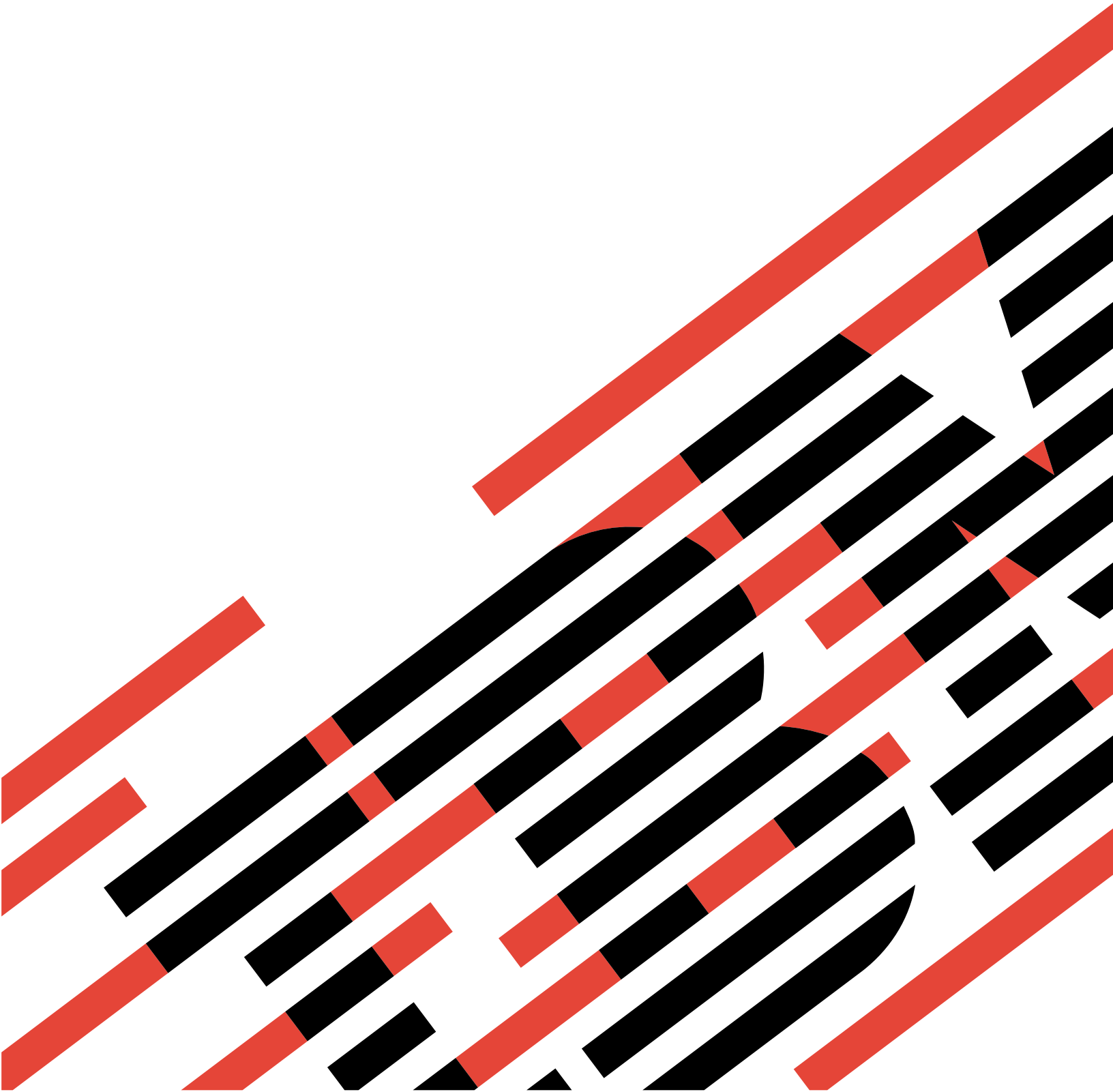
IBM

eserver

iSeries

통합 파일 시스템 소개

버전 5



IBM

@server

iSeries

통합 파일 시스템 소개

버전 5

목차

통합 파일 시스템 소개

이 서적에서는 다음과 같은 통합 파일 시스템에 대한 개요를 제공합니다.

- 통합 파일 시스템의 정의
- 통합 파일 시스템이 필요한 경우
- 통합 파일 시스템의 개념 및 용어
- 통합 파일 시스템과 관련하여 사용할 수 있는 인터페이스
- 통합 파일 시스템과 관련하여 프로그램 작성에 사용할 수 있는 기술 및 API
- 개별 파일 시스템의 특성

통합 파일 시스템 소개 서적 사용자

이 서적은 통합 파일 시스템과 사용 방법을 이해하려는 iSeries 서버 사용자, 프로그래머, 관리자를 대상으로 합니다.

