어야 합니다.

저널 관련 명령의 일부에 이름을 지정하고 이름이 실제로 여러 하드 링크이면 오브젝트는 '한 번만' 조작됩니다. 다른 하드 링크는 근본적으로 무시됩니다.

여러 하드 링크가 같은 오브젝트를 가리키고 저널 항목에 오브젝트에 대해 같은 파일 ID만 있으므로 경로명을 표시하는 저널 인터페이스(예: 저널 표시(DSPJRN))는 오브젝트에 대해 하나의 링크명만 표시합니다. 그러나 하나의 링크가 임의의 이름으로 오브젝트에 대해 조작하고 같은 결과를 가져올 수 있으므로 이것이 문제점을 유발해서는 안됩니다.

관련 개념

13 페이지의『하드 링크』

하드 링크는 때때로 단지 링크라고만 하며 실제 오브젝트로 링크되지 않는 한 존재할 수 없습니다.

저널링 시작

저널링을 시작하려면 System i Navigator를 통한 오브젝트에 다음 단계를 수행하십시오.

1. **System i Navigator**에서 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. 저널하려는 오브젝트를 오른쪽 마우스 버튼으로 클릭한 다음, **저널링**을 선택하십시오.

4. 적절한 저널링 옵션을 선택한 다음, **시작**을 클릭하십시오.

문자 기반 인터페이스를 통해 오브젝트에서 저널링을 시작하려면 STRJRN(저널 시작) 명령 또는 QjoStartJournal API를 사용할 수 있습니다.

관련 정보

STRJRN(저널 시작) 명령

저널 시작(QjoStartJournal) API

저널 관리

저널링 변경

오브젝트에서 저널링을 시작한 이후 어떤 이유로든 저널링을 종료하고 재시작하지 않고 오브젝트에서 저널링 속성을 변경하려고 하는 경우에 저널링된 오브젝트를 변경할 수 있게 하는 CHGJRNOBJ(저널링된 오브젝트 변경) 명령을 사용할 수 있습니다.

관련 태스크

111 페이지의 『저널링 시작』

저널링을 시작하려면 System i Navigator를 통한 오브젝트에 다음 단계를 수행하십시오.

『저널링 종료』

오브젝트에서 저널링을 시작한 이후 어떤 이유로든 이 오브젝트의 저널링을 종료하려는 경우 이 주제에 설명된 단계를 사용할 수 있습니다.

관련 정보

CHGJRNOBJ(저널링된 오브젝트 변경) 명령

저널링 종료

오브젝트에서 저널링을 시작한 이후 어떤 이유로든 이 오브젝트의 저널링을 종료하려는 경우 이 주제에 설명된 단계를 사용할 수 있습니다.

System i Navigator를 통해 오브젝트의 저널링을 종료하려면 다음 단계를 수행하십시오.

1. **System i Navigator**에서 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. 저널링을 중지하려는 오브젝트를 오른쪽 마우스 버튼으로 클릭한 다음, **저널링**을 선택하십시오.
4. **종료**를 클릭하십시오.

문자 기반 인터페이스를 통해 오브젝트의 저널링을 종료하려면 ENDJRN(저널 종료) 명령 또는 QjoEndJournal API를 사용할 수 있습니다.

관련 태스크

111 페이지의 『저널링 시작』

저널링을 시작하려면 System i Navigator를 통한 오브젝트에 다음 단계를 수행하십시오.

관련 정보

ENDJRN(저널 종료) 명령

저널 종료(QjoEndJournal) API

저널 관리

″루트″ (/), QOpenSys 및 사용자 정의 파일 시스템의 재생 조작

RCLLNK(오브젝트 링크 재생) 및 RCLSTG(기억장치 재생) 명령을 사용하여 ″루트″(/), QOpenSys 및 사용자 정의 파일 시스템 재생을 완료할 수 있습니다.

RCLLNK 및 RCLSTG 명령을 사용하여 다음 타스크를 수행할 수 있습니다.

- 오브젝트 사용자 프로파일 문제점 정정
- 사용자 정의 파일 시스템 문제점 정정
- 내부 오브젝트 문제점 정정
- 유효하지 않은 오브젝트 링크 제거
- 손상된 오브젝트 처리
- 누락된 시스템 오브젝트 작성
- 내부 파일 시스템 문제점 정정(RCLSTG만)
- 유실된 오브젝트 찾기(RCLSTG만)
- 자동 이름 변환 중 오류가 발생했을 경우 유니코드 표준 4.0을 사용하려면 디렉토리 변환을 완료하십시오.

RCLLNK(오브젝트 링크 재생) 및 RCLSTG(기억장치 재생) 명령 비교

RCLLNK(오브젝트 링크 재생) 및 기억장치 재생(RCLSTG) 명령 모두를 사용하여 『루트』(/), QOpenSys 및 사용자 정의 파일 시스템의 문제점을 정정할 수 있습니다.

RCLLNK 명령은 사용 중인 마운트된 파일 시스템에서 문제점을 식별하고 가능한 경우 문제점을 정정합니다. RCLSTG 명령은 이 기능을 포함하지 않습니다. 그러나 RCLSTG 명령은 RCLLNK 명령이 식별하거나 정정할 수 없는 문제점을 정정할 수 있습니다. 다음 표에서는 두 명령 간의 차이에 대한 추가 정보를 제공합니다.

표 11. RCLLNK 및 RCLSTG 명령 비교

	RCLLNK OBJ('/MyDir/MyObj')	RCLSTG ASPDEV(*SYSBAS)	RCLSTG ASPDEV(<IASPNAME>)
시스템이 제한 상태에 있어야 합니까?	아니오	예	아니오
재생 조작 중에 모든 파일 시스템을 사용할 수 있습니까?	예	아니오	재생 중인 독립 ASP의 파일 시스템은 사용할 수 없습니다.
어떤 ASP의 오브젝트가 재생될 수 있습니까?	시스템, 사용자 및 독립 ASP의 오브젝트를 재생합니다.	시스템 및 사용자 ASP의 오브젝트를 재생합니다.	독립 ASP의 오브젝트를 재생합니다.
어떻게 오브젝트가 재생됩니까?	오브젝트가 명령에 지정된 대로 개별적 또는 서브트리 기반으로 재생됩니다.	오브젝트가 시스템 기반으로 재생됩니다.	오브젝트가 독립 ASP 기반으로 재생됩니다.

표 11. RCLLNK 및 RCLSTG 명령 비교 (계속)

	RCLLNK OBJ('/MyDir/MyObj')	RCLSTG ASPDEV(*SYSBAS)	RCLSTG ASPDEV(<IASPNAME>)
가능한 경우 식별되고 정정되는 알려진 적용 가능한 파일 시스템 문제점은 무엇입니까?	대부분(자세한 정보는 113 페이지의 『″루트″ (/), QOpenSys 및 사용자 정의 파일 시스템의 재생 조작』을 참조하십시오.)	모두	모두
유실된 오브젝트가 발견되었습니까?	아니오	예	예
마운트 해제된 파일 시스템의 오브젝트가 재생됩니까?	아니오	예	예
명령이 스레드세이프입니까?	예	아니오	아니오
동시에 수행될 수 있는 명령의 인스턴스 수는 얼마입니까?	복수 인스턴스	단일 인스턴스	단일 인스턴스
필요한 경우 다시 작성되는 적용 가능한 통합 파일 시스템 제공 오브젝트는 무엇입니까?	모두	대부분 (자세한 정보는 115 페이지의 『통합 파일 시스템 제공 오브젝트 다시 작성』을 참조하십시오.)	없음
재생하지 않고 손상된 오브젝트를 식별할 수 있습니까?	예	아니오	아니오

관련 개념

115 페이지의 『예: RCLLNK(오브젝트 링크 재생) 명령』
이 예는 RCLLNK(오브젝트 링크 재생) 명령을 사용하여 ″루트″(/), QOpenSys 및 마운트된 사용자 정의 파일 시스템의 오브젝트를 복구할 수 있는 상황에 대해 설명합니다.

관련 참조

115 페이지의 『통합 파일 시스템 제공 오브젝트 다시 작성』
이 표는 존재하지 않는 경우 RCLLNK(오브젝트 링크 재생) 명령으로 다시 작성하는 통합 파일 시스템 제공 오브젝트를 표시합니다. 보통 이러한 오브젝트는 초기 프로그램 로드(IPL) 중에 작성됩니다. 필요한 경우 RCLSTG(기억장치 재생) 명령을 사용하여 이러한 오브젝트 중 일부를 다시 작성할 수도 있습니다.

관련 정보

RCLSTG(기억장치 재생) 명령

RCLLNK(오브젝트 링크 재생) 명령

RCLLNK(오브젝트 링크 재생) 명령

RCLLNK(오브젝트 링크 재생) 명령은 시스템을 제한 상태에 두지 않고 『루트』(/), QOpenSys의 손상된 오브젝트 및 마운트된 사용자 정의 파일 시스템을 식별하고 복구합니다. 이것은 생산성을 감소시키지 않고 파일 시스템의 문제점을 정정할 수 있습니다.

RCLLNK 명령은 대부분 상황에서 RCLSTG(기억장치 재생) 명령 대신으로 사용될 수 있습니다. 예를 들어, RCLLNK는 다음 상황에서 문제점을 식별하고 정정하는 데 이상적입니다.

- 문제점이 단일 오브젝트로 분리되었습니다.
- 문제점이 오브젝트 그룹으로 분리되었습니다.

- 손상된 오브젝트가 식별되거나 삭제되어야 합니다.
- 재생 조작 중에 시스템이 제한 상태에 있을 수 없습니다.
- 재생 조작 중에 독립 ASP를 사용할 수 있어야 합니다.

통합 파일 시스템 제공 오브젝트 다시 작성

이 표는 존재하지 않는 경우 RCLLNK(오브젝트 링크 재생) 명령으로 다시 작성하는 통합 파일 시스템 제공 오브젝트를 표시합니다. 보통 이러한 오브젝트는 초기 프로그램 로드(IPL) 중에 작성됩니다. 필요한 경우 RCLSTG(기억장치 재생) 명령을 사용하여 이러한 오브젝트 중 일부를 다시 작성할 수도 있습니다.

표 12. 통합 파일 시스템에서 제공하고 RCLLNK 및 RCLSTG 명령으로 다시 작성하는 오브젝트

경로명	유형	RCLLNK로 다시 작성됨	RCLSTG ASPDEV(*SYSBASE)로 다시 작성됨
/dev/zero	*CHRSF	예	예
/dev/null	*CHRSF	예	예
/dev/xti/tcp	*CHRSF	예	아니오
/dev/xti/udp	*CHRSF	예	아니오
/etc/vfs	*STMF	예	아니오

RCLLNK 명령이 존재하지 않는 통합 파일 시스템에서 제공된 오브젝트를 다시 작성하도록 하려면 상위 디렉토리를 지정하는 동안 *DIR 또는 *ALL로 설정된 SUBTREE 매개변수와 함께 실행되어야 합니다. 명령이 시스템 오브젝트의 상위 디렉토리를 정상적으로 재생해야 합니다. 예를 들어,

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

존재하지 않는 경우 /dev/zero 및 /dev/null *CHRSF 오브젝트를 다시 작성합니다.

RCLSTG 명령이 존재하지 않는 통합 파일 시스템 제공 오브젝트를 다시 작성하도록 하려면 *SYSBAS로 설정된 **ASPDEV** 매개변수와 함께 실행되어야 하고 재생의 디렉토리 복구 부분이 생략되어서는 안됩니다.

관련 개념

7 페이지의『제공되는 디렉토리』

시스템이 재시작될 때 통합 파일 시스템은 여기에 나열된 디렉토리가 존재하지 않는다면 이를 생성합니다. 시스템에 의해 생성된 후 이 디렉토리를 이동시키거나 이름 변경하지 않아야 합니다.

관련 정보

RCLLNK(오브젝트 링크 재생) 명령

예: RCLLNK(오브젝트 링크 재생) 명령

이 예는 RCLLNK(오브젝트 링크 재생) 명령을 사용하여 ″루트″(/), QOpenSys 및 마운트된 사용자 정의 파일 시스템의 오브젝트를 복구할 수 있는 상황에 대해 설명합니다.

예: 오브젝트에 대한 문제점 정정

이 상황에서 알려진 문제점은 한 개의 오브젝트로 분리됩니다. 오브젝트가 손상되고 사용할 수 없으며 매체로부터 오브젝트의 백업 버전을 복원할 수 없습니다. 정상적인 파일 시스템 조작을 중단하지 않고 신속하게 문제점을 정정해야 합니다.

오브젝트를 재생하려면 다음 명령을 사용하십시오.

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

여기서 /MyDir/MyBadObject는 손상되고 사용할 수 없는 오브젝트명입니다.

예: 디렉토리 서브트리에 존재하는 문제점 정정

이 상황에서 알려진 문제점은 디렉토리 서브트리 내의 오브젝트 그룹으로 분리됩니다. 디렉토리 서브트리 내의 문제점으로 인해 어플리케이션이 실패했습니다. 정상적인 파일 시스템 조작을 중단하지 않고 신속하게 문제점을 정정해야 합니다.

디렉토리 서브트리 내의 오브젝트를 재생하려면 다음 명령을 사용하십시오.

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

여기서 MyApplicationInstallDirectory는 문제가 발생한 오브젝트를 포함하는 디렉토리명입니다.

예: "루트"(/), QOpenSys 및 마운트된 사용자 정의 파일 시스템의 모든 손상된 오브젝트 찾기

이 상황에서 디스크 장애는 여러 오브젝트를 손상시킬 수 있습니다. 올바르게 복구하는 방법을 판별하기 전에 손상된 오브젝트를 식별해야 합니다.

손상된 오브젝트를 식별하지만 그에 대한 조치를 취하지 않도록 하는 솔루션이 필요합니다. 정상적인 파일 시스템 조작을 중단해서는 안됩니다.

손상된 오브젝트를 식별하려면 다음 명령을 사용하십시오.

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

또한 이 명령은 손상된 오브젝트를 식별할 때 손상된 오브젝트 이외의 문제점도 정정합니다.

예: "루트" (/), QOpenSys 및 마운트된 사용자 정의 파일 시스템의 모든 손상된 오브젝트 삭제

이 상황에서 디스크 장애는 여러 오브젝트를 손상시킬 수 있습니다. 매체로부터 오브젝트의 백업 사본을 복원할 수 있도록 손상된 오브젝트를 삭제해야 합니다.

손상된 오브젝트를 삭제하려면 다음 명령을 사용하십시오.

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

정상적인 파일 시스템 조작을 중단하지 않고 손상된 오브젝트가 삭제됩니다. 또한 손상된 오브젝트가 삭제될 때 손상 이외의 문제점이 정정됩니다.

예: 복수 RCLLNK 명령을 실행하여 ″루트″(/), QOpenSys 및 마운트된 사용자 정의 파일 시스템의 모든 오브젝트를 신속하게 재생

이 상황에서 루틴 시스템 유지보수의 일부로 『루트』(/), QOpenSys 및 마운트된 사용자 정의 파일 시스템의 모든 오브젝트가 재생됩니다. 추가 시스템 유지보수가 완료되도록 하기 위해 재생 조작을 가능한 빠르게 완료하고자 합니다.

재생 조작을 보다 빠르게 완료하기 위해 재생 조작을 별도 그룹으로 구분하여 복수 RCLLNK 명령을 동시에 수행할 수 있습니다.

키 시스템 디렉토리 및 다른 최상위 레벨 디렉토리에 대해 복수 재생 조작을 실행하려면 다음 명령을 사용하십시오(별도 작업 또는 스레드에서 각 명령 사용).

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

여기서 OtherTopLevelDirectories는 재생하려는 다른 디렉토리입니다.

프로그래밍 지원

통합 파일 시스템의 스트림 파일, 디렉토리 및 기타 지원을 활용하려면 통합 파일 시스템 기능에 액세스하기 위해 제공되는 어플리케이션 프로그램 인터페이스(API) 세트를 사용해야 합니다.

또한 통합 파일 시스템의 추가를 통해 실제 데이터베이스 파일과 스트림 파일 간에 데이터를 복사할 수 있습니다. CL 명령, System i Access 제품군의 데이터 전송 기능 또는 API를 사용하여 이러한 복사를 수행할 수 있습니다.

스트림 파일과 데이터베이스 파일 사이의 데이터 복사

데이터 서술 스펙(DDS)과 같은 레코드 지향 기능을 사용하는 데이터베이스 파일 조작에 익숙한 경우 스트림 파일 조작 방식에 있어 몇 가지 기본적인 차이점을 발견할 수 있습니다.

데이터베이스 파일과 비교해서 스트림 파일의 구조가 다르기 때문에(또는 구조가 없기 때문에) 차이가 발생합니다. 스트림 파일의 데이터에 액세스하기 위해서는 바이트 오프셋과 길이를 표시합니다. 사용자는 데이터베이스 파일내 데이터에 액세스하기 위해 사용할 필드와 처리할 레코드 수를 정의합니다.

레코드 지향 파일 특성 및 형식을 미리 정의하므로 오퍼레이팅 시스템은 파일에 대해 알게 되고, 사용자가 파일 형식 및 특성에 적합하지 않은 조작을 수행하는 것을 피하도록 합니다. 스트림 파일의 경우에는 오퍼레이팅 시스템이 파일 형식에 대해 거의 알지 못하거나 전혀 알지 못합니다. 어플리케이션은 파일 형식과 조작 방법에 대해 알고 있어야 합니다. 스트림 파일은 매우 융통성있는 프로그래밍 환경을 허용하나, 오퍼레이팅 시스템으

로부터 도움을 거의 받지 못하거나 전혀 받지 못합니다. 즉, 스트림 파일은 일부 프로그래밍 상황에 보다 적합한 반면, 레코드 지향 파일은 다른 프로그래밍 상황에 보다 적합합니다.

관련 개념

16 페이지의『스트림 파일』

*스트림 파일*은 단순하게 바이트가 나열된 구조로 랜덤 액세스가 가능합니다.

CL 명령을 사용한 데이터 복사

스트림 파일과 데이터베이스 파일 멤버 간에 데이터를 복사할 수 있도록 하는 두 개의 CL 명령 세트가 있습니다.

CPYTOSTMF 및 CPYFRMSTMF 명령

CPYFRMSTMF(스트림 파일로부터 복사) 및 CPYTOSTMF(스트림 파일로 복사) 명령을 사용하여 스트림 파일과 데이터베이스 파일 멤버 간에 데이터를 복사할 수 있습니다. CPYTOSTMF 명령을 사용하여 데이터베이스 파일 멤버로부터 스트림 파일을 작성할 수 있습니다. 또한 CPYFRMSTMF 명령을 사용하여 스트림 파일로부터 데이터베이스 파일 멤버를 작성할 수 있습니다. 복사 목표인 파일 또는 멤버가 존재하지 않는 경우, 파일이나 멤버가 작성됩니다.

그러나 몇 가지 제한사항이 있습니다. 데이터베이스 파일은 반드시 하나의 텍스트 필드가 들어 있는 소스 실제 파일이거나 하나의 필드가 들어 있는 프로그램 설명 실제 파일이어야 합니다. 명령어는 복사 중인 데이터를 변환 및 재형식화하기 위한 다양한 옵션을 제공합니다.

CPYTOSTMF 및 CPYFRMSTMF 명령을 사용하여 스트림 파일과 저장 파일 간에 데이터를 복사할 수도 있습니다.

CPYTOIMPF 및 CPYFRMIMPF 명령

CPYTOIMPF(가져오기 파일로 복사) 및 CPYFRMIMPF(가져오기 파일로부터 복사) 명령을 사용하여 스트림 파일과 데이터베이스 멤버 간에 데이터를 복사할 수도 있습니다. CPYTOSTMF 및 CPYFRMSTMF 명령을 사용하여 복잡한 외부 설명(DDS 설명) 데이터베이스 파일에서 데이터를 이동할 수 없습니다. *가져오기 파일*이란 단어는 스트림 유형 파일을 말하며, 이 용어는 일반적으로 이종 데이터베이스 간에 데이터를 복사하려는 목적으로 작성되는 파일을 말합니다.

스트림(또는 가져오기) 파일로부터 복사할 때 CPYFRMIMPF 명령을 사용하여 스트림 파일의 데이터를 설명하는 필드 정의 파일(FDF)을 지정할 수 있습니다. 또는 스트림 파일이 구분된다는 것과, 스트링, 필드 및 레코드 경계를 표시하기 위해 사용되는 문자를 지정할 수 있습니다. 시간과 날짜와 같은 특수 데이터 유형을 변환하기 위한 옵션도 제공됩니다.

목표 스트림 파일이나 데이터베이스 멤버가 이미 존재하는 경우 이들 명령에 데이터 변환이 제공됩니다. 파일이 존재하지 않는 경우, 다음 두 단계 메소드를 사용하여 데이터를 변환할 수 있습니다.

1. CPYTOIMPF 및 CPYFRMIMPF 명령을 사용하여 외부 설명 파일과 소스 실제 파일 간에 데이터를 복사하십시오.

2. CPYTOSTMF 및 CPYFRMSTMF 명령(대상 파일 존재 여부에 관계없이 전체 데이터 변환 제공)을 사용하여 소스 실제 파일과 스트림 파일 간에 데이터를 복사하십시오.

예는 다음과 같습니다.

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

DTAFMT 매개변수는 입력 스트림(가져오기) 파일을 구분하도록 지정하며, 다른 선택은 필드 정의 파일을 지정하도록 요구하는 DTAFMT(*FIXED)입니다. FLDDLM, RCDDLM 및 STRDLM 매개변수는 필드, 레코드 및 스트링에 대한 분리문자 또는 분리자로서 사용되는 문자를 식별합니다.

DATFMT 및 TIMFMT 매개변수는 가져오기 파일에 복사되는 날짜 및 시간 정보에 대한 형식을 지정합니다.

이 명령들은 프로그램에 배치할 수 있고 전적으로 시스템에서 실행되기 때문에 유용합니다. 그러나 인터페이스가 복잡합니다.

관련 정보

CPYTOSTMF(스트림 파일로 복사) 명령

CPYFRMSTMF(스트림 파일로부터 복사) 명령

CPYTOIMPF(가져오기 파일로 복사) 명령

CPYFRMIMPF(가져오기 파일로부터 복사) 명령

제어 언어(CL)

API를 사용한 데이터 복사

어플리케이션에서 데이터베이스 파일 멤버를 스트림 파일에 복사하는 경우 통합 파일 시스템 open(), read() 및 write() 함수를 사용하여 멤버를 열어 데이터를 읽고 해당 멤버 또는 다른 파일에 데이터를 기록할 수 있습니다.

관련 정보

open()--파일 열기 API

read()--설명자에서 읽기 API

write()--설명자로 쓰기 API

통합 파일 시스템 API

데이터 전송 기능을 사용하여 데이터 복사

System i Access 제품군 라이센스 프로그램의 데이터 전송 어플리케이션은 따라하기 쉬운 그래픽 인터페이스, 자동 숫자 및 문자 데이터 변환의 장점을 가집니다.

그러나 자료 전송을 하려면 System i Access 제품군 제품을 설치해야 하며, PC와 i5/OS 자원 모두를 사용하고 이들 사이의 통신이 필요합니다.

PC와 시스템에 System i Access 제품군을 설치한 경우, 데이터 전송 어플리케이션을 사용하여 스트림 파일과 데이터베이스 파일 간에 데이터를 전송할 수 있습니다. 기존 데이터베이스 파일을 기반으로 신규 데이터베이스 파일이나 외부 서술 데이터베이스 파일로 데이터를 이동시키거나 신규 데이터베이스 파일 정의와 파일로 전송할 수도 있습니다.

데이터베이스 파일에서 스트림 파일로 데이터 전송:

데이터베이스 파일에서 시스템의 스트림 파일로 파일을 전송하려면 다음 단계를 따르십시오.

1. 시스템으로의 연결을 설정하십시오.
2. i5/OS 파일 시스템의 해당 경로에 네트워크 드라이브를 맵핑시키십시오.
3. Windows용 System i Access 창에서, **System i에서 데이터 전송**을 클릭하십시오.
4. 전송할 시스템을 선택하십시오.
5. i5/OS 데이터베이스 라이브러리, 파일명을 사용하여 복사할 파일명을 선택하고, 결과 스트림 파일 위치에 대한 네트워크 드라이브를 선택하십시오. **PC 파일 세부사항**을 선택하여 스트림 파일에 대한 PC 파일 형식을 선택할 수도 있습니다. 데이터 전송은 ASCII 텍스트, BIFF3, CSV, DIF, 탭 분리 텍스트 또는 WK4와 같은 일반 PC 파일 유형을 지원합니다.
6. 파일 전송을 실행하려면 **System i에서 데이터 전송**을 클릭하십시오.

또한 데이터 전송 어플리케이션을 사용하여 일괄처리 작업으로 데이터 이동을 수행할 수 있습니다. 위와 같이 진행하되, **파일** 메뉴 옵션을 선택하여 전송 요구를 저장하십시오. System i로의 데이터 전송 어플리케이션은 .DTT 또는 .TFR 파일을 작성합니다. System i에서 데이터 전송 어플리케이션은 .DTF 또는 .TTO 파일을 작성합니다. System i Access 제품군 디렉토리에서 다음과 같이 명령행에서 두 프로그램을 일괄처리로 실행할 수 있습니다.

- RTOPCB는 .DTF 또는 .TTO 파일을 매개변수로 취합니다.
- RFROMPCB는 .DTT 또는 .TFR 파일을 매개변수로 취합니다.

이들 명령 중 하나를 설정하여 스케줄러 어플리케이션으로 스케줄 기준으로 실행할 수 있습니다. 예를 들어, 시스템 에이전트 툴(Microsoft® Plus Pack의 한 부분)을 사용하여 실행할 프로그램(예를 들면, `RTOPCB MYFILE.TTO`) 및 프로그램을 실행하려는 시간을 지정할 수 있습니다.

스트림 파일에서 데이터베이스 파일로 데이터 전송:

스트림 파일에서 시스템의 데이터베이스 파일로 데이터를 전송하려면 다음 단계를 따르십시오.

1. 시스템으로의 연결을 설정하십시오.
2. i5/OS 파일 시스템의 해당 경로에 네트워크 드라이브를 맵핑시키십시오.
3. Windows용 System i Access 창에서 **System i로 데이터 전송**을 클릭하십시오.
4. 전송할 PC 파일명을 선택하십시오. PC 파일명의 경우, 사용자가 할당한 네트워크 드라이브에 대한 **브라우즈**를 선택하여 스트림 파일을 선택할 수 있습니다. PC에 위치한 스트림 파일을 사용할 수도 있습니다.
5. 외부 서술 데이터베이스 파일이 위치할 시스템을 선택하십시오.
6. 파일 전송을 실행하려면 **System i로 데이터 전송**을 클릭하십시오.

주: 데이터를 시스템의 기존 데이터베이스 파일 정의로 이동시키는 경우, System i로의 데이터 전송 어플리케이션은 연관된 형식 설명 파일(FDF)을 사용할 것을 요구합니다. FDF 파일은 스트림 파일의 형식을 기술하며, 데이터가 데이터베이스 파일에서 스트림 파일로 전송될 때 System i로의 데이터 전송 어플리케이션에 의해 작성됩니다. 스트림 파일에서 데이터베이스 파일로의 데이터 전송을 완료하려면 **System i로 데이터 전송**을 클릭하십시오. 기존 .FDF 파일을 사용할 수 없는 경우 신속하게 .FDF 파일을 작성할 수 있습니다.

또한 데이터 전송 어플리케이션을 사용하여 일괄처리 작업으로 데이터 이동을 수행할 수 있습니다. 위와 같이 진행하되, **파일** 메뉴 옵션을 선택하여 전송 요구를 저장하십시오. System i로의 데이터 전송 어플리케이션은 .DTT 또는 .TFR 파일을 작성합니다. System i에서 데이터 전송 어플리케이션은 .DTF 또는 .TTO 파일을 작성합니다. System i Access 제품군 디렉토리에서 다음과 같이 명령행에서 두 프로그램을 일괄처리로 실행할 수 있습니다.

- RTOPCB는 .DTF 또는 .TTO 파일을 매개변수로 취합니다.
- RFROMPCB는 .DTT 또는 .TFR 파일을 매개변수로 취합니다.

이들 명령 중 하나를 설정하여 스케줄러 어플리케이션으로 스케줄 기준으로 실행할 수 있습니다. 예를 들어, 시스템 에이전트 툴(Microsoft Plus Pack의 한 부분)을 사용하여 실행할 프로그램(예를 들면, `RTOPCB MYFILE.TTO`) 및 프로그램을 실행하려는 시간을 지정할 수 있습니다.

관련 참조

『형식 설명 파일 작성』

데이터를 시스템의 기존 데이터베이스 파일 정의로 이동하는 경우 System i로의 데이터 전송 어플리케이션은 연관된 형식 정의 파일(FDF)을 사용할 것을 요구합니다.

새로 작성된 데이터베이스 파일 정의와 파일로 데이터 전송:

새로 작성된 데이터베이스 파일 정의 및 파일로 자료를 전송하려면 다음 지침에 따르십시오.

1. 시스템으로의 연결을 설정하십시오.
2. i5/OS 파일 시스템의 해당 경로에 네트워크 드라이브를 맵핑시키십시오.
3. Windows용 System i Access 창에서 **System i로 데이터 전송**을 클릭하십시오.
4. System i로 데이터 전송 어플리케이션의 툴을 여십시오.
5. **System i 데이터베이스 파일 작성**을 클릭하십시오.

   기본 PC 파일로부터 새로운 System i 데이터베이스 파일을 작성할 수 있는 마법사가 표시됩니다. System i 파일의 기반이 되는 PC 파일명, 작성할 System i 파일명 및 기타 몇 가지 필요한 세부사항을 지정해야 합니다. 이 툴은 주어진 스트림 파일을 분석하여 결과 데이터베이스 파일에 필요한 필드의 수, 유형 및 크기를 판별합니다. 그러면 툴이 시스템에 데이터베이스 파일 정의를 작성할 수 있게 됩니다.

형식 설명 파일 작성:

데이터를 시스템의 기존 데이터베이스 파일 정의로 이동하는 경우 System i로의 데이터 전송 어플리케이션은 연관된 형식 정의 파일(FDF)을 사용할 것을 요구합니다.

FDF 파일은 스트림 파일의 형식을 기술하며, 데이터가 데이터베이스 파일에서 스트림 파일로 전송될 때 System i로의 데이터 전송 어플리케이션에 의해 작성됩니다.

.FDF 파일을 작성하려면 다음 단계를 수행하십시오.

1. 소스 스트림 파일과 대응하는 형식(필드의 수, 데이터의 유형)으로 외부 설명 데이터베이스 파일을 작성하십시오.
2. 데이터베이스 파일 내에 하나의 임시 데이터 레코드를 작성하십시오.
3. 데이터베이스 파일에서 스트림 파일 및 연관된 .FDF 파일을 작성하려면 System i에서 데이터 전송 기능을 사용하십시오.

   이제 System i로 데이터 전송 기능을 사용할 수 있습니다. 전송하려는 소스 스트림 파일로 이 .FDF 파일을 지정하십시오.

   관련 참조

   120 페이지의『데이터베이스 파일에서 스트림 파일로 데이터 전송』
   데이터베이스 파일에서 시스템의 스트림 파일로 파일을 전송하려면 다음 단계를 따르십시오.

   120 페이지의『스트림 파일에서 데이터베이스 파일로 데이터 전송』
   스트림 파일에서 시스템의 데이터베이스 파일로 데이터를 전송하려면 다음 단계를 따르십시오.