코드 면책사항 관련 정보

이 문서에는 프로그래밍 예제가 들어 있습니다.

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

모든 샘플 예제는 IBM에 의해 예시 목적으로만 제공됩니다. 이러한 예제는 모든 조건하에서 철저히 테스트된 것은 아닙니다. 따라서 IBM은 이들 프로그램의 신뢰성, 실용성 또는 기능에 대해 보증할 수 없습니다.

여기에 포함된 모든 프로그램은 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 어떠한 종류의 보증 없이 ″현상태대로″ 제공됩니다.

제 1 장 통합 파일 시스템 소개

다음 주제에서는 iSeries 서버의 통합 파일 시스템에 대해 설명하고 서버에서 어떻게 사용되는지 설명합니다.

통합 파일 시스템의 정의

통합 파일 시스템은 OS/400의 일부로서 퍼스널 컴퓨터 및 UNIX 오퍼레이팅 시스템과 유사한 스트림 입/출력 및 기억장치 관리를 지원하며, 서버에 저장된 모든 정보에 대해 통합된 구조를 제공합니다.

통합 파일 시스템의 주요 피처는 다음과 같습니다.

- 긴 연속 자료열이 들어 있는 스트림 파일에 정보 저장 지원. 예를 들어, 이러한 자료열은 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 디렉토리로 경로를 지정하여 오브젝트에 액세스할 수 있습니다.
- 사용자와 어플리케이션이 스트림 파일뿐만 아니라 서버에 저장된 데이터베이스 파일, 문서 및 기타 오브젝트에 액세스할 수 있도록 하는 공통 인터페이스.
- 서버, iSeries용 통합 xSeries 서버 또는 리모트 Windows NT 서버에 로컬로 저장된 스트림 파일의 공통 보기. 또한, 스트림 파일은 근거리 통신망(LAN) 서버, Novell NetWare 서버, 다른 리모트 iSeries 서버 또는 네트워크 파일 시스템 서버에 리모트로 저장될 수 있습니다.

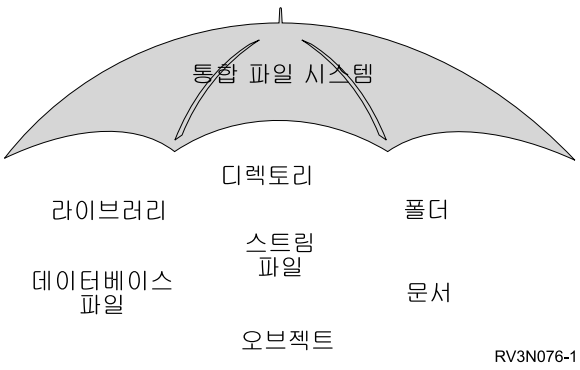


그림 1. iSeries 서버에 저장된 모든 정보의 구조

통합 파일 시스템 사용 목적

통합 파일 시스템은 OS/400의 광범위한 자료 관리 기능과 더불어 클라이언트 서버, 개방 시스템 및 멀티미디어와 같은 차세대 정보 처리 양식 지원을 향상시킨 것입니다.

통합 파일 시스템을 사용하여 다음을 수행할 수 있습니다.

- OS/400 자료(특히 OS/400 파일 서버를 사용하는 Client Access와 같은 어플리케이션의 경우)에 빠르게 액세스할 수 있습니다.
- 이미지, 오디오 및 비디오와 같은 스트림 자료를 보다 효율적으로 처리할 수 있습니다.
- POSIX(Portable Operationg System Interface for Computer Environments) 및 XPG와 같은 UNIX 기반 개방 시스템 표준을 지원하기 위한 파일 시스템 기반과 디렉토리 기반을 제공할 수 있습니다. 또한, 이 파일 구조와 디렉토리 구조는 DOS(Disk Operating System) 및 Windows 오퍼레이팅 시스템과 같은 PC 오퍼레이팅 시스템 사용자에게 익숙한 환경을 제공합니다.
- 고유한 기능(예: 레코드 지향 데이터베이스 파일, UNIX 기반 스트림 파일, 파일 서비스)을 가진 파일 지원이 별도의 파일 시스템으로 처리되도록 하면서, 이들 모두가 공통 인터페이스를 통해 관리되도록 할 수 있습니다.

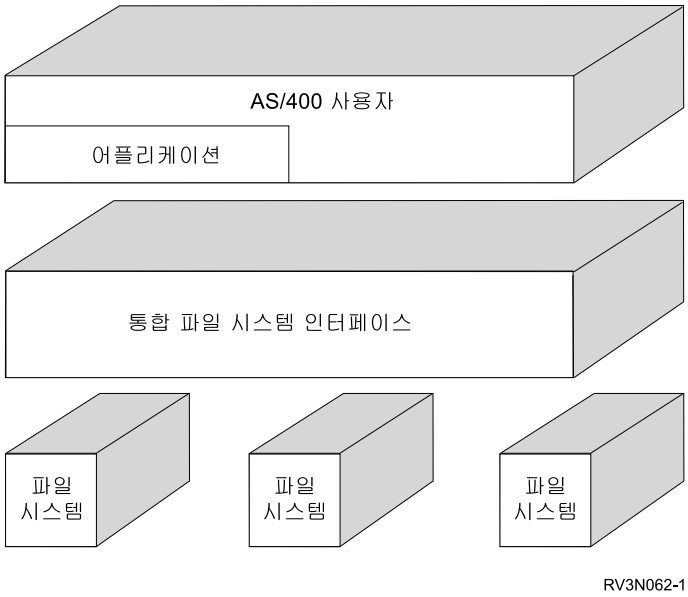


그림 2. 개별 파일 시스템에 대한 공통 인터페이스

- PC 사용자가 그래픽 사용자 인터페이스의 장점을 보다 잘 이용할 수 있도록 할 수 있습니다. 예를 들면, Windows 사용자는 Windows 그래픽 툴을 사용하여 PC에 저장된 파일에 대해 작업할 때와 동일한 방식으로 iSeries 서버 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.

제 2 장 통합 파일 시스템 개념

스트림 파일

스트림 파일은 단순하게 바이트가 나열된 구조로 랜덤 액세스가 가능합니다. 통합 파일 시스템은 스트림 파일 형식의 정보에 대한 작업 및 저장을 지원합니다. 서버의 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, iSeries 데이터베이스 파일과 비교하는 것이 도움이 됩니다. 데이터베이스 파일은 레코드를 중심으로 이루어지며, 데이터베이스 파일에는 길이 및 자료 유형과 같은 특성을 가진 하나 이상의 필드로 구성된 사전 정의 부속 영역이 있습니다.

그림 3. 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 서버의 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 자료 스트림을 저장하기에 매우 적합합니다.

통합 파일 시스템의 스트림 파일에 대한 자세한 정보는 다음을 참조하십시오.

- 49 페이지의『스트림 파일과 데이터베이스 파일 사이의 자료 복사』
- 4 페이지의『*TYPE1 및 *TYPE2 스트림 파일』

*TYPE1 및 *TYPE2 스트림 파일

파일 형식에는 *TYPE1 스트림 파일 및 *TYPE2 스트림 파일의 두 가지 옵션이 있습니다.

*TYPE1 스트림 파일의 형식은 OS/400 버전 4, 릴리스 4 이전 릴리스에서 작성된 스트림 파일과 같은 형식입니다. 이 형식은 *TYPE2 스트림 파일보다 빨리 OS/400 버전 4 릴리스 4 이전 릴리스로 저장됩니다. 이것은 최소 4096바이트를 가집니다.

*TYPE2 스트림 파일은 고성능의 파일 액세스가 가능하며 OS/400 버전 4 릴리스 4에서 처음 소개되었습니다. 이 형식은 *TYPE1 스트림 파일보다 느리게 OS/400 버전 4 릴리스 4 이전 릴리스로 저장됩니다. 이것은 최소 4096바이트의 오브젝트 크기를 가집니다. V4R4 및 그 이후 시스템에서 작성된 모든 파일은 *TYPE2 스트림 파일입니다.

*TYPE2 스트림 파일은 V4R4 및 그 이후 시스템에서만 작동하지만, *TYPE2 스트림 파일을 저장하여 V4R4 이전 시스템에서 복원할 수 있습니다. 그러나 이 프로세스는 시간이 많이 걸릴 수 있습니다.

통합 파일 시스템의 파일 시스템

파일 시스템은 논리 장치(LU)를 구성하는 요소인 기억장치의 특정 세그먼트에 액세스할 수 있도록 지원합니다. 서버에서 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. 구조와 규칙 측면에서 보면, 데이터베이스 파일과 기타 다양한 오브젝트 유형에 라이브러리를 통해 액세스할 수 있도록 지원하는 OS/400의 지원 시스템을 하나의 파일 시스템으로 생각할 수 있습니다. 마찬가지로 실제로는 스트림 파일인 문서에 폴더를 통해 액세스할 수 있도록 지원하는 OS/400 지원 시스템을 별도의 파일 시스템으로 생각할 수 있습니다.

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

각 파일 시스템의 피처와 제한사항의 비교를 보려면, 65 페이지의 『파일 시스템 비교』를 참조하십시오.

통합 파일 시스템의 파일 시스템은 다음과 같습니다.

『루트』 (/)
: ″루트″**(/)** 파일 시스템. 이 파일 시스템은 통합 파일 시스템의 계층적 디렉토리 구조와 스트림 파일 지원을 최대한 이용합니다. 루트 파일 시스템은 DOS 및 OS/2 파일 시스템의 특징을 갖습니다.