스트림 파일과 저장 파일 간에 데이터 복사

저장 파일은 그 밖의 테이프나 디스켓에 작성될 데이터를 보유하기 위해 저장 및 복원 명령과 함께 사용됩니다.

또한 저장 파일을 데이터베이스 파일과 같이 사용하여 저장/복원 정보가 들어 있는 레코드를 읽거나 쓸 수 있습니다. SNADS 네트워크의 다른 사용자에게 오브젝트를 송신하는 데 저장 파일을 사용할 수도 있습니다.

CPY(오브젝트 복사) 명령을 사용하여 스트림 파일 간에 저장 파일을 복사할 수 있습니다. 그러나 다시 저장 파일 오브젝트로 스트림 파일을 복사할 때, 데이터는 유효한 저장 파일 데이터(저장 파일에서 비롯되어 스트림 파일로 복사된 데이터)여야 합니다.

또한 PC 클라이언트를 사용하여 저장 파일에 액세스하고 PC 기억장치나 LAN으로 데이터를 복사할 수 있습니다. 그러나 네트워크 파일 시스템(NFS)을 통해 저장 파일의 데이터에 액세스할 수 없다는 점에 유의하십시오.

관련 정보

CPY(오브젝트 복사) 명령

API를 사용한 조작 수행

통합 파일 시스템 오브젝트에서 조작을 수행하는 API(Application Programming Interface)의 대부분은 C 언어 함수 형태입니다.

다음 두 세트의 함수 중 하나를 선택하여 통합 언어 환경(ILE) C를 사용하여 작성되는 프로그램에서 사용할 수 있습니다.

- i5/OS 오퍼레이팅 시스템에 포함되어 있는 통합 파일 시스템 C 함수
- ILE C 라이센스 프로그램이 제공하는 C 함수

통합 파일 시스템을 지원하는 기존 프로그램에 대한 정보는 127 페이지의 표 14를 참조하십시오.

통합 파일 시스템 기능은 통합 파일 시스템 스트림 I/O 지원을 통해서만 작동합니다. 다음 API가 지원됩니다.

표 13. 통합 파일 시스템 API

함수	설명
access()	파일 액세스 가능성 판별
accessx()	사용자 클래스에 대한 파일 액세스 가능성 판별
chdir()	현재 디렉토리 변경
chmod()	파일 권한 변경
chown()	파일 소유자 및 그룹 변경
close()	파일 설명자 닫기
closedir()	디렉토리 닫기
creat()	새로운 파일 작성 또는 기존 파일 다시쓰기
creat64()	새로운 파일 작성 또는 기존 파일 다시쓰기(대용량 파일 작동가능)
DosSetFileLocks()	파일의 바이트 범위 잠그기 및 잠금 해제
DosSetFileLocks64()	파일의 바이트 범위 잠그기 및 잠금 해제(대용량 파일 작동가능)
DosSetRelMaxFH()	파일 설명자의 최대 수 변경
dup()	열린 파일 설명자 복사
dup2()	다른 설명자로 열린 파일 설명자 복사
faccessx()	설명자를 사용한 사용자 클래스에 대한 파일 액세스 가능성 판별
fchdir()	설명자를 사용한 현재 디렉토리 변경
fchmod()	설명자를 사용한 파일 권한 변경
fchown()	설명자를 사용한 파일 소유자 및 그룹 변경
fclear()	파일 지우기
fclear64()	파일 지우기(대용량 파일 작동가능)
fcntl()	파일 제어 조치 수행
fpathconf()	설명자를 사용한 구성 가능 경로명 변수 가져오기
fstat()	설명자를 사용한 파일 정보 가져오기
fstat64()	설명자를 사용한 파일 정보 가져오기(대용량 파일 작동가능)
fstatvfs()	설명자를 사용한 정보 가져오기
fstatvfs64()	설명자를 사용한 정보 가져오기(64 비트 작동가능)
fsync()	파일에 대한 변경 동기화
ftruncate()	파일 절단
ftruncate64()	파일 절단(대용량 파일 작동가능)
getcwd()	현재 디렉토리의 경로명 가져오기
getegid()	유효한 그룹 ID 가져오기
geteuid()	유효한 사용자 ID 가져오기
getgid()	실제 그룹 ID 가져오기
getgrgid()	그룹 ID를 사용한 그룹 정보 가져오기

표 13. 통합 파일 시스템 API (계속)

함수	설명
getgrnam()	그룹명을 사용한 그룹 정보 가져오기
getgroups()	그룹 ID 가져오기
getpwnam()	사용자명에 대한 사용자 정보 가져오기
getpwuid()	사용자 ID에 대한 사용자 정보 가져오기
getuid()	실제 사용자 ID 가져오기
givedescriptor()	다른 작업으로의 파일 액세스 부여
ioctl()	파일 I/O 제어 조치 수행
link()	파일에 대한 링크 작성
lseek()	파일 읽기/쓰기 오프셋 설정
lseek64()	파일 읽기/쓰기 오프셋 설정(대용량 파일 작동가능)
lstat()	파일 또는 링크 정보 가져오기
lstat64()	파일 또는 링크 정보 가져오기(대용량 파일 작동가능)
mkdir()	디렉토리 작성
mkfifo()	FIFO 특수 파일 작성
mmap()	메모리 맵 작성
mmap64()	메모리 맵 작성(대용량 파일 작동가능)
mprotect()	메모리 맵 보호 변경
msync()	메모리 맵 동기화
munmap()	메모리 맵 제거
open()	파일 열기
open64()	파일 열기(대용량 파일 작동가능)
opendir()	디렉토리 열기
pathconf()	구성가능 경로명 변수 가져오기
pread()	오프셋을 사용하여 설명자에서 읽기
pread64()	오프셋을 사용하여 설명자에서 읽기(대용량 파일 사용)
pwrite()	오프셋을 사용하여 설명자에 쓰기
pwrite64()	오프셋을 사용하여 설명자에 쓰기(대용량 파일 사용)
QjoEndJournal()	저널링 종료
QjoRetrieveJournal Information()	저널 정보 검색
QjoRetrieveJournalEntries()	저널 항목 검색
QJORJIDI()	저널 ID 정보 검색
QJOSJRNE()	저널 항목 송신
QjoStartJournal()	저널링 시작
QlgAccess()	파일 액세스 가능성 판별(NLS 작동가능 경로명 사용)
QlgAccessx()	사용자 클래스에 대한 파일 액세스 가능성 판별(NLS 사용 가능 경로명 사용)
QlgChdir()	현재 디렉토리 변경(NLS 작동가능 경로명 사용)
QlgChmod()	파일 권한 변경(NLS 작동가능 경로명 사용)
QlgChown()	파일 소유자와 그룹을 변경(NLS 작동가능 경로명 사용)
QlgCreat()	신규 파일 작성 또는 기존 파일 다시쓰기(NLS 작동가능 경로명 사용)

표 13. 통합 파일 시스템 API (계속)

함수	설명
QlgCreat64()	신규 파일 작성 또는 기존 파일 다시쓰기(대용량 파일 작동가능 및 NLS 작동가능 경로명 사용)
QlgCvtPathToQSYSObjName()	통합 파일 시스템 경로명을 QSYS 오브젝트명으로 분석(NLS 작동가능 경로명 사용)
QlgGetAttr()	오브젝트에 대한 시스템 속성 가져오기(NLS 작동가능 경로명 사용)
QlgGetcwd()	현재 디렉토리의 경로명 가져오기(NLS 작동가능 경로명 사용)
QlgGetPathFromFileID()	파일 ID로부터 오브젝트의 경로명 가져오기(NLS 작동가능 경로명 사용)
QlgGetpwnam()	사용자명에 대한 사용자 정보 가져오기(NLS 작동가능 경로명 사용)
QlgGetpwnam_r()	사용자명에 대한 사용자 정보 가져오기(NLS 작동가능 경로명 사용)
QlgGetpwuid()	사용자 ID에 대한 사용자 정보 가져오기(NLS 작동가능 경로명 사용)
QlgGetpwuid_r()	사용자 ID에 대한 사용자 정보 가져오기(NLS 작동가능 경로명 사용)
QlgLchown()	기호 링크 소유자와 그룹 변경(NLS 작동가능 경로명 사용)
QlgLink()	파일로의 링크 작성(NLS 작동가능 경로명 사용)
QlgLstat()	파일 또는 링크 정보 가져오기(NLS 작동가능 경로명 사용)
QlgLstat64()	파일 또는 링크 정보 가져오기(대용량 파일 작동가능 및 NLS 작동가능 경로명 사용)
QlgMkdir()	디렉토리 작성(NLS 작동가능 경로명 사용)
QlgMkfifo()	FIFO 특수 파일 작성(NLS 작동가능 경로명 사용)
QlgOpen()	파일 열기(NLS 작동가능 경로명 사용)
QlgOpen64()	파일 열기(대용량 파일 작동가능 및 NLS 작동가능 경로명 사용)
QlgOpendir()	디렉토리 열기(NLS 작동가능 경로명 사용)
QlgPathconf()	구성가능 경로명 변수 가져오기(NLS 작동가능 경로명 사용)
QlgProcessSubtree()	디렉토리 트리에서 디렉토리 또는 오브젝트 처리(NLS 작동가능 경로명 사용)
QlgReaddir()	디렉토리 항목 읽기(NLS 작동가능 경로명 사용)
QlgReaddir_r()	디렉토리 항목 읽기(스레드세이프 및 NLS 작동가능 경로명 사용)
QlgReadlink()	기호 링크 값 읽기(NLS 작동가능 경로명 사용)
QlgRenameKeep()	파일 또는 디렉토리 이름 변경, 이미 있으면 *신규* 유지(NLS 작동가능 경로명 사용)
QlgRenameUnlink()	파일 또는 디렉토리 이름 변경, 이미 있으면 *신규* 링크 해제(NLS 작동가능 경로명 사용)
QlgRmdir()	디렉토리 제거(NLS 작동가능 경로명 사용)
QlgSaveStgFree()	오브젝트 데이터 저장 기억장치 비우기(NLS 작동가능 경로명 사용)
QlgSetAttr()	오브젝트에 대한 시스템 속성 설정(NLS 작동가능 경로명 사용)
QlgStat()	파일 정보 가져오기(NLS 작동가능 경로명 사용)
QlgStat64()	파일 정보 가져오기(대용량 파일 작동가능 및 NLS 작동가능 경로명 사용)
QlgStatvfs()	파일 시스템 정보 가져오기(NLS 작동가능 경로명 사용)
QlgStatvfs64()	파일 시스템 정보 가져오기(대용량 파일 작동가능 및 NLS 작동가능 경로명 사용)

표 13. 통합 파일 시스템 API (계속)

함수	설명
QlgSymlink()	기호 링크 작성(NLS 작동가능 경로명 사용)
QlgUnlink()	파일 링크 해제(NLS 작동가능 경로명 사용)
QlgUtime()	파일 액세스 및 수정 시간 설정(NLS 작동가능 경로명 사용)
QP0FPTOS()	기타 파일 시스템 기능 수행
QP0LCHSG()	스캔 서명 변경
Qp0lCvtPathToSYSObjName()	통합 파일 시스템 경로명을 QSYS 오브젝트명으로 분석
QP0LFLOP()	오브젝트에 대한 기타 조작 수행
Qp0lGetAttr()	오브젝트에 대한 시스템 속성 가져오기
Qp0lGetPathFromFileID()	파일 ID로부터 오브젝트의 경로명 가져오기
Qp0lOpen()	NLS 작동가능 경로명이 있는 파일 열기
Qp0lProcessSubtree()	디렉토리 트리 내에서 디렉토리나 오브젝트 처리
Qp0lRenameKeep()	파일 또는 디렉토리를 이름 변경하고, 이미 있으면 *신규* 유지
Qp0lRenameUnlink()	파일 또는 디렉토리 이름 변경, 이미 있으면 *신규* 링크 해제
QP0LROR()	오브젝트 참조 검색
QP0LRRO()	참조된 오브젝트 검색
QP0LRTSG()	스캔 서명 검색
Qp0lSaveStgFree()	오브젝트 데이터 저장 및 기억장치 비우기
Qp0lSetAttr()	오브젝트에 대한 시스템 속성 설정
Qp0lUnlink()	NLS 작동가능 경로명이 있는 파일 링크 해제
Qp0zPipe()	소켓을 사용한 프로세스간 채널 작성
qsysetegid()	유효한 그룹 ID 설정
qsyseteuid()	유효한 사용자 ID 설정
qsysetgid()	그룹 ID 설정
qsysetregid()	실질적이고 효율적인 그룹 ID 설정
qsysetreuid()	실질적이고 효율적인 사용자 ID 설정
qsysetuid()	사용자 ID 설정
QZNFRTVE()	NFS 내보내기 정보 검색
read()	파일에서 읽기
readdir()	디렉토리 항목 읽기
readdir_r()	디렉토리 항목 읽기(스레드세이프)
readlink()	기호 링크 값 읽기
readv()	파일에서 읽기(벡터)
rename()	파일 또는 디렉토리 이름 변경. Qp0lRenameKeep() 또는 Qp0lRenameUnlink()의 의미를 정의할 수 있음.
rewinddir()	디렉토리 스트림 재설정
rmdir()	디렉토리 제거
select()	복수 파일 설명자의 I/O 상태 검사
stat()	파일 정보 가져오기
stat64()	파일 정보 가져오기(대용량 파일 작동가능)
statvfs()	파일 시스템 정보 가져오기

표 13. 통합 파일 시스템 API (계속)

함수	설명
statvfs64()	파일 시스템 정보 가져오기(대용량 파일 작동가능)
symlink()	기호 링크 작성
sysconf()	시스템 구성 변수 가져오기
takedescriptor()	다른 작업에서 파일 액세스 가져오기
umask()	작업에 대한 권한 마스크 설정
unlink()	파일에 대한 링크 제거
utime()	파일 액세스 및 수정 횟수 설정
write()	파일 기록
writev()	파일 기록(벡터)

주: 일부 함수는 i5/OS 소켓에 대해서도 사용됩니다.

표 14. 통합 파일 시스템 종료 프로그램

함수	설명
닫기 종료 프로그램에 대한 통합 파일 시스템 스캔	close() API와 같은 닫기 처리 중 호출됨. 이 종료 프로그램은 사용자에 의해 제공되어야 합니다.
열기 종료 프로그램에 대한 통합 파일 시스템 스캔	open() API와 같은 열기 처리 중 호출됨. 이 종료 프로그램은 사용자에 의해 제공되어야 합니다.
경로명 처리	호출자 선택 기준에 일치하는 API의 검색에 있는 각 오브젝트를 위한 Qp0lProcessSubtree() API에 의해 호출됨. 이 종료 프로그램은 사용자에 의해 제공되어야 합니다.
사용 가능한 저장 기억장치	*STMF 오브젝트 유형을 저장하기 위해 Qp0lSaveStgFree() API에 의해 호출됨. 이 종료 프로그램은 사용자에 의해 제공되어야 합니다.

관련 개념

26 페이지의『파일 시스템』

*파일 시스템*은 논리 장치(LU)를 구성하는 요소인 기억장치의 특정 세그먼트에 액세스할 수 있도록 지원합니다. 시스템에서 논리 장치는 파일, 디렉토리, 라이브러리 및 오브젝트입니다.

관련 참조

134 페이지의『예: 통합 파일 시스템 C 함수』

다음의 간단한 C 언어 프로그램은 여러 통합 파일 시스템 함수가 사용되는 예를 보여줍니다.

119 페이지의『API를 사용한 데이터 복사』

어플리케이션에서 데이터베이스 파일 멤버를 스트림 파일에 복사하는 경우 통합 파일 시스템 open(), read() 및 write() 함수를 사용하여 멤버를 열어 데이터를 읽고 해당 멤버 또는 다른 파일에 데이터를 기록할 수 있습니다.

관련 정보

어플리케이션 프로그램 인터페이스(API)

ILE C 함수

ILE C는 미국 표준 협회(ANSI)가 정의하는 표준 C 함수를 제공합니다.