QOpenSys
: 개방 시스템 파일 시스템. 이 파일 시스템은 POSIX 및 XPG와 같은 UNIX용 개방 시스템 표준과 호환 가능합니다. 루트 파일 시스템과 같이 이 파일 시스템도 통합 파일 시스템이 제공하는 디렉토리 지원 및 스트림 파일을 이용합니다. 또한 대소문자로 구분되는 오브젝트명을 지원합니다.

UDFS
: 사용자 정의 파일 시스템. 이 파일 시스템은 사용자가 선택한 ASP(Auxiliary Storage Pool)나 독립 ASP(Auxiliary Storage Pool)에 상주합니다. 사용자가 이 파일 시스템을 작성하고 관리합니다.

QSYS.LIB

라이브러리 파일 시스템. 이 파일 시스템은 서버의 라이브러리 구조를 지원합니다. 이 파일 시스템은 시스템 및 기본 사용자 ASP에서 데이터베이스 파일과 라이브러리 지원이 관리하는 다른 모든 iSeries 서버 오브젝트 유형에 대한 액세스를 제공합니다.

독립 ASP QSYS.LIB

독립 ASP QSYS.LIB 파일 시스템. 이 파일 시스템은 사용자가 작성하고 정의한 ASP(Auxiliary Storage Pool)의 서버 라이브러리 구조를 지원합니다. 이 파일 시스템은 데이터베이스 파일과 라이브러리 지원이 관리하는 다른 모든 iSeries 서버 오브젝트 유형에 대한 액세스를 제공합니다.

QDLS 문서 라이브러리 서비스 파일 시스템. 이 파일 시스템은 문서와 폴더에 대한 액세스를 제공합니다.

QOPT

광 파일 시스템. 이 파일 시스템은 광 매체에 저장된 스트림 자료에 대한 액세스를 제공합니다.

QNetWare

QNetWare 파일 시스템. 이 파일 시스템은 Novell Netware 4.10 또는 4.11을 실행하는 서버에 저장된 로컬 또는 리모트 자료 및 오브젝트에 대한 액세스 또는 Novell NetWare 3.12, 4.10, 4.11 또는 5.0을 실행 중인 독립형 PC 서버에 대한 액세스를 제공합니다. 기존의 로컬 파일 시스템에 대해 NetWare 파일 시스템을 동적으로 마운트할 수 있습니다.

QNTC

Windows NT Server 파일 시스템. 이 파일 시스템은 Windows NT 4.0 이상을 실행중인 서버에 저장된 자료와 오브젝트에 대한 액세스를 제공합니다. iSeries 서버 어플리케이션이 Windows NT 클라이언트와 동일한 자료를 사용할 수 있도록 합니다. 여기에는 통합 PC 서버에서 실행 중인 Windows NT 서버의 자료에 대한 액세스도 포함됩니다. 자세한 내용은 Windows NT 서버와 OS/400-AS/400 통합, SC41-5439-01(SC41-5439)을 참조하십시오.

QFileSvr.400

이 파일 시스템은 리모트 iSeries 서버에 상주하는 다른 파일 시스템에 대한 액세스를 제공합니다.

NFS 네트워크 파일 시스템. 이 파일 시스템은 리모트 NFS 서버에 저장된 자료와 오브젝트에 대한 액세스를 제공합니다. 따라서, NFS 서버는 NFS 클라이언트가 동적으로 마운트시키는 네트워크 파일 시스템을 내보낼 수 있습니다.

공통 인터페이스를 통해 모든 파일 시스템과 대화식 작업을 할 수 있습니다. 이 인터페이스는 자료 관리 인터페이스를 통하여 제공되는 레코드 입출력과는 달리 스트림 자료의 입출력을 위하여 최적화되어 있습니다. 공통 인터페이스를 통한 파일 시스템과 상호작용은 제공된 명령, 메뉴 및 표시 화면과 API(어플리케이션 프로그램 인터페이스)를 통해 이루어집니다.