이 함수는 C 프로그램을 작성할 때 사용자가 지정한 것에 따라 데이터 관리 I/O 지원 또는 통합 파일 시스템 스트림 I/O 지원을 통해 작동할 수 있습니다. 컴파일러는 사용자가 다르게 지시하지 않는 한, 데이터 관리 I/O를 사용합니다.

컴파일러가 통합 파일 시스템 스트림 I/O를 사용하도록 지시하려면 CRTCMOD(ILE C 모듈 작성) 또는 CRTBNDC(바인드된 C 프로그램 작성) 명령의 SYSIFCOPT(시스템 인터페이스 옵션) 매개변수에 *IFSIO를 지정해야 합니다. *IFSIO를 지정할 때 통합 파일 시스템 I/O 함수가 데이터 관리 I/O 함수 대신 바인드됩니다. 사실상 ILE C 함수는 I/O를 수행하기 위해 통합 파일 시스템 함수를 사용합니다.

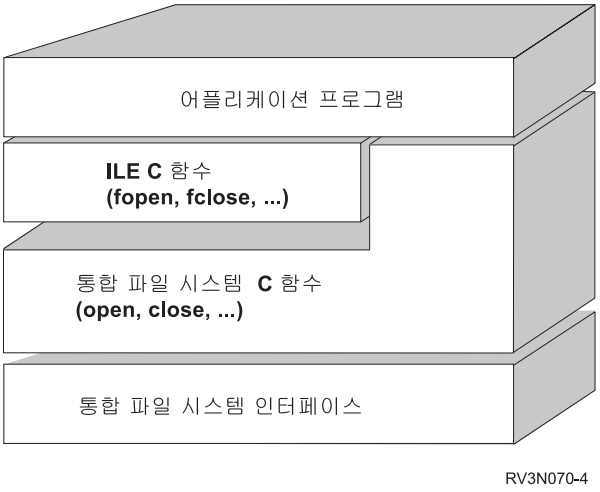

그림 10. ILE C 함수는 통합 파일 시스템 스트림 I/O 함수를 사용합니다.

통합 파일 시스템 스트림 I/O로 ILE C 함수를 사용하는 것에 대한 자세한 정보는 WebSphere® Development Studio: ILE C/C++ Programmer's Guide 서적을 참조하십시오. ILE C의 C 함수에 대한 세부사항은 WebSphere Development Studio: C/C++ Language Reference 서적을 참조하십시오.

대용량 파일 지원

어플리케이션에서 매우 대용량 파일을 저장하고 처리할 수 있도록 통합 파일 시스템 API가 향상되었습니다. 통합 파일 시스템은 『루트』(/), QOpenSys 및 사용자 정의 파일 시스템에서 스트림 파일 크기를 약 1TB(1TB는 약 1,099,511,627,776바이트와 같음)까지 허용합니다.

통합 파일 시스템은 64비트 UNIX 유형 API 세트를 제공하며 8바이트 정수 인수를 사용하여 대용량 파일 크기 및 오프셋에 액세스할 수 있는 64비트 API 대 기존 32비트 API의 맵핑을 용이하게 합니다.

어플리케이션에서 대용량 파일 지원을 사용할 수 있도록 다음 상황이 제공됩니다.

- 매크로 레이블 `_LARGE_FILE_API`가 컴파일 시간에 정의되는 경우, 어플리케이션은 64비트를 사용할 수 있는 API 및 데이터 구조에 액세스합니다. 예를 들어, stat64() API 및 `stat64` 구조를 사용하려는 어플리케이션은 컴파일 시간에 `_LARGE_FILE_API`를 정의해야 합니다.

- 어플리케이션이 매크로 레이블 _LARGE_FILES를 컴파일 시간에 정의하는 경우, 기존 API 및 데이터 구조는 해당 64 비트 버전에 맵핑됩니다. 예를 들어, 어플리케이션이 컴파일 시간에 _LARGE_FILES를 정의하는 경우 stat() API에 대한 호출은 stat64() API로 맵핑되고 stat() 구조는 stat64() 구조에 맵핑됩니다.

대용량 파일 지원을 사용하려는 어플리케이션은 컴파일 시간에 _LARGE_FILE_API를 정의하여 코드가 64 비트 API에 직접 코드화하거나 컴파일 시간에 _LARGE_FILES를 정의할 수 있습니다. 그런 다음, 모든 적절한 API 및 데이터 구조는 64 비트 버전에 자동으로 맵핑됩니다.

대용량 파일 지원을 사용하지 않으려는 어플리케이션은 영향을 받지 않으며 변경사항 없이 계속해서 통합 파일 시스템 API를 사용할 수 있습니다.

관련 정보

통합 파일 시스템 API

stat64()--파일 정보 얻기(큰 파일 사용 가능) API

stat()--파일 정보 얻기 API

API에 대한 경로명 규칙

오브젝트에 대해 조작하기 위해 통합 파일 시스템 또는 ILE C API를 사용하는 경우 디렉토리 경로를 제공하여 오브젝트를 식별합니다. 다음은 API에 경로명을 지정할 때 유의해야 할 규칙을 요약한 것입니다.

이 규칙에서 *오브젝트*는 디렉토리, 파일, 링크 또는 기타 오브젝트를 말합니다.

- 경로명은 계층 순서상 디렉토리 계층의 최고 레벨로 시작하여 지정됩니다. 경로의 각 구성요소명은 슬래시(/)로 구분되며, 다음은 그 예입니다.

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

  백슬래시(₩)는 분리자로 인식되지 않습니다. 이름에서 다른 문자로 처리됩니다.
- 오브젝트명은 디렉토리 내에서 고유해야 합니다.
- 각 파일 시스템에 대한 경로명의 각 구성요소의 최대 길이 및 경로명 스트링의 최대 길이는 달라질 수 있습니다.
- 경로명 시작 부분의 / 문자는 경로가 /(『루트』) 디렉토리에서 시작함을 의미하며, 다음은 그 예입니다.

  ```
  /Dir1/Dir2/Dir3/UsrFile
  ```
- 경로명이 / 문자로 시작하지 않은 경우, 경로는 현재 디렉토리에서 시작한다고 간주되며, 다음은 그 예입니다.

  ```
  MyDir/MyFile
  ```

  여기서 MyDir은 현재 디렉토리의 서브디렉토리입니다.
- i5/OS 특수 값과의 혼돈을 피하기 위해 경로명은 단 하나의 별표(*) 문자로 시작할 수 없습니다. 문자로 시작하는 경로명을 지정하는 경우 다음 예와 같이 별표(*) 두 개를 사용하십시오.

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

  이것은 별표(*) 앞에 기타 다른 문자가 없는 상대 경로명에만 적용됩니다.

- QSYS.LIB 파일 시스템의 오브젝트에 대해 조작할 때 구성요소명은 *name.object-type*의 형태여야 하며 예는 다음과 같습니다.

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

- 독립 ASP QSYS.LIB 파일 시스템의 오브젝트에 대해 조작할 때 구성요소명은 *name.object-type*의 형태여야 하며 예는 다음과 같습니다.

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

- 경로명에 콜론(:)을 사용하지 마십시오. 이는 시스템 내에서 특별한 의미를 가집니다.
- 통합 파일 시스템 명령의 경로명과 달리 별표(*), 의문 부호(?), 작은 따옴표('), 인용 부호(") 및 틸드(~)에는 특별한 의미가 없습니다. 이들은 단지 이름에서 다른 여러 문자처럼 처리됩니다. i5/OS 특수 값과의 혼동을 피하기 위해 경로명은 단일 별표(*) 문자로 시작되어서는 안됩니다. 이 규칙의 예외인 유일한 API는 QjoEndJournal 및 QjoStartJournal입니다.
- Qlg(NLS 작동가능 경로명 사용) API 인터페이스를 사용할 때, 널(null) 문자가 경로명 분리문자로 지정되지 않는 한 널(null) 문자 값은 경로명의 문자 중 하나로 허용되지 않습니다.

관련 참조

77 페이지의 『CL 명령 및 표시장치에 대한 경로명 규칙』
통합 파일 시스템 명령이나 화면을 사용하여 오브젝트에 대해 조작할 때 경로명을 제공하여 오브젝트를 식별합니다.

관련 정보

저널 종료(QjoEndJournal) API

저널 시작(QjoStartJournal) API

파일 설명자

파일상에서 조작을 수행하기 위해 미국 표준 협회(ANSI)가 지정한 대로 ILE C 스트림 I/O 함수를 사용하면 포인터를 사용하여 파일을 인식하게 됩니다. 통합 파일 시스템 C 함수를 사용할 때는 파일 설명자를 지정하여 파일을 식별합니다. *파일 설명자*는 각 작업에 고유한 양의 정수입니다.

이 작업은 파일에서 조작을 수행할 때 열린 파일을 식별하기 위해 파일 설명자를 사용합니다. 파일 설명자는 통합 파일 시스템에서 작동하는 C 함수의 *fildes* 변수와 소켓에서 작동하는 C 함수의 *descriptor* 변수로 표시됩니다.

각 파일 설명자는 파일 오프셋, 파일 상태, 파일에 대한 액세스 모드와 같은 정보가 들어 있는 *열린 파일 설명*을 말합니다. 동일한 열린 파일 설명은 하나 이상의 파일 설명자에 의해 참조될 수 있으나, 파일 설명자는 하나의 열린 파일 설명만을 참조할 수 있습니다.

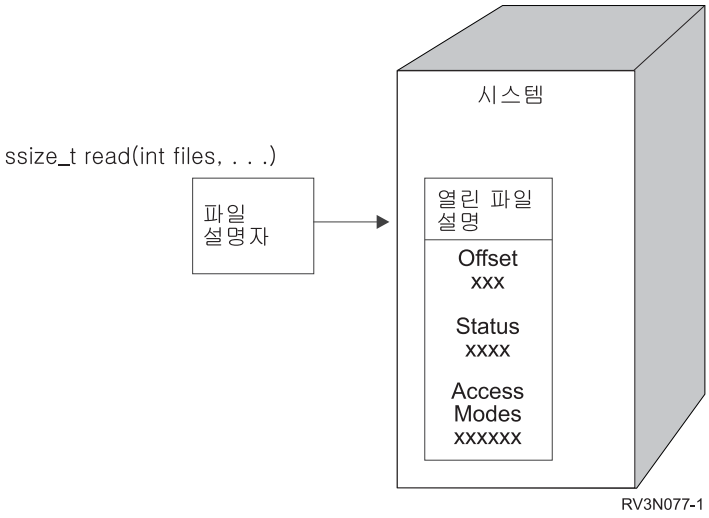

그림 11. 파일 설명자 및 열린 파일 설명

통합 파일 시스템과 함께 ILE C 스트림 I/O 함수가 사용되는 경우 ILE C 런타임 지원은 파일 포인터를 파일 설명자로 변환합니다.

『루트』(/), QOpenSys 또는 사용자 정의 파일 시스템을 사용할 때 작업이 파일에 액세스할 수 있도록 한 작업에서 다른 작업으로 열린 파일 설명에 대한 액세스를 전달할 수 있습니다. 이는 작업간의 파일 설명자를 전달하기 위해 givedescriptor(), takedescriptor(), sendmsg() 또는 recvmsg() 함수를 사용함으로써 이루어집니다.

관련 정보

givedescriptor()--설명자 액세스를 다른 작업으로 전달하는 API

takedescriptor()--다른 작업에서 소켓 액세스를 수신하는 API

sendmsg()--소켓을 통한 메세지 송신

recvmsg()--소켓을 통한 메세지 수신

소켓 프로그래밍

소켓 API

보안

통합 파일 시스템 API 사용 시 데이터 관리 인터페이스를 사용할 때와 마찬가지로 오브젝트에 대한 액세스를 제한할 수 있습니다. 그러나 채택 권한은 지원되지 않는 것에 유의하십시오. 통합 파일 시스템 API는 작업이 실행 중인 사용자 프로파일의 권한을 사용합니다.

각 파일 시스템에는 각각의 특수 권한 조건이 있을 수 있습니다. NFS 서버 및 파일 서버 작업에는 특수 고려사항이 있습니다. 이 작업은 일반적으로 작업의 사용자 프로파일을 소유하고 있지는 않은 사용자를 대신하여 이 기능을 수행합니다. NFS 서버 요구는 사용자의 ID(UID) 번호가 요구 NFS 서버에 의해 수신되었던 사용자의 프로파일에 수행됩니다. 다른 파일 서버 작업은 서버에 접속한 사용자의 요청을 수행합니다.

시스템에 대한 권한은 UNIX 시스템에서의 *권한*과 대등합니다. 권한 유형은(파일 또는 디렉토리에 대한) 읽기 및 기록, (파일에 대한) 실행 또는 (디렉토리에 대한) 탐색입니다. 파일 또는 디렉토리의 액세스 모드를 구성하

는 허용 비트 세트에 의해 허용 권한이 표시됩니다. 변경 모드 함수 chmod() 또는 fchmod()를 사용하면 허용 비트를 변경할 수 있습니다. 또한, umask() 함수를 사용하여 작업이 파일을 작성할 때마다 설정된 파일 허용 비트를 제어할

관련 정보

chmod()--파일 권한 변경 API

fchmod()--설명자로 파일 권한 변경하는 API

umask()--작업 권한 마스크 설정 API

통합 파일 시스템 API

보안 참조

소켓 지원

사용자의 어플리케이션이 『루트』(/), QOpenSys 또는 사용자 정의 파일 시스템을 사용하고 있는 경우 통합 파일 시스템 *로컬 소켓* 지원을 이용할 수 있습니다. 로컬 소켓 오브젝트(오브젝트 유형 *SOCKET)를 이용하여 동일한 시스템에서 두 개의 작업이 수행될 수 있으며, 이로 인해 통신 연결이 이루어집니다.

작업 중 하나는 로컬 소켓 오브젝트를 작성하기 위해 bind() C 언어 함수를 사용하여 연결 지점을 설정합니다. 다른 작업은 connect(), sendto() 또는 sendmsg() 함수에서 로컬 소켓 오브젝트명을 지정합니다.

연결이 설정된 후에 두 작업은 write() 및 read()와 같은 통합 파일 시스템 함수를 사용하여 서로 데이터를 송수신할 수 있습니다. 전송된 데이터 중 어느 것도 사실상 소켓 오브젝트를 통과하지 않습니다. 소켓 오브젝트는 두 개의 작업이 서로를 찾아 만날 수 있는 지점입니다.

두 작업이 통신을 끝낼 때, 각 작업은 close() 함수를 사용하여 소켓 연결을 닫습니다. 로컬 소켓 오브젝트는 unlink() 함수 또는 RMVLNK(링크 제거) 명령을 사용하여 제거될 때까지 시스템에 남아 있습니다.

로컬 소켓 오브젝트는 저장될 수 없습니다.

관련 정보

소켓 프로그래밍

write()--설명자로 쓰기 API

read()--설명자에서 읽기 API

close()--파일 또는 소켓 설명자를 닫는 API

unlink()--파일 링크 제거 API

RMVLNK(링크 제거) 명령

명명 및 국제적 지원

오브젝트명의 문자는 『루트』(/) 및 QOpenSys 파일 시스템 지원으로 인해 서로 다른 자국어 지원 제품 및 장치에 사용되는 코드화 체계와는 상관없이 상수로 남아 있게 됩니다.

| 오브젝트명이 시스템으로 전달될 때 오브젝트명의 각 문자는 모든 문자가 표준 코드 표시가 되도록 16비트 형
| 식으로 변환됩니다. 통합 파일 시스템 인터페이스에서 입력으로 이름이 사용되는 경우 그것은 호출자가 사용하
| 는 코드 페이지에서 적합한 코드화된 형식으로 변환됩니다. 이름이 변환되는 코드 페이지에 원래 이름에 사용
| 된 문자가 들어 있지 않은 경우 그 결과는 오류 또는 인쇄할 수 없는 문자를 포함하여 리턴된 정보입니다.

문자가 코드 페이지간에 상수로 남아 있게 되므로, 특정 코드 페이지가 사용될 때 특정 문자가 다른 특정 문자로 변경된다는 가정아래 조작을 해서는 안됩니다. 예를 들어, 숫자 기호 문자와 파운드 문자가 서로 다른 코드 페이지에서 같은 코드화 표시를 갖더라도, 숫자 기호 문자가 파운드 문자로 변경되지는 않습니다.

오브젝트의 확장 속성명은 오브젝트명과 같은 방식으로 변환되어 동일한 고려사항이 적용됨에 유의하십시오.

관련 개념

18 페이지의 『이름 연속성』

『루트』(/), QOpenSys 및 사용자 정의 파일 시스템을 사용할 때 오브젝명의 문자가 동일하게 남아 있도록 하는 시스템 지원을 이용할 수 있습니다.

103 페이지의 『자동 이름 변환 개요』

""루트""(/)와 같이 대소문자 구분을 하지 않는 파일 시스템 및 CASE(*MONO)로 생성된 UDFS는 유니코드 표준 4.0으로 저장된 이름을 지원합니다. 시스템은 이름에 포함된 추가 문자를 지원하기 위해 자동 이름 변환을 실행합니다.

데이터 변환

통합 파일 시스템을 통해 파일에 액세스할 때 파일의 데이터는 파일이 열릴 때 요청되는 열린 모드에 따라 변환되거나 변환되지 않을 수 있습니다.

열린 파일은 다음 두 가지 열림 모드 중 하나가 될 수 있습니다.

2진 변환되지 않고 파일로부터 데이터가 읽히고 파일로 데이터가 기록됩니다. 어플리케이션은 데이터 처리를 담당합니다.

텍스트 데이터는 텍스트 양식으로 가정되어 파일에서 읽히고 파일에 기록됩니다. 파일에서 데이터를 읽을 때 데이터는 파일의 코드화 문자 세트 ID(CCSID)에서 데이터를 받는 시스템, 어플리케이션 또는 작업의 CCSID로 변환됩니다. 파일에 데이터를 기록할 때 데이터는 어플리케이션, 작업 또는 시스템의 CCSID에서 파일의 CCSID로 변환됩니다. 진정한 스트림 파일의 경우, 행 형식 문자(캐리지 리턴, 탭, 파일의 끝)는 한 CCSID에서 다른 CCSID로 변환됩니다.

스트림 파일로 사용 중인 레코드 파일로부터 읽을 때, 행 종료 문자(캐리지 리턴 및 라인 피드)는 각 레코드의 데이터 끝에 첨가됩니다. 레코드 파일로 기록될 때는 다음 상황이 발생합니다.

- 행 종료 문자는 제거됩니다.
- 탭 문자는 해당 공백만큼 다음 탭 위치로 교체됩니다.
- 행은 레코드 끝부분까지 공백(소스 실제 파일 멤버인 경우)이나 널 문자(데이터 실제 파일 멤버인 경우)로 채워집니다.

열기 요청시 다음 중 하나가 지정될 수 있습니다.

2진, 강제

실제 파일 내용에 관계없이 데이터는 2진으로 처리됩니다. 어플리케이션이 데이터 처리 방법을 알아냅니다.

텍스트, 강제

데이터는 텍스트로 간주됩니다. 데이터는 파일의 CCSID에서 어플리케이션의 CCSID로 변환됩니다.

디폴트인 *2진, 강제*가 통합 파일 시스템 open() 함수에 대해 사용됩니다.

관련 정보

open()--파일 열기 API

예: 통합 파일 시스템 C 함수

다음의 간단한 C 언어 프로그램은 여러 통합 파일 시스템 함수가 사용되는 예를 보여줍니다.

프로그램은 다음 연산을 수행합니다.

1 실제 사용자 ID(uid)를 판별하기 위해 getuid() 함수를 사용합니다.

2 현재 디렉토리를 판별하기 위해 getcwd() 함수를 사용합니다.

3 파일을 작성하기 위해 open() 함수를 사용합니다. 소유자(파일을 작성한 사람) 파일에 대한 읽기, 쓰기, 실행하기 권한을 확립합니다.

4 파일에 1바이트의 스트링을 작성하기 위해 write() 함수를 사용합니다. 열기 조작(3)에서 제공된 파일 설명자가 파일을 식별합니다.

5 파일을 닫기 위해 close() 함수를 사용합니다.

6 현재 디렉토리에 새로운 서브디렉토리를 작성하기 위해 mkdir() 함수를 사용합니다. 소유자는 서브디렉토리에 읽기, 쓰기 및 실행 액세스를 할 수 있습니다.

7 새로운 서브디렉토리를 현재 디렉토리로 변경하기 위해 chdir() 함수를 사용합니다.

8 이전에 작성된 파일로의 링크(3)를 작성하기 위해 link() 함수를 사용합니다.

9 읽기 전용 파일을 열기 위해 open() 함수를 사용합니다. (8)에서 작성된 링크에서 파일 액세스를 허용합니다.

10 파일로부터 바이트 열을 읽기 위해 read() 함수를 사용합니다. 열기 조작(9)에서 제공된 파일 설명자가 파일을 식별합니다.

11 파일을 닫기 위해 close() 함수를 사용합니다.

12 파일로의 링크를 제거하기 위해 unlink() 함수를 사용합니다.

13 새로운 서브디렉토리가 작성된 상위 디렉토리로 다시 현재의 디렉토리를 변경하기 위해 chdir() 함수를 사용합니다.

14 이전에 작성된 서브디렉토리(6)를 제거하기 위해 rmdir() 함수를 사용합니다.

15 이전에 작성된 파일(3)을 제거하기 위해 unlink() 함수를 사용합니다.

주: 이 샘플 프로그램은 프로그램이 수행되는 작업의 CCSID가 37인 시스템에서 제대로 수행됩니다. 통합 파일 시스템 API에는 반드시 작업의 CCSID로 코드화된 오브젝트와 경로가 있어야 하지만 C 컴파일러가 CCSID 37로 문자 상수를 저장합니다. 완전한 호환성을 위해 작업의 CCSID로 API를 전달하기 전에 오브젝트나 경로명 등과 같은 문자 상수를 변환하십시오.

주: 해당 코드 예제를 사용하는 것은 155 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

```
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>


#define BUFFER_SIZE       2048
#define NEW_DIRECTORY     "testdir"
#define TEST_FILE         "test.file"
#define TEST_DATA         "Hello World!"
#define USER_ID           "user_id_"
#define PARENT_DIRECTORY  ".."


char  InitialFile[BUFFER_SIZE];
char  LinkName[BUFFER_SIZE];
char  InitialDirectory[BUFFER_SIZE] = ".";
char  Buffer[32];
int   FilDes = -1;
int   BytesRead;
int   BytesWritten;
uid_t UserID;


void CleanUpOnError(int level)
{
      printf("Error encountered, cleaning up.₩n");
      switch ( level )
      {
              case 1:
                      printf("Could not get current working directory.₩n");
                      break;
              case 2:
                      printf("Could not create file %s.₩n",TEST_FILE);
                      break;
              case 3:
                      printf("Could not write to file %s.₩n",TEST_FILE);
                      close(FilDes);
                      unlink(TEST_FILE);
                      break;
              case 4:
                      printf("Could not close file %s.₩n",TEST_FILE);
                      close(FilDes);
                      unlink(TEST_FILE);
                      break;
              case 5:
```

```
        printf("Could not make directory %s.₩n",NEW_DIRECTORY);
        unlink(TEST_FILE);
        break;
case 6:
        printf("Could not change to directory %s.₩n",NEW_DIRECTORY);
        rmdir(NEW_DIRECTORY);
        unlink(TEST_FILE);
        break;
case 7:
        printf("Could not create link %s to %s.₩n",LinkName,InitialFile);
        chdir(PARENT_DIRECTORY);
        rmdir(NEW_DIRECTORY);
        unlink(TEST_FILE);
        break;
case 8:
        printf("Could not open link %s.₩n",LinkName);
        unlink(LinkName);
        chdir(PARENT_DIRECTORY);
        rmdir(NEW_DIRECTORY);
        unlink(TEST_FILE);
        break;
case 9:
        printf("Could not read link %s.₩n",LinkName);
        close(FilDes);
        unlink(LinkName);
        chdir(PARENT_DIRECTORY);
        rmdir(NEW_DIRECTORY);
        unlink(TEST_FILE);
        break;
case 10:
        printf("Could not close link %s.₩n",LinkName);
        close(FilDes);
        unlink(LinkName);
        chdir(PARENT_DIRECTORY);
        rmdir(NEW_DIRECTORY);
        unlink(TEST_FILE);
        break;
case 11:
        printf("Could not unlink link %s.₩n",LinkName);
        unlink(LinkName);
        chdir(PARENT_DIRECTORY);
        rmdir(NEW_DIRECTORY);
        unlink(TEST_FILE);
        break;
case 12:
        printf("Could not change to directory %s.₩n",PARENT_DIRECTORY);
        chdir(PARENT_DIRECTORY);
        rmdir(NEW_DIRECTORY);
        unlink(TEST_FILE);
        break;
case 13:
        printf("Could not remove directory %s.₩n",NEW_DIRECTORY);
        rmdir(NEW_DIRECTORY);
        unlink(TEST_FILE);
        break;
case 14:
        printf("Could not unlink file %s.₩n",TEST_FILE);
        unlink(TEST_FILE);
```

```
                        break;
                default:
                        break;
        }
        printf("Program ended with Error.₩n"₩
                      "All test files and directories may not have been removed.₩n");
}

int main ()
{
 1
/* Get and print the real user id with the getuid() function. */
   UserID = getuid();
      printf("The real user id is %u. ₩n",UserID);

 2
/* Get the current working directory and store it in InitialDirectory. */
   if ( NULL == getcwd(InitialDirectory,BUFFER_SIZE) )
      {
            perror("getcwd Error");
            CleanUpOnError(1);
        return 0;
      }
      printf("The current working directory is %s. ₩n",InitialDirectory);

 3
/* Create the file TEST_FILE for writing, if it does not exist.
      Give the owner authority to read, write, and execute. */
   FilDes = open(TEST_FILE, O_WRONLY | O_CREAT | O_EXCL, S_IRWXU);
      if ( -1 == FilDes )
      {
            perror("open Error");
            CleanUpOnError(2);
        return 0;
      }
      printf("Created %s in directory %s.₩n",TEST_FILE,InitialDirectory);

 4
/* Write TEST_DATA to TEST_FILE via FilDes */
      BytesWritten = write(FilDes,TEST_DATA,strlen(TEST_DATA));
      if ( -1 == BytesWritten )
      {
            perror("write Error");
            CleanUpOnError(3);
        return 0;
      }
      printf("Wrote %s to file %s.₩n",TEST_DATA,TEST_FILE);

 5
/* Close TEST_FILE via FilDes */
      if ( -1 == close(FilDes) )
      {
            perror("close Error");
            CleanUpOnError(4);
        return 0;
      }
      FilDes = -1;
      printf("File %s closed.₩n",TEST_FILE);
```

```
 6
/* Make a new directory in the current working directory and
      grant the owner read, write and execute authority */
      if ( -1 == mkdir(NEW_DIRECTORY, S_IRWXU) )
      {
            perror("mkdir Error");
            CleanUpOnError(5);
        return 0;
      }
      printf("Created directory %s in directory %s.₩n",NEW_DIRECTORY,InitialDirectory);

 7
/* Change the current working directory to the
      directory NEW_DIRECTORY just created. */
   if ( -1 == chdir(NEW_DIRECTORY) )
      {
            perror("chdir Error");
            CleanUpOnError(6);
        return 0;
      }
      printf("Changed to directory %s/%s.₩n",InitialDirectory,NEW_DIRECTORY);

/* Copy PARENT_DIRECTORY to InitialFile and
      append  "/" and TEST_FILE to InitialFile. */
   strcpy(InitialFile,PARENT_DIRECTORY);
      strcat(InitialFile,"/");
      strcat(InitialFile,TEST_FILE);

/* Copy USER_ID to LinkName then append the
      UserID as a string to LinkName. */
   strcpy(LinkName, USER_ID);
      sprintf(Buffer, "%d₩0", (int)UserID);
      strcat(LinkName, Buffer);

 8
/* Create a link to the InitialFile name with the LinkName. */
   if ( -1 == link(InitialFile,LinkName) )
      {
            perror("link Error");
            CleanUpOnError(7);
        return 0;
      }
      printf("Created a link %s to %s.₩n",LinkName,InitialFile);

 9
/* Open the LinkName file for reading only. */
   if ( -1 == (FilDes = open(LinkName,O_RDONLY)) )
      {
            perror("open Error");
            CleanUpOnError(8);
        return 0;
      }
      printf("Opened %s for reading.₩n",LinkName);

 10
/* Read from the LinkName file, via FilDes, into Buffer. */
   BytesRead = read(FilDes,Buffer,sizeof(Buffer));
```

```
      if ( -1 == BytesRead )
      {
            perror("read Error");
            CleanUpOnError(9);
        return 0;
      }
      printf("Read %s from %s.₩n",Buffer,LinkName);
      if ( BytesRead != BytesWritten )
      {
            printf("WARNING: the number of bytes read is "₩
                        "not equal to the number of bytes written.₩n");
      }

 11
/* Close the LinkName file via FilDes. */
   if ( -1 == close(FilDes) )
      {
            perror("close Error");
            CleanUpOnError(10);
        return 0;
      }
      FilDes = -1;
      printf("Closed %s.₩n",LinkName);

 12
/* Unlink the LinkName link to InitialFile. */
   if ( -1 == unlink(LinkName) )
      {
            perror("unlink Error");
            CleanUpOnError(11);
        return 0;
      }
      printf("%s is unlinked.₩n",LinkName);

 13
/* Change the current working directory
      back to the starting directory. */
   if ( -1 == chdir(PARENT_DIRECTORY) )
      {
            perror("chdir Error");
            CleanUpOnError(12);
        return 0;
      }
      printf("changing directory to %s.₩n",InitialDirectory);

 14
/* Remove the directory NEW_DIRECTORY */
      if ( -1 == rmdir(NEW_DIRECTORY) )
      {
            perror("rmdir Error");
            CleanUpOnError(13);
        return 0;
      }
      printf("Removing directory %s.₩n",NEW_DIRECTORY);

 15
/* Unlink the file TEST_FILE */
      if ( -1 == unlink(TEST_FILE) )
```

```
      {
            perror("unlink Error");
            CleanUpOnError(14);
        return 0;
      }
      printf("Unlinking file %s.₩n",TEST_FILE);


      printf("Program completed successfully.₩n");
     return 0;
}
```

System i Navigator를 사용한 파일 및 폴더에 대한 작업

파일 및 폴더에 대해 다음 타스크를 수행할 수 있습니다.

폴더 작성

폴더를 작성하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → *시스템* → **파일 시스템** → **통합 파일 시스템**을 확장하십시오.
2. 새로운 폴더를 추가할 파일 시스템 내의 파일 시스템 또는 폴더를 마우스 오른쪽 버튼으로 클릭하고 **새 폴더**를 선택하십시오.
3. **신규 폴더** 대화상자에서 폴더명을 입력하고 폴더 속성을 지정합시오.
   - "루트"(/), QOpenSys 및 사용자 정의 파일 시스템의 필드 값을 지정할 수 있습니다. 지정할 수 있는 필드 값은 **이름 변경 및 링크 제거 제한**, **폴더에서 생성된 감사 오브젝트**, 및 **폴더에서 생성된 스캔 오브젝트**입니다.
   - 다른 파일 시스템에서 작성된 폴더에 대해서는 **폴더에서 생성된 감사 오브젝트** 필드에 대한 값을 지정할 수 있습니다.
4. **확인**을 클릭하십시오.

System i 플랫폼에서 폴더를 작성할 경우 저널 관리를 사용하여 새 폴더(또는 오브젝트)를 보호할지를 고려해야 합니다. 이 폴더에 작성되는 오브젝트의 스캔 여부도 고려해야 합니다.