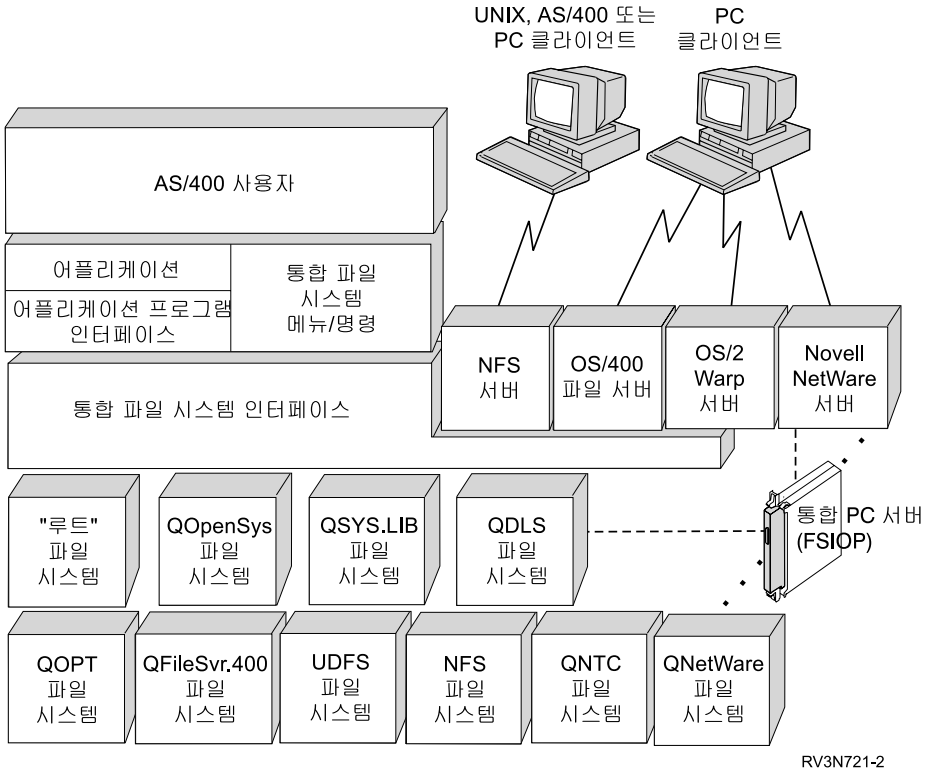


그림 4. 파일 시스템 파일 서버 및 통합 파일 시스템 인터페이스

보다 자세히 알려면 다음 주제 및 서적을 참조하십시오.

- 광 지원
- OS/400 네트워크 파일 시스템 지원

디렉토리

디렉토리는 사용자가 지정한 이름으로 오브젝트를 찾는 데 사용되는 특수 오브젝트입니다. 각 디렉토리에는 이에 연결된 오브젝트 리스트가 들어 있습니다. 그 리스트에는 다른 디렉토리도 들어 있을 수 있습니다.

통합 파일 시스템은 서버의 모든 오브젝트에 액세스할 수 있도록 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. 디렉토리 분기에는 서브디렉토리라고 불리우는 종속적 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. 라이브러리
| 의 오브젝트는 서브디렉토리의 오브젝트처럼 처리됩니다. 데이터베이스 파일에 오브젝트(데이터베이스 파일 멤

| 버)가 포함되어 있으므로, 이들 파일은 오브젝트가 아닌 서브디렉토리인 것처럼 처리됩니다. 문서 라이브러리
| 서비스 파일 시스템(QDLS 서브트리)에서 폴더는 서브디렉토리처럼 핸들되고, 폴더의 문서들은 서브디렉토리의
| 오브젝트처럼 처리됩니다.

파일 시스템간에 차이가 있으므로, 디렉토리 계층상 한 서브트리에서 수행할 수 있는 작업이 다른 서브트리에서는 작동되지 않을 수도 있습니다.

통합 파일 시스템 디렉토리 지원은 DOS 파일 시스템의 디렉토리 지원과 유사합니다. 또한, 파일을 한번만 저장하지만 링크를 사용하여 여러 경로를 통해 액세스할 수 있는 것과 같은 UNIX 시스템의 전형적인 피처를 제공합니다.

| 통합 파일 시스템 디렉토리에 대한 자세한 정보는 다음 주제를 참조하십시오.

- 8 페이지의『현재 디렉토리 및 홈 디렉토리』
- 37 페이지의『통합 파일 시스템에서 제공하는 디렉토리』
- | 9 페이지의『*TYPE2 디렉토리』