관련 태스크

147 페이지의 『오브젝트가 스캔되어야 하는지 여부 설정』

"루트"(/), QOpenSys 및 사용자 정의 파일 시스템에서 오브젝트를 스캔할 수 있는지 여부를 지정할 수 있습니다. 이 스캔 옵션 설정을 하려면 다음 단계를 수행하십시오.

관련 정보

저널 관리

파일 또는 폴더 제거

파일 또는 폴더를 제거하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → *시스템* → **파일 시스템** → **통합 파일 시스템**을 확장하십시오. 제거할 오브젝트가 나타날 때까지 계속 확장하십시오.

2. 파일 또는 폴더를 마우스 오른쪽 버튼으로 클릭하고 **삭제**를 선택하십시오. 삭제할 항목의 목록을 보여 주는 확인 패널이 나타납니다. 삭제하기 원하지 않는 항목은 선택을 해제합니다. 확인 패널에서 **삭제**를 누르십시오.

   주: 폴더 삭제 시 폴더 내 모든 컨텐츠는 삭제됩니다.

다른 파일 시스템으로 파일 또는 폴더 이동

각 파일 시스템에는 고유한 특성이 있습니다. 그러나 다른 파일 시스템으로 오브젝트를 이동하면 오브젝트가 현재 저장된 파일 시스템의 장점을 잃게 될 수 있습니다. 해당 특성을 이용하기 위해 한 파일 시스템에서 다른 파일 시스템으로 오브젝트를 이동하고자 할 수 있습니다.

오브젝트를 다른 파일 시스템으로 이동하기 전에 통합 파일 시스템의 파일 시스템 및 해당 특성을 잘 알고 있어야 합니다.

또한 다음 상황을 고려해야 합니다.

- 현재 오브젝트가 들어 있는 파일 시스템의 장점을 사용하는 어플리케이션을 사용하고 있습니까?

  일부 파일 시스템은 통합 파일 시스템 지원의 일부가 아닌 인터페이스를 지원합니다. 이러한 인터페이스를 사용하는 어플리케이션은 다른 파일 시스템으로 이동하는 오브젝트로 액세스할 수 없습니다. 예를 들어, QDLS 및 QOPT 파일 시스템은 문서 및 폴더 오브젝트에 대해 작업하기 위해 계층 파일 시스템(HFS) API 및 명령을 지원합니다. 다른 파일 시스템에 있는 오브젝트상에서 이 인터페이스를 사용할 수 없습니다.
- 오브젝트의 어떤 특성이 중요합니까?

  모든 파일 시스템이 모든 특성을 지원하지는 않습니다. 예를 들어, QSYS.LIB 또는 독립 ASP QSYS.LIB 파일 시스템은 소수의 확장 속성만을 저장하고 검색하도록 지원하는 반면 『루트』(/) 및 QOpenSys 파일 시스템은 모든 확장 속성을 저장하고 검색하도록 지원합니다. 따라서 QSYS.LIB 및 독립 ASP QSYS.LIB는 확장 속성을 가지는 오브젝트를 저장하기에 적합하지 않습니다.

  이동하기 좋은 파일은 QDLS에 저장된 PC 파일입니다. 대부분의 PC 어플리케이션은 QDLS에서 다른 파일 시스템으로 이동되는 PC 파일에 대한 작업을 계속할 수 있어야 합니다. ″루트″(/), QOpenSys 및 QNTC 파일 시스템은 이러한 PC 파일을 저장하는 데 적합합니다. 이 파일 시스템들은 많은 OS/2 파일 시스템 특성을 지원하므로 파일에 대해 더 빠른 액세스를 제공할 수 있습니다.

System i Navigator에서 오브젝트를 새 위치로 드래깅하여 파일 또는 폴더를 다른 파일 시스템으로 이동할 수 있습니다. 또한, 복사하여 붙여넣기 및 잘라내어 붙여넣기를 사용하여 이 작업을 수행할 수 있습니다.

CL 명령을 사용하여 다른 파일 시스템으로 오브젝트를 이동하려면 다음 단계를 수행하십시오.

1. 이동시키려는 모든 오브젝트 사본을 저장하십시오.

   백업을 함으로써 어플리케이션이 파일을 이동시킨 파일 시스템의 오브젝트에 액세스할 수 없을 경우, 원래의 파일 시스템으로 오브젝트를 복원할 수 있습니다.

   주: 한 파일 시스템에서 저장된 오브젝트를 다른 파일 시스템에서 복원할 수는 없습니다.

2. CRTDIR(디렉토리 작성) 명령을 사용하여 오브젝트를 이동하려는 파일 시스템에 디렉토리를 작성하십시오.

   작성되는 디렉토리에 이 속성을 복사할 것인지를 결정하려면, 현재 오브젝트가 있는 디렉토리 속성을 주의 깊게 조사해야 합니다. 예를 들면, 디렉토리를 작성하는 사용자가 이전 디렉토리를 소유한 사용자가 아닌 소유자인 경우입니다. 파일 시스템이 디렉토리 소유자 설정을 지원하는 경우, 디렉토리 작성 후 디렉토리의 소유권을 이전시킬 수 있습니다.

3. MOV(이동) 오브젝트 명령을 사용하여 선택한 파일 시스템으로 파일을 이동시키십시오.

   파일 시스템이 오브젝트 소유권 설정을 지원하는 경우 MOV가 오브젝트 소유권을 보존하므로 MOV가 권장됩니다. 그러나 OWNER(*KEEP) 매개변수를 사용하여 오브젝트 소유권을 보존하려면 CPY(복사) 오브젝트 명령을 사용할 수 있습니다. 이것은 오브젝트 소유자 설정을 지원하는 파일 시스템에 대해서만 작동한다는 점을 기억하십시오. MOV 또는 CPY를 사용할 때 다음을 주의하십시오.

   - 속성이 일치하지 않거나 삭제될 수 있습니다.
   - 확장 속성이 삭제될 수 있습니다.
   - 권한이 동등하지 않거나 삭제될 수 있습니다.

   이는 원래의 파일 시스템으로 오브젝트를 리턴하기로 한 경우, 삭제된 속성과 권한으로 인해 오브젝트를 이동하거나 복사하지 않을 수도 있다는 것입니다. 오브젝트를 리턴하는 가장 안전한 방법은 저장했던 버전을 복원하는 것입니다.

   관련 개념

   26 페이지의 『파일 시스템』

   *파일 시스템*은 논리 장치(LU)를 구성하는 요소인 기억장치의 특정 세그먼트에 액세스할 수 있도록 지원합니다. 시스템에서 논리 장치는 파일, 디렉토리, 라이브러리 및 오브젝트입니다.

   관련 참조

   27 페이지의 『파일 시스템 비교』

   다음 표는 각 파일 시스템의 기능 및 제한사항을 요약한 것입니다.

   관련 정보

   CRTDIR(디렉토리 작성) 명령

   MOV(오브젝트 이동) 명령

   COPY(오브젝트 복사) 명령

권한 설정

오브젝트에 권한을 추가하면 해당 오브젝트를 조작하기 위해 다른 오브젝트의 기능을 제어할 수 있습니다. 권한을 사용하면 일부 사용자는 오브젝트를 열람하기만 하고 실제로 오브젝트 편집은 다른 사용자들이 하도록 할 수 있습니다.

파일 또는 폴더에 대한 권한을 설정하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → ***시스템*** → **파일 시스템** → **통합 파일 시스템**을 확장하십시오. 권한이 추가될 오브젝트가 나타날 때까지 계속 확장하십시오.

2. 권한을 추가할 오브젝트를 오른쪽 마우스 버튼으로 클릭하고 **권한**을 선택하십시오.
3. **권한** 대화상자에서 **추가**를 클릭하십시오.
4. **추가** 대화상자의 사용자 또는 그룹명 필드에서 하나 이상의 사용자와 그룹을 선택하거나 사용자 또는 그룹명을 입력하십시오.
5. **확인**을 클릭하십시오. 사용자 또는 그룹이 리스트 맨 위에 추가됩니다.
6. 자세한 권한을 구현하려면 **세부사항** 버튼을 클릭하십시오.
7. 해당 선택란의 상자를 체크하여 사용자에게 원하는 권한을 적용하십시오.
8. **확인**을 클릭하십시오.

텍스트 파일 변환 설정

System i Navigator에서 자동 텍스트 파일 변환을 설정할 수 있습니다. 자동 텍스트 파일 변환을 사용하면 파일 데이터 변환용 파일 확장자를 사용할 수 있습니다.

통합 파일 시스템은 System i 플랫폼 과 PC 사이에 데이터 파일이 전송될 때 이 파일을 변환할 수 있습니다. PC에서 데이터 파일에 액세스할 경우 이 파일은 ASCII 형식처럼 처리됩니다.

파일 텍스트 변환을 설정하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → ***시스템*** → **파일 시스템**을 확장하십시오.
2. **통합 파일 시스템**을 오른쪽 마우스 버튼으로 클릭하고 **등록 정보**를 선택하십시오.
3. **자동 텍스트 파일 변환을 위한 파일 확장자** 텍스트 상자에서 자동으로 변환할 파일 확장자를 입력하고 **추가**를 클릭하십시오.
4. 자동으로 변환할 모든 파일 확장자에 대해 단계 3을 반복하십시오.
5. **확인**을 클릭하십시오.

다른 시스템으로 파일 또는 폴더 송신

다른 시스템으로 파일 또는 폴더를 송신하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → ***시스템*** → **파일 시스템** → **통합 파일 시스템**을 확장하십시오. 송신할 오브젝트가 나타날 때까지 계속 확장하십시오.
2. 파일 또는 폴더를 마우스 오른쪽 버튼으로 클릭하고 **송신**을 선택하십시오. 파일 또는 폴더가 파일 송신 위치 대화상자의 선택된 파일 및 폴더 리스트에 나타납니다.
3. 사용할 수 있는 시스템 및 그룹 리스트를 확장하십시오.
4. 시스템을 선택하고 **추가**를 선택하여 시스템을 **목표 시스템 및 그룹** 리스트에 추가하십시오. 이 파일 또는 폴더를 송신할 모든 파일 시스템에 대해 이 단계를 반복하십시오.
5. 파일 또는 폴더를 송신하려면 **확인**을 클릭하십시오.

관련 태스크

『파일 또는 폴더 전송 옵션 변경』
시스템이 파일 또는 폴더를 다른 시스템으로 전송할 때 하위 폴더를 포함 여부 및 존재하는 파일 대체 여부를 정의할 수 있습니다. 시스템의 파일 또는 폴더 전송 시간을 스케줄할 수 있습니다. 파일 전송 옵션을 변경하려면 다음 단계를 수행하십시오.

파일 또는 폴더 전송 옵션 변경

시스템이 파일 또는 폴더를 다른 시스템으로 전송할 때 하위 폴더를 포함 여부 및 존재하는 파일 대체 여부를 정의할 수 있습니다. 시스템의 파일 또는 폴더 전송 시간을 스케줄할 수 있습니다. 파일 전송 옵션을 변경하려면 다음 단계를 수행하십시오.

1. 143 페이지의 『다른 시스템으로 파일 또는 폴더 송신』 단계를 완료하십시오.
2. **옵션** 탭을 클릭하십시오. 디폴트 옵션에는 파일을 패키지화하여 송신할 때 서브폴더가 포함되며, 기존 파일이 송신되고 있는 파일로 대체됩니다.
3. 필요한 옵션을 변경하십시오.
4. 확장 저장 및 복원 옵션을 설정하려면 **확장**을 클릭하십시오.
5. 확장 옵션을 저장하려면 **확인**을 클릭하십시오.
6. 파일 또는 폴더를 전송하려면 **확인**을 누르고 전송 시간을 설정하려면 **스케줄**을 누르십시오.
7. 파일 또는 폴더 송신 시기 옵션을 선택하십시오. 스케줄러 기능은 편리한 시간에 작업을 수행할 수 있도록 하는 유연성을 제공합니다.

파일 공유 작성

*파일 공유*는 System i 네트워크의 PC 클라이언트와 공유하는 i5/OS NetServer 디렉토리 경로입니다. 파일 공유는 System i 플랫폼의 임의의 통합 파일 시스템 디렉토리로 구성될 수 있습니다.

파일 공유를 작성하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → ***시스템*** → **파일 시스템** → **통합 파일 시스템**을 확장하십시오.
2. 공유를 작성할 폴더가 들어 있는 파일 시스템을 확장하십시오.
3. 공유를 작성할 폴더를 오른쪽 마우스 버튼으로 클릭하고 **공유**를 선택하십시오.
4. **신규 공유**를 선택하십시오.
5. 파일 공유 대화상자가 나타나면 새 파일 공유 속성을 지정하고 **확인**을 누르십시오.

파일 공유 변경

*파일 공유*는 System i 네트워크의 PC 클라이언트와 공유하는 i5/OS NetServer 디렉토리 경로입니다. 파일 공유는 System i 플랫폼의 임의의 통합 파일 시스템 디렉토리로 구성될 수 있습니다.

파일 공유를 변경하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → ***시스템*** → **파일 시스템** → **통합 파일 시스템**을 확장하십시오.
2. 공유가 정의된 변경할 폴더를 확장하십시오.

3. 변경하려는 파일 공유를 포함하고 있는 폴더에서 마우스 오른쪽 단추를 누르고 **공유**를 선택하십시오.
4. 변경할 파일 공유명을 선택하십시오.
5. 파일 공유 대화상자가 나타나면 파일 공유 속성을 변경하고 변경사항을 확정하기 위해 **확인**을 누르십시오.

파일 공유 제거

*파일 공유*는 i5/OS NetServer가 System i 네트워크의 PC 클라이언트와 공유하는 디렉토리 경로입니다. 파일 공유는 System i 플랫폼의 임의의 통합 파일 시스템 디렉토리로 구성될 수 있습니다. System i Navigator를 사용하여 기존의 파일 공유를 중단할 수 있습니다.

파일 공유를 제거하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → ***시스템*** → **파일 시스템** → **통합 파일 시스템**을 확장하십시오.
2. 공유를 중단하려는 파일 공유를 포함하고 있는 파일 시스템을 확장하십시오.
3. 공유를 중단하려는 공유 디렉토리에서 마우스 오른쪽 단추를 누르고 **공유** → **공유 중단**을 선택하십시오.
4. 다음 표시할 공유 중단 창에서 **확인**을 누르십시오.

새로운 사용자 정의 파일 시스템 작성

사용자 정의 파일 시스템(UDFS)은 속성을 작성하고 정의하는 파일 시스템입니다. UDFS는 시스템의 보조 기억장치 풀(ASP) 및 독립 ASP에 상주합니다.

사용자 정의 파일 시스템(UDFS)을 작성하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → ***시스템*** → **파일 시스템** → **통합 파일 시스템** → **루트** → **dev**를 확장하십시오.
2. 새로운 UDFS를 포함할 보조 기억장치 풀(ASP)을 클릭하십시오.
3. 파일 메뉴에서 **신규 UDFS**를 선택하십시오.
4. 새 사용자 정의 시스템 대화상자에서 UDFS 이름, 설명(선택적), 감사 값, 디폴트 파일 형식, 디폴트 스캔 속성, 디폴트 디스크 공간 할당, 디폴트 메모리 할당 및 새 UDFS 내 파일 이름의 대소문자 구분 여부를 지정하십시오.

   주: 디폴트 디스크 공간 할당 및 **디폴트 메모리 할당**은 V6R1 또는 이후 버전에서만 사용 가능합니다.

사용자 정의 파일 시스템 마운트

UDFS에 저장된 데이터를 보거나 액세스하려면 매 IPL 이후 UDFS를 마운트해야 합니다.

UDFS를 마운트할 때 이것은 폴더 계층에서 마운트 포인트 아래 존재하는 모든 파일 시스템, 디렉토리 또는 오브젝트를 포함합니다. 이로 인해 UDFS를 마운트 해제할 때까지 해당 파일 시스템, 디렉토리 또는 오브젝트를 액세스할 수 없게 됩니다. 통합 파일 시스템의 모든 데이터에 대한 액세스가 유지보수되도록 하려면 UDFS를 빈 폴더에 마운트하십시오. UDFS가 마운트되면 해당 폴더 내에서 UDFS 내의 폴더를 액세스할 수 있습니다. 폴더에 작성된 변경사항은 포함되는 폴더가 아니라 UDFS에 대한 변경사항입니다.

주: 독립 ASP에 있는 UDFS는 마운트될 수 없습니다.

사용자 정의 파일 시스템(UDFS)을 마운트하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → *시스템* → **파일 시스템** → **통합 파일 시스템** → **루트** → **dev**를 확장하십시오.
2. 마운트할 UDFS를 포함하는 보조 기억장치 풀(ASP)을 클릭하십시오.
3. System i Navigator의 오른쪽 분할창에 있는 **UDFS명** 열에서 마운트할 UDFS를 마우스 오른쪽 버튼으로 클릭하십시오.
4. **마운트**를 선택하십시오.
5. UDFS 마운트 대화상자가 나타나면 마운트할 디렉토리 경로, 액세스 유형(읽기 전용, 읽기 및 쓰기) 및 사용자 및 그룹 ID 설정 가능 여부를 지정하십시오. **확인**을 클릭하십시오.

끌어 놓기를 사용하려면 동일한 시스템의 통합 파일 시스템 내의 폴더로 끌어 UDFS를 마운트할 수 있습니다. /dev, /dev/QASPxx, /dev/asp_name, 다른 시스템 또는 데스크탑에 UDFS를 놓을 수 없습니다.

사용자 정의 파일 시스템 마운트 해제

UDFS를 마운트할 때 이것은 폴더 계층에서 마운트 포인트 아래 존재하는 모든 파일 시스템, 디렉토리 또는 오브젝트를 포함합니다. 이로 인해 UDFS를 마운트 해제할 때까지 해당 파일 시스템, 디렉토리 또는 오브젝트를 액세스할 수 없게 됩니다.

사용자 정의 파일 시스템(UDFS)을 마운트 해제하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → *시스템* → **파일 시스템** → **통합 파일 시스템** → **루트** → **dev**를 확장하십시오.
2. 마운트 해제할 UDFS를 포함하는 보조 기억장치 풀(ASP)을 클릭하십시오.
3. System i Navigator의 오른쪽 분할창에 있는 **이름** 열에서 마운트 해제할 UDFS를 마우스 오른쪽 버튼으로 클릭하십시오.
4. **마운트 해제**를 선택하십시오. 마운트 해제할 UDFS를 표시하는 마운트 해제 확인 패널이 나타납니다.
5. 마운트 해제를 원하지 않는 UDFS는 선택을 해제합니다. 그런 다음 **마운트 해제**를 클릭하십시오.

동적으로 마운트된 파일 시스템에 대한 작업

동적 마운트 정보 기능을 사용하여 어떤 동적 마운트 파일 시스템이 현재 마운트되어 있는 동적 마운트 파일 시스템 및 이것의 등록정보를 보고 필요 시 마운트 해체할 수 있습니다.

이 기능을 사용하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → *시스템* → **파일 시스템** → **통합 파일 시스템**을 확장하십시오.
2. **통합 파일 시스템**을 오른쪽 마우스 버튼으로 클릭하십시오.
3. 팝업 메뉴에서 **동적 마운트 정보**를 선택하십시오.
4. 현재 마운트된 모든 파일 시스템을 리스트하고 있는 동적 마운트 정보 창이 열립니다. 이 창은 마운트된 파일 시스템의 이름, 파일 시스템이 마운트된 리모트 시스템 및 마운트 유형을 표시합니다. 지원되는 마운

트 유형은 사용자 정의 파일 시스템(UDFS), Network File System 버전 2(NFSv2), Network File System 버전 3(NFSv3) 및 Network File System 버전 4(NFSv4)입니다. 나열된 파일 시스템 중 특정 파일 시스템의 등록정보를 보고 마운트 해제할 수 있습니다.

주: NFSv4는 V6R1부터 사용 가능합니다.

- 파일 시스템을 마운트 해제하려면 목록에서 해당 파일 시스템을 선택하고 **마운트 해제**를 누르십시오. 마운트 해제 확인 창이 열립니다. 표시된 파일 시스템이 마운트 해제하려는 것인지 확인하십시오. 마운트 해제하기 원하지 않는 항목은 선택을 해제합니다. 그런 다음, 조작을 확인하려면 **마운트 해제**를 클릭하십시오.
- 파일 시스템의 등록정보를 보려면 목록에서 해당 파일 시스템을 선택하고 **등록정보**를 누르십시오. 마운트 등록정보 창이 열립니다.
  - 사용자 정의 파일 시스템(UDFS)의 마운트 등록정보 창은 **일반** 탭을 포함하고 있습니다. 이것은 다음 등록정보를 표시합니다: 이름, UDFS가 마운트된 경로, 마운트 유형, 마운트 시간, 파일 시스템의 읽기 전용 여부, 사용자 및 그룹 설정 가능 여부.
  - NFS(Network File System)의 마운트 등록정보 창은 **일반** 탭 및 **고급** 탭을 포함하고 있습니다. **일반** 탭은 다음 등록정보를 표시합니다: 이름, NFS가 마운트된 경로, 마운트 유형, 마운트 시간, 파일 시스템의 읽기 전용 여부, 사용자 및 그룹 설정 가능 여부. **고급** 탭은 다음 등록정보를 표시합니다: 마운트 유형, 시간종료 값, 읽기 버퍼 크기, 쓰기 버퍼 크기, 재시도 가능 횟수, 일반 오브젝트 속성 최소 및 최대 시간, 폴더 속성 최소 및 최대 시간, 속성 새로 고침 강제실행 여부, 속성 및 이름 캐싱 가능 여부.

오브젝트가 스캔되어야 하는지 여부 설정

"루트"(/), QOpenSys 및 사용자 정의 파일 시스템에서 오브젝트를 스캔할 수 있는지 여부를 지정할 수 있습니다. 이 스캔 옵션 설정을 하려면 다음 단계를 수행하십시오.

1. System i Navigator에서 **내 연결** → *시스템* → **파일 시스템** → **통합 파일 시스템**을 확장하십시오. 관심이 있는 오브젝트가 나타날 때까지 계속 확장하십시오.
2. 폴더나 파일을 마우스 오른쪽 버튼으로 클릭하고 **등록 정보**를 선택하십시오.
3. **보안** 탭을 선택하십시오.
4. 원하는 옵션과 함께 **오브젝트 스캔**을 선택하십시오.

옵션에 대한 자세한 정보는 다음 섹션을 참조하십시오. 이 옵션에 대한 설명은 파일용입니다. 파일만 스캔될 수 있습니다. 폴더 및 사용자 정의 파일 시스템을 사용하여 그 폴더나 사용자 정의 파일 시스템에 작성된 파일에 지정되어야 하는 스캔 속성을 지정할 수 있습니다.

- 예

  오브젝트가 수정되었거나 마지막으로 오브젝트가 스캔된 이후 스캐닝 소프트웨어가 갱신된 경우 오브젝트는 스캔 관련 종료 프로그램에서 설명하는 규칙에 따라 스캔됩니다.
- 아니오

오브젝트는 스캔 관련 종료 프로그램에 의해 스캔되지 않습니다.

주: 이 속성을 가지는 오브젝트가 복원될 때 시스템 값에 오브젝트가 복원된 이후 다음 액세스 시 스캔 옵션이 선택되면 오브젝트는 복원 후 최소 한 번 스캔됩니다.

- 오브젝트가 변경된 경우에만

  마지막으로 오브젝트가 스캔된 이후 오브젝트가 수정된 경우 오브젝트는 스캔 관련 종료 프로그램에서 설명하는 규칙에 따라 스캔됩니다. 스캐닝 소프트웨어가 갱신된 경우 오브젝트는 스캔되지 않습니다.

  제어 스캔으로 오브젝트가 변경된 경우에만 속성 사용 옵션이 시스템 값으로 지정되지 않으면 이 오브젝트 변경만 속성은 사용되지 않으며 오브젝트가 수정된 이후와 스캔 소프트웨어가 갱신을 표시할 때 오브젝트가 스캔됩니다.

주:

1. 파일에 대한 이 탭에서 오브젝트의 스캔 상태를 결정할 수도 있습니다.
2. 이 속성을 가지는 오브젝트가 복원될 때 시스템 값에 오브젝트가 복원된 이후 다음 액세스 시 스캔 옵션이 선택되면 오브젝트는 복원 후 최소 한 번 스캔됩니다.

오브젝트 체크 인

팝업 메뉴에서 **체크 인** 옵션을 사용하거나 **등록정보** 페이지를 사용하여 폴더 내 파일 또는 모든 체크 인 가능 오브젝트를 체크 인할 수 있습니다.

다음 요구사항을 만족시키는 오브젝트를 체크 인할 수 있습니다.

- 오브젝트 체크 인 명령(CHKIN)이 해당 오브젝트 유형을 지원해야 합니다.
- 현재 오브젝트가 체크 아웃된 상태입니다.

팝업 메뉴에 오브젝트를 체크 인하려면 다음 단계를 따릅니다.

주: 이 방법은 System i Navigator V6R1 또는 그 이후 버전에서만 사용 가능합니다. 이전 버전에서는 **등록정보** 페이지 방법을 사용합니다.

1. System i Navigator에서 **내 연결** → *시스템* → **파일 시스템** → **통합 파일 시스템**을 확장하십시오.
2. 체크 인하고자 하는 파일 또는 체크 인하려는 모든 것을 포함하고 있는 폴더에서 마우스 오른쪽 단추을 누르십시오.
3. 팝업 메뉴에서 **체크 인**을 선택하십시오.

등록정보 페이지로 오브젝트를 체크 인하려면 다음 단계를 따릅니다.

1. System i Navigator에서 **내 연결** → *시스템* → **파일 시스템** → **통합 파일 시스템**을 확장하십시오.
2. 체크 인하고자 하는 파일 또는 체크 인하려는 모든 것을 포함하고 있는 폴더에서 마우스 오른쪽 단추를 누르십시오.

   주: 폴더 내 모든 것을 체크 인하는 기능은 V6R1 또는 이후 버전에서만 사용 가능합니다.

3. 팝업 메뉴에서 **등록정보**를 선택하십시오.
4. 열린 **등록정보** 창에서 **사용** 탭을 누릅니다.
5. 파일 또는 폴더 내 모든 오브젝트를 체크 인하십시오.
   - 파일을 체크 인하려면 **체크 인**을 누릅니다.
   - 폴더 내 모든 오브젝트를 체크 인하려면 **체크 인**을 누릅니다. 확인 창이 열립니다. 체크 인 작업을 계속해서 진행하려면 **계속**을 누릅니다. 이 작업은 체크 인하는 오브젝트의 수에 따라 완료될 때까지 긴 시간이 걸릴 수 있습니다.

오브젝트 체크 아웃

팝업 메뉴에서 **체크 아웃** 옵션을 사용하거나 **등록정보** 페이지를 사용하여 폴더 내 파일 또는 모든 체크 아웃 가능 오브젝트를 체크 인할 수 있습니다.

다음 요구사항을 만족시키는 오브젝트를 체크 아웃할 수 있습니다.

- 오브젝트 체크 아웃 명령(CHKOUT)이 해당 오브젝트 유형을 지원해야 합니다.
- 현재 오브젝트가 체크 인된 상태입니다.

팝업 메뉴에 오브젝트를 체크 아웃하려면 다음 단계를 따릅니다.

주: 이 방법은 System i Navigator V6R1 또는 그 이후 버전에서만 사용 가능합니다. 이전 버전에서는 **등록정보** 페이지 방법을 사용합니다.

1. System i Navigator에서 **내 연결** → ***시스템*** → **파일 시스템** → **통합 파일 시스템**을 확장하십시오.
2. 체크 아웃하려는 파일 또는 체크 아웃하려는 모든 것을 포함하고 있는 폴더에서 마우스 오른쪽 단추를 누르십시오.
3. 팝업 메뉴에서 **체크 아웃**을 선택하십시오.

등록정보 페이지로 오브젝트를 체크 아웃하려면 다음 단계를 따릅니다.

1. System i Navigator에서 **내 연결** → ***시스템*** → **파일 시스템** → **통합 파일 시스템**을 확장하십시오.
2. 체크 아웃하려는 파일 또는 체크 아웃하려는 모든 것을 포함하고 있는 폴더에서 마우스 오른쪽 단추를 누르십시오.

   주: 폴더 내 모든 것을 체크 아웃하는 기능은 V6R1 또는 이후 버전에서만 사용 가능합니다.
3. 팝업 메뉴에서 **등록정보**를 선택하십시오.
4. 열린 **등록정보** 창에서 **사용** 탭을 누릅니다.
5. 파일 또는 폴더 내 모든 오브젝트를 체크 아웃하십시오.
   - 파일을 체크 아웃하려면 **체크 아웃**을 누릅니다.
   - 폴더 내 모든 오브젝트를 체크 아웃하려면 **체크 아웃**을 누릅니다. 확인 창이 열립니다. 체크 아웃 작업을 계속해서 진행하려면 **계속**을 누릅니다. 이 작업은 체크 아웃하는 오브젝트의 수에 따라 완료될 때까지 긴 시간이 걸릴 수 있습니다.

전송-독립 리모트 프로시듀어 호출

Sun Microsystems에서 개발된 리모트 프로시듀어 호출(RPC)은 클라이언트 어플리케이션을 서버 메카니즘으로부터 쉽게 분리하고 분산시킵니다.

RPC에는 여러 유형의 기계가 전송된 데이터에 액세스할 수 있게 하는 데이터 표시의 표준(외부 데이터 표시(XDR)라고도 함)이 포함됩니다. 전송-독립 RPC(TI-RPC)는 RPC의 최신 버전입니다. 이 버전은 한 프로토콜에서 다른 프로토콜로 무리없는 전환을 제공하면서, 네트워크 계층에 사용되는 기본 프로토콜을 분리하는 방법을 제공합니다. System i 플랫폼에서는 현재 TCP와 UDP 프로토콜만 사용할 수 있습니다.

네트워크 간에 걸친 분산 어플리케이션의 개발은 RPC를 사용할 때 무리없는 타스크가 됩니다. 1차적인 목표는 사용자 인터페이스나 데이터 검색을 분산시키는 데 중점을 두고 있는 어플리케이션입니다.

네트워크 선택 API

다음 API는 어플리케이션이 실행되어야 하는 전송을 선택할 수 있는 수단을 제공합니다.

이 API를 위해서는 시스템에 *STMF /etc/netconfig 파일이 있어야 합니다. netconfig 파일이 /etc 디렉토리에 없는 경우, /QIBM/ProdData/OS400/RPC 디렉토리에서 해당 파일을 복사해야 합니다. netconfig 파일은 항상 /QIBM/ProdData/OS400/RPC 디렉토리에 있습니다.

API	설명
endnetconfig()	netconfig 파일에 저장된 레코드 포인터를 해제시킵니다.
freenetconfigent()	호출에서 getnetconfigent() 함수로 리턴된 netconfig 구조를 해제시킵니다.
getnetconfig()	netconfig 파일에서 현재 레코드로 포인터를 리턴시킨 후 그 포인터를 다음 레코드에 대해 증분시킵니다.
getnetconfigent()	포인터를 입력 netid에 해당하는 netconfig 구조로 리턴시킵니다.
setnetconfig()	netconfig 파일에서 첫 번째 항목에 대한 레코드 포인터를 초기화시킵니다. setnetconfig() 함수는 getnetconfig() 함수가 처음으로 사용되기 전에 사용되어야 합니다. setnetconfig() 함수는 getnetconfig() 함수가 사용할 고유 핸들(netconfig 파일에 저장된 레코드에 대한 포인터)을 리턴합니다.