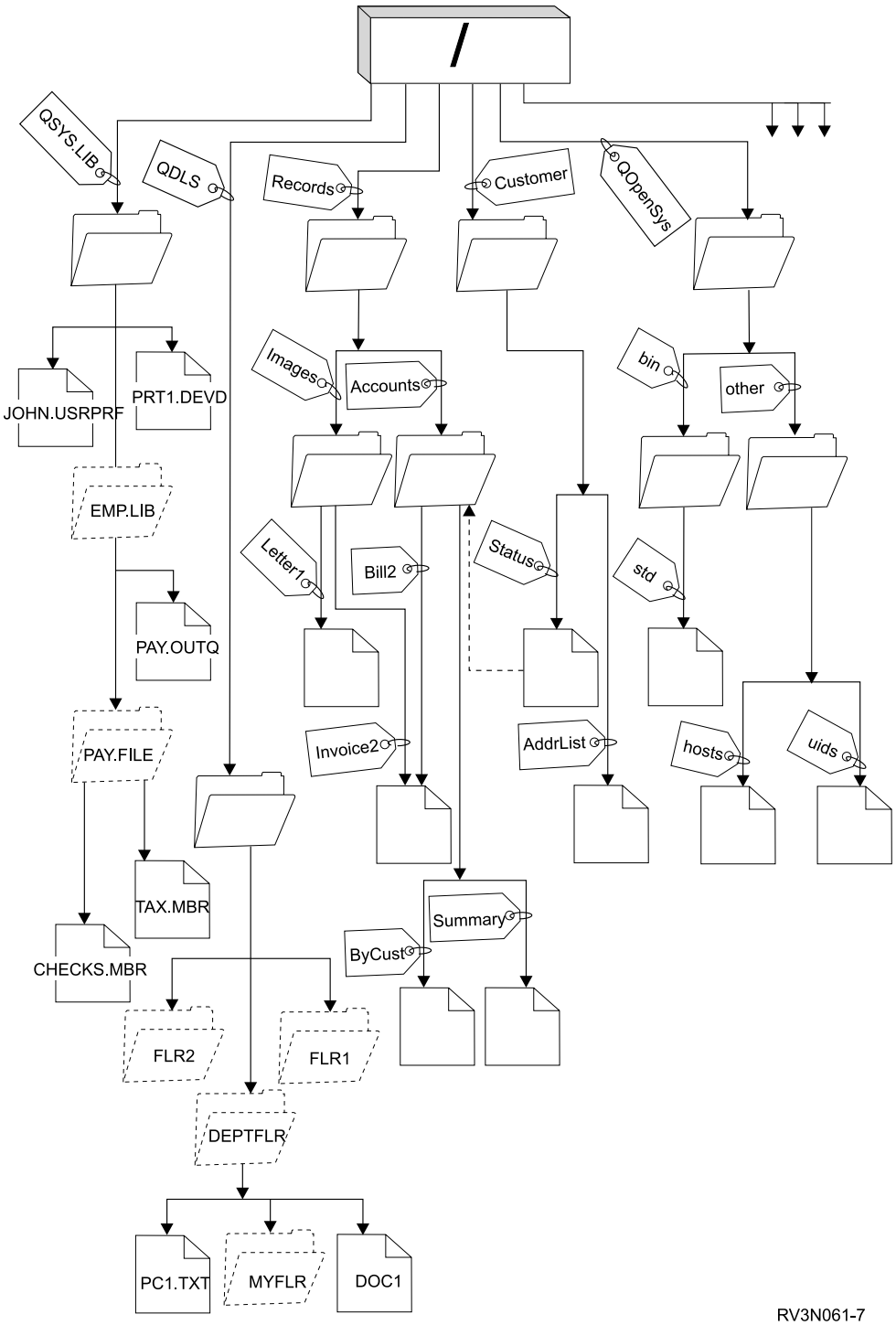


그림 5. 파일 시스템과 오브젝트는 통합 파일 시스템 디렉토리 트리상의 분기입니다.

현재 디렉토리 및 홈 디렉토리

현재 디렉토리는 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 디렉토

리 경로를 지정하지 않는 한 사용자의 현재 디렉토리에서 오브젝트를 탐색합니다. 현재 디렉토리는 현재 라이브러리 개념과 유사합니다. 또한, **현재 작업 디렉토리** 또는 **작업 디렉토리**라고도 합니다.

홈 디렉토리는 사용자가 시스템을 사인온할 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』(/)의 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.

| 현재 디렉토리 변경(CHGCURDIR) CL 명령, chdir( ) API 또는 fchdir() API를 사용하여 사인 온한 후, 언
| 제든지 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.

QWTCHGJB API에 대한 세부사항은 API(Application programming interfaces)를 참조하십시오.

| *TYPE2 디렉토리

| 통합 파일 시스템에서 ″루트″(/), QOpenSys 및 UDFS(user-defined file systems)는 *TYPE2 디렉토리 형식을 지원합니다. *TYPE2 디렉토리 형식은 기존의 *TYPE1 디렉토리 형식을 개선한 것입니다. *TYPE2 디렉토리의 내부 구조 및 구현은 *TYPE1 디렉토리와 다릅니다.

| *TYPE2 디렉토리의 장점은 다음과 같습니다.

| • 향상된 성능
| • 향상된 신뢰성
| • 추가된 기능
| • 여러 경우, 보조 기억장치 공간을 적게 사용함

*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 회복됩니다. *TYPE1 디렉토리가 완전히 회복되기 위해서는 RCLSTG(기억장치 재생) 명령을 사용해야 합니다.

*TYPE2 디렉토리는 다음과 같은 추가 기능을 제공합니다.

1. *TYPE2 디렉토리는 모노케이스 파일 시스템에서 이름의 대소문자 변경을 지원합니다(예: A를 a로 이름 변경).
2. *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 백만개의 서브디렉토리가 포함될 수 있다는 것을 의미합니다.
3. iSeries Navigator를 사용하면 *TYPE2 형식의 디렉토리를 열 때 항목 리스트가 2진 순서로 자동으로 정렬됩니다.