관련 정보

API 파인더

이름 대 주소 변환 API

다음 API는 어플리케이션이 전송 독립 방식으로 서비스 또는 지정된 호스트의 주소를 확보할 수 있도록 합니다.

API	설명
netdir_free()	이름 대 주소 변환 API가 할당한 구조를 해제시킵니다.
netdir_getbyaddr()	주소를 호스트명과 서비스명으로 맵핑시킵니다.
netdir_getbyname()	서비스 매개변수에서 지정된 호스트명과 서비스명을 netconfig 구조에서 식별된 전송과 일치하는 주소 세트에 맵핑시킵니다.

API	설명
netdir_options()	TCP 및 UDP의 브로드캐스트 주소와 예약된 포트 기능과 같은 전송 고유 기능과의 인터페이스를 제공합니다.
netdir_sperror()	이름 대 주소 변환 API가 실패한 이유를 설명하는 정보용 메세지를 발행합니다.
taddr2uaddr()	전송 고유(로컬) 주소를 전송-독립(범용) 주소로 변환합니다.
uaddr2taddr()	전송 독립(범용) 주소를 전송 고유(로컬) 주소(netbuf 구조)로 변환합니다.

관련 정보

API 파인더

외부 데이터 표시(XDR) API

다음 API는 RPC 어플리케이션이 다른 호스트의 바이트 순서나 구조 배치 규약과 관계없이 임의 데이터 구조를 처리할 수 있도록 합니다.

API	설명
xdr_array()	가변 길이 배열과 해당 외부 표시간에 변환시키는 필터 원어(primitive). 이 함수는 배열의 각 요소들을 코드화거나 해독하기 위해 호출됩니다.
xdr_bool()	Boolean(C 정수)과 해당 외부 표시간에 변환시키는 필터 원어 데이터를 코드화할 때 이 필터의 값은 1 또는 0이 됩니다.
xdr_bytes()	계수된 바이트 배열과 해당 외부 표시 간에서 변환시키는 필터 원어 이 함수는 배열 요소의 크기가 1로 알려지거나 각 요소의 외부 설명이 내장된 총칭 배열의 서브세트로 취급합니다. 바이트 순서의 길이는 명시적으로 부호없는 정수에 위치합니다. 바이트 순서는 널 문자로 끝나지 않습니다. 바이트의 외부 표시는 내부 표시와 같습니다.
xdr_char()	C 언어 문자와 해당 외부 표시간에 변환시키는 필터 원어
xdr_double()	C 언어 배정밀도 수와 해당 외부 표시간에 변환시키는 필터 원어
xdr_double_char()	C 언어 2 바이트 문자와 해당 외부 표시간에 변환시키는 필터 원어
xdr_enum()	C 언어 열거(enum)와 해당 외부 표시간에 변환시키는 필터 원어
xdr_free()	전달된 포인터가 가리키는 오브젝트를 반복적으로 해제시킵니다.
xdr_float()	C 언어 부동 소수점 수(표준화된 단일 부동 소수점 수)와 해당 외부 표시간에 변환시키는 필터 원어
xdr_int()	C 언어 정수와 해당 외부 표시간에 변환시키는 필터 원어
xdr_long()	C 언어 긴 정수와 해당 외부 표시간에 변환시키는 필터 원어
xdr_netobj()	가변 길이 opaque 데이터와 해당 외부 표시간에 변환시키는 필터 원어
xdr_opaque()	고정 길이 opaque 데이터와 해당 외부 표시 간에 변환시키는 필터 원어
xdr_pointer()	구조 안에서 추적하기 위한 포인터를 제공하고 널 포인터를 차례로 나열합니다. 2진 트리나 연결 리스트 등과 같은 순환 데이터 구조를 표시할 수 있습니다.
xdr_reference()	구조 안에서 추적하기 위한 포인터를 제공하는 필터 원어 이 원어는 다른 구조에서 참조되는 한 구조 안에서 어느 포인터든지 차례로 나열하거나 역으로 나열하거나 해제시킬 수 있도록 합니다. xdr_reference() 함수는 일련화 중에 널(null) 포인터에 특별한 의미를 추가하지 않지만 널(null) 포인터의 주소로 전달하는 것이 메모리 오류의 원인이 될 수 있습니다. 따라서, 프로그래머는 반드시 두 면의 구분 집합으로 데이터를 설명해야 합니다. 한 면은 포인터가 유효할 때 사용하기 위한 것이고, 다른 면은 포인터가 널일 때 사용하기 위한 것입니다.
xdr_short()	C 언어 짧은 정수와 해당 외부 표시간에 변환시키는 필터 원어

API	설명
xdr_string()	C 언어 스트링과 해당 외부 표시간에 변환시키는 필터 원어
xdr_u_char()	부호없는 C 언어 문자와 해당 외부 표시간에 변환시키는 필터 원어
xdr_u_int()	C 언어 부호없는 정수와 해당 외부 표시간에 변환시키는 필터 원어
xdr_u_long()	C 언어 부호없는 긴 정수와 해당 외부 표시간에 변환시키는 필터 원어
xdr_u_short()	C 언어 부호없는 짧은 정수와 해당 외부 표시간에 변환시키는 필터 원어
xdr_union()	구분되는 C 집합과 해당 외부 표시간에 변환시키는 필터 원어
xdr_vector()	고정 길이 배열과 해당 외부 표시간에 변환시키는 필터 원어
xdr_void()	매개변수가 없습니다. 매개변수를 필요로 하는 다른 RPC 함수로 전달되지만 데이터를 전송하지는 않습니다.
xdr_wrapstring()	xdr_string(xdr, sp, maxuint) API를 호출하는 원어, 여기서 maxuint는 부호 없는 정수의 최대값입니다. xdr_wrapstring()은 RPC 패키지가 매개변수로 최대 두 개의 XDR 함수를 전달하므로 유용한 반면 xdr_string() 함수에는 세 개가 필요합니다.

관련 정보

API 파인더

인증 API

다음 API는 TI-RPC 어플리케이션에 인증을 제공합니다.

API	설명
auth_destroy()	auth 매개변수에서 가리키는 인증 정보 구조를 파기시킵니다.
authnone_create()	각 리모트 프로시듀어 호출과 함께 널 인증 정보를 전달하는 디폴트 RPC 인증 핸들을 작성 및 리턴시킵니다.
authsys_create()	인증 정보를 가진 RPC 인증 핸들을 작성 및 리턴시킵니다.

관련 정보

API 파인더

전송 독립 RPC(TI-RPC) API

다음 API는 특정 전송 기능으로부터 어플리케이션을 분리시켜 분산 어플리케이션 개발 환경을 제공합니다. 이것은 전송에 사용상의 편리함을 추가한 것입니다.

관련 정보

API 파인더

TI-RPC 단순 APIs

다음 단순 API는 사용할 전송 유형을 지정합니다. 이 레벨을 사용하는 어플리케이션은 명시적으로 핸들을 작성하지 않아도 됩니다.

API	설명
rpc_call()	지정 시스템에 리모트 프로시듀어를 호출합니다.
rpc_reg()	RPC 서비스 패키지와 함께 프로시듀어를 등록합니다.

관련 정보

API 파인더

TI-RPC 최고 레벨 API

다음 API는 어플리케이션이 전송 유형을 지정할 수 있도록 합니다.

API	설명
clnt_call()	클라이언트와 연관된 리모트 프로시듀어를 호출합니다.
clnt_control()	클라이언트 오브젝트에 관한 정보를 변경합니다.
clnt_create()	일반 클라이언트 핸들을 작성합니다.
clnt_destroy()	클라이언트의 RPC 핸들을 파기합니다.
svc_create()	서버 핸들을 작성합니다.
svc_destroy()	RPC 서비스 전송 핸들을 파기합니다.

관련 정보

API 파인더

TI-RPC 중간 레벨 API

다음 API는 최고 레벨 API와 유사하지만 사용자 어플리케이션에서 네트워크 선택 API를 사용하여 전송 고유 정보를 선택합니다.

API	설명
clnt_tp_create()	클라이언트 핸들을 작성합니다.
svc_tp_create()	서버 핸들을 작성합니다.

관련 정보

API 파인더

TI-RPC 최고 레벨 API

다음 API는 어플리케이션이 사용할 전송을 선택할 수 있게 합니다. 또한 CLIENT와 SVCXPRT 핸들의 세부 사항에 있어서 증가된 제어 레벨도 제공합니다. 이들 API는 이름 대 주소 변환 API를 사용함으로써 제공되는 추가 제어를 가진 중간 레벨 API와 유사합니다.

이름 대 주소 변환 API를 사용함으로써 제공되는 추가 제어는 다음과 같습니다.

API	설명
clnt_tli_create()	클라이언트 핸들을 작성합니다.
rpcb_getaddr()	서비스의 범용 주소를 찾습니다.
rpcb_set()	RPCbind와 함께 서버 주소를 등록합니다.
rpcb_unset()	주소 등록을 취소하기 위해 서버에 의해 사용됩니다.
svc_reg()	프로그램과 버전을 디스패치와 연관시킵니다.
svc_tli_create()	서버 핸들을 작성합니다.
svc_unreg()	svc_reg()에 의해 설정된 연관을 삭제합니다.

관련 정보

API 파인더

기타 TI-RPC API

이 API는 여러 어플리케이션이 단순, 최고 레벨, 중간 레벨, 전문가 레벨 API와 협조하여 작업할 수 있도록 합니다.

API	설명
clnt_freeres()	RPC나 XDR 시스템에서 할당한 데이터를 해제시킵니다.
clnt_geterr()	클라이언트 핸들로부터 오류 구조를 가져옵니다.
svc_freeargs()	RPC나 XDR 시스템에서 할당한 데이터를 해제시킵니다.
svc_getargs()	RPC 요구 인수를 해독합니다.
svc_getrpccaller()	호출자의 네트워크 주소를 가져옵니다.
svc_run()	RPC 요구가 도착하기를 기다립니다.
svc_sendreply()	리모트 클라이언트로 프로시듀어 호출 결과를 송신합니다.
svcerr_decode()	오류 해독을 위해 클라이언트로 정보를 송신합니다.
svcerr_noproc()	프로시듀어 번호 오류에 대해 클라이언트로 정보를 송신합니다.
svcerr_systemerr()	시스템 오류에 대해 클라이언트로 정보를 송신합니다.

관련 정보

API 파인더

통합 파일 시스템 관련 정보

제품 매뉴얼 및 다른 Information Center 주제 콜렉션은 이 통합 파일 시스템 주제 콜렉션과 연관된 정보를 가지고 있습니다. 임의 PDF 파일을 보거나 인쇄할 수 있습니다.

매뉴얼

- i5/OS Network File System Support (2105KB)
  이 책은 일련의 실제 어플리케이션을 통해 네트워크 파일 시스템에 대해 설명합니다. 여기에서는 내보내기, 마운트, 파일 잠금, 보안 고려사항등을 다루도 있습니다. 이 서적을 통해서 보안 네트워크 이름 공간을 구성하고 개발하기 위한 NFS의 사용법을 배울 수 있습니다.

- WebSphere Development Studio: ILE C/C++ Language Reference (4490KB)
  이 책은 System i 플랫폼에서 ILE C 프로그램을 설계, 편집, 컴파일, 실행 및 디버그하는 데 필요한 정보를 제공합니다.

- APPC Programming (1497KB)
  이 책은 System i 플랫폼에 대한 APPC(Advanced Program-to-Program Communication) 지원에 대해 설명합니다. APPC를 사용하는 어플리케이션 프로그램 개발 및 APPC용 통신 환경 정의 안내가 나와 있습니다.

- Recovering your system  (8404KB)
  이 책은 System i 플랫폼의 복구 및 가용성 옵션에 대한 일반 정보를 제공합니다.

기타 정보

- **Experience Report**

  Experience Report는 IBM 개발자가 기록한 것이며 실제 시나리오 및 솔루션을 구현한 본인의 실제 경험을 문서화한 것입니다. System i 솔루션(단계별 지침 및 추가 정보를 완비한)의 특정 구현과 함께 IBM 개발자의 경험을 따르려면 이 보고서를 사용하십시오. Experience Report Backing up the integrated file system은 파일 및 파일 시스템과 관련되어 있습니다.
- 제어 언어(CL)
- i5/OS 국제화
- API(Application Programming Interface)
- 저널 관리
- 확약 제어
- 보안 참조

코드 라이센스 및 면책사항 정보

IBM은 사용자가 자신의 특정 필요에 맞게 유사한 기능을 생성하는 데 모든 프로그래밍 코드 예를 사용할 수 있도록 비배타적 저작권을 부여합니다.

배제할 수 없는 합법적인 보증에 근거하여 IBM, 해당 프로그램 개발자 및 공급자는 프로그램 또는 기술 지원이 있는 경우 이와 관련하여 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증도 하지 않습니다.

어떠한 경우에도 IBM, 해당 프로그램 개발자 또는 공급자는 그 가능성을 통지받았더라도 다음 사항에 대해 책임을 지지 않습니다.

1. 데이터 손실 또는 손상
2. 직접적이고, 특수한, 우발적이거나 간접 손상 또는 경제적으로 수반되는 손상 또는
3. 손실된 수익, 비즈니스, 수입, 신용 또는 예상되는 절약

일부 법령에서는 직접적이고, 우발적이거나 간접 손상의 배제 또는 제한을 허용하지 않으므로 위 제한사항 또는 배제사항의 일부 또는 전부가 사용자에게 적용되지 않을 수 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 담당자에게 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, 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 데이터는 본 IBM 제품 데이터의 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 라이센스 프로그램 및 이 프로그램에 대해 사용 가능한 모든 라이센스가 있는 데이터는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA), IBM 시스템 코드의 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 데이터 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 데이터 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.

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

프로그래밍 인터페이스 정보

이 통합 파일 시스템 서적은 고객이 IBM i5/OS의 서비스를 확보할 수 있도록 프로그래밍 인터페이스를 문서화합니다.

상표

다음 용어는 미국 또는 기타 국가에서 사용되는 IBM Corporation의 등록상표입니다.

DB2
i5/OS
IBM
IBM(로고)
Integrated Language Environment
NetServer
OS/2
OS/400
System i
System x
WebSpherexSeries

Adobe, Adobe 로고, PostScript 및 PostScript 로고는 미국 및/또는 Adobe Systems Incorporated의 등록상표 또는 상표입니다.

Linux는 미국 또는 기타 국가에서 사용되는 Linus Torvalds의 등록상표입니다.

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

UNIX는 미국 또는 기타 국가에서 사용되는 The Open Group의 등록상표입니다.

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

조건

다음 조건에 따라 본 문서를 사용할 수 있습니다.

개인적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 본 문서를 개인적, 비상업적 용도로 복제할 수 있습니다. 귀하는 IBM의 명시적 동의 없이 본 문서 또는 그 일부를 배포 또는 전시하거나 2차적 저작물을 만들 수 없습니다.

상업적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 본 문서를 귀하 사업장 내에서만 복제, 배포 및 전시할 수 있습니다. 귀하는 IBM의 명시적 동의 없이 본 문서의 2차적 저작물을 만들거나 본 문서 또는 그 일부를 복제, 배포 또는 전시 할 수 없습니다.

본 허가에서 명시적으로 부여된 경우를 제외하고, 본 문서나 본 문서에 포함된 정보, 데이터, 소프트웨어 또는 기타 지적 재산권에 대한 어떠한 허가나 라이센스 또는 권한도 명시적 또는 묵시적으로 부여되지 않습니다.

IBM은 본 문서의 사용이 IBM의 이익을 해친다고 판단되거나 위에서 언급된 지시사항이 준수되지 않는다고 판단하는 경우 언제든지 이 사이트에서 부여한 허가를 철회할 수 있습니다.

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

IBM은 본 문서의 내용에 대해 어떠한 보증도 제공하지 않습니다. 본 문서는 상품성, 무해함 및 특정 목적에의 적합성에 대한 보증을 포함하여(단, 이에 한하지 않음) 명시적이든 묵시적이든 일체의 보증 없이 ″현상태대로″ 제공됩니다.