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.

시스템에서 *TYPE2 디렉토리를 만드는 데에는 몇 가지 방법이 있습니다.

- 독립 ASP(Auxiliary Storage Pool)의 사용자 정의 파일 시스템(UDFS)은 독립 ASP가 OS/400 V5R2가 설치된 시스템으로 처음 연결변환될 때 *TYPE2 형식으로 변환됩니다.
- UDFS를 제외한 독립 ASP의 다른 모든 지원 파일 시스템은 CVTDIR(디렉토리 변환) 명령을 사용하여 *TYPE2로 변환되어야 합니다.
- OS/400 V5R2로 사전 로드된 신규 iSeries 서버에는 *TYPE2 디렉토리가 있습니다. ASP 1-32의 ″루트″(/), QOpenSys 및 UDFS의 경우에는 변환이 필요 없습니다.
- iSeries 서버의 OS/400 V5R2 스크래치 설치에는 *TYPE2 디렉토리가 있습니다. ASP 1-32의 ″루트″(/), QOpenSys 및 UDFS의 경우에는 변환이 필요 없습니다.

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

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

주: *TYPE2 디렉토리는 OS/400 V5R1에서 지원되지만, 일반 *TYPE2 디렉토리 지원과는 약간 다릅니다. 자세한 정보는 OS/400 V5R1에서 *TYPE2 디렉토리 사용을 참조하십시오.

*TYPE2 디렉토리에 대한 자세한 정보는 다음 주제를 참조하십시오.

- *TYPE2 디렉토리로 변환
- ″루트″, QOpenSys 또는 UDFS 비가용성
- 보조 기억장치 요구사항

- 기호 링크 고려사항
- 독립 ASP(Auxiliary Storage Pool)
- 저장/복원 고려사항
- *TYPE2 변환 준비
- 변환 처리
- 예: 모든 파일 시스템(적은 수의 오브젝트) 변환
- 예: 모든 파일 시스템(많은 수의 오브젝트) 변환
- 예: 특정 ASP만 변환

OS/400 V5R1에서 *TYPE2 디렉토리 사용

통합 파일 시스템에서 ″루트″(/), QOpenSys 및 UDFS(user-defined file systems)는 OS/400 V5R1의 *TYPE2 디렉토리 형식을 지원합니다. *TYPE2 디렉토리 형식은 기존의 *TYPE1 디렉토리 형식을 개선한 것입니다. *TYPE2 디렉토리의 내부 구조는 *TYPE1 디렉토리와 다르며 성능과 신뢰성이 향상되었습니다.

V5R1을 사용하는 경우 V5R1 디렉토리를 *TYPE2 디렉토리 형식으로 변환할 수 있습니다. OS/400의 새로운 릴리스를 설치하기 **전에** *TYPE2 디렉토리 형식으로 변환하는 것이 좋습니다. 그렇지 않으면 설치하는 동안 자동으로 디렉토리 변환이 수행될 수 있습니다. 설치 중 자동 변환이 수행되면 설치에 소요되는 시간이 **상당히** 증가합니다.

주: OS/400 V5R1 또는 V5R2로 업그레이드하는 경우에는 *TYPE2 디렉토리 형식으로 자동 변환이 수행되지 **않습니다**. 설치하기 전에 디렉토리를 변환할 필요가 없습니다.

V5R1에서의 *TYPE2 디렉토리 지원은 수정 프로그램(PTF)을 통해 제공됩니다. V5R1의 변환 유틸리티는 V5R2 버전과 약간 다릅니다. V5R1의 *TYPE2 디렉토리에 대한 전체 문서는 정보용 APAR II13161을 참조하십시오. APAR에 액세스하려면 다음 방법 중 하나를 사용하십시오.

1. 정보용 APAR를 사용자 iSeries 서버로 다운로드한 후 볼 수 있습니다. 다음 명령을 사용하십시오.

   ```
   SNDPTFORD PTFID((II13161))
   DSPPTFCVR LICPGM(INFOAS4) SELECT(II13161)
   ```

2. http://www-912.ibm.com 웹 사이트에서 정보용 APAR 온라인을 볼 수 있습니다. **Authorized Program Analysis Reports(APARs) --> V5R1 APARs --> APAR 번호 II13161**을 선택하십시오.

*TYPE2 디렉토리로 변환

CVTDIR 명령은 *TYPE1 디렉토리를 *TYPE2 디렉토리로 변환합니다. 또한 이 명령은 파일 시스템을 *TYPE2 디렉토리 형식으로 변환하는 방법에 대한 정보를 제공합니다. CVTDIR은 다음을 수행합니다.

- *TYPE2 디렉토리를 지원하는 기존 파일 시스템에 대한 현재 디렉토리 형식을 나열합니다.
- 변환을 수행하는 데 걸리는 시간을 계산합니다.
- 변환에 필요한 보조 기억장치를 계산합니다.
- 파일 시스템을 *TYPE2 형식으로 변환합니다. 모든 기존 디렉토리가 *TYPE2로 변환됩니다. 변환 후에 작성되는 신규 디렉토리는 모두 *TYPE2입니다.

파일 시스템의 디렉토리가 변환되는 방법에는 여러 가지가 있습니다.

- CVTDIR 명령을 사용하여 수동으로 변환
- 독립 ASP가 OS/400 V5R2가 설치된 시스템에 처음 연결변환될 때 자동으로 변환
- 시스템이 비정상적으로 종료되었을 때 파일 시스템 변환이 진행 중이었다고 판별되는 경우 IPL 시 변환
- 누락된 *TYPE1 디렉토리가 *TYPE2 형식으로 변환된 파일 시스템의 일부인 경우 RCLSTG SELECT(*ALL)(기억 장치 재생) 중에 변환

"루트", QOpenSys 또는 UDFS 비가용성

"루트"(/) 또는 QOpenSys 파일 시스템 변환은 시스템이 제한 상태에 있을 때 수행되어야 합니다. UDFS를 변환할 경우에는 시스템이 제한 상태에 있을 필요는 없습니다. 그러나 해당 ASP의 UDFS는 변환 중에 사용할 수 없습니다. 변환 수행에 필요한 시간은 파일 시스템 *크기*에 따라 다릅니다. 따라서 변환을 수행할 최적 시기를 결정하려면 계획을 세워야 합니다. CVTDIR 명령의 *ESTIMATE 옵션은 지정된 파일 시스템을 변환하는 데 걸리는 시간을 계산합니다. 계산된 시간은 최대 예상값입니다. 단일 스레드 작업에서 실행되는 변환을 기준으로 시간을 계산합니다. 실제 변환은 복수 스레드를 사용하므로 예상 시간보다는 적게 걸립니다. 일반적으로 40,000개 이상의 링크가 있는 파일 시스템의 경우 예상 시간의 30 - 50%가 소요됩니다. 그러나 실제 시간은 서버의 구성 및 하드웨어에 따라 달라집니다.

CVTDIR 명령이 실행되는 동안 시스템이 비정상적으로 종료되면 변환 기능이 후속 IPL 시에 SCPF 작업에서 실행됩니다. SCPF 작업은 복수 스레드 활동을 허용하지 않습니다. 따라서 파일 시스템 변환을 IPL 중에 완료해야 하는 경우에는 단일 스레드를 사용하여 실행됩니다. IPL 중에 SRC C900 2A85가 표시되면 변환 기능이 실행되며 변환 진행을 나타내는 상태 메세지 CPIA089가 표시됩니다.

이미 *TYPE2로 변환된 파일 시스템에서 누락된 *TYPE1 디렉토리가 있는 경우 변환 기능은 RCLSTG 중에 실행됩니다. 변환 기능은 RCLSTG 명령을 발행하는 작업에서 실행됩니다. 변환해야 할 누락된 디렉토리가 발견되면 시스템 제한으로 인해 변환이 단일 스레드로 실행됩니다.

보조 기억장치 요구사항

파일 시스템의 디렉토리를 *TYPE2 형식으로 변환하려면 먼저 보조 기억장치 요구사항을 고려해야 합니다. 보조 기억장치 요구사항과 관련된 몇 가지 사항이 있습니다.

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

*TYPE2 디렉토리의 최종 *크기*가 *TYPE1 디렉토리보다 작은 경우가 많습니다. 일반적으로 350개 미만의 오브젝트가 있는 *TYPE2 디렉토리는 같은 수의 오브젝트가 있는 *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 많은 수의 디렉토리가 *TYPE1 버전 및 *TYPE2 버전으로 동시에 존재할 수 있어야 합니다. 이 수는 변환될 파일 시스템의 디렉토리 구조 및 iSeries 서버 구성에 따라 다릅니다.

CVTDIR 명령의 *ESTIMATE 옵션은 변환 중에 필요한 예상 보조 기억장치 양을 나타내는 정보를 제공합니다.

기호 링크 고려사항

기호 링크는 다른 오브젝트 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. 오브젝트 이름 변경에 대한 세부사항은 오브젝트 이름 변경을 참조하십시오.

독립 ASP(Auxiliary Storage Pool)

독립 ASP가 OS/400 V5R2가 설치된 시스템에 처음 연결변환되면 디렉토리가 *TYPE2로 변환됩니다. OS/400 V5R1에는 변환 계획을 세우기 용이하도록 변환 실행 시간에 대한 정보를 제공하는 계산 함수가 있습니다. 독립 ASP를 V5R2 서버에 연결변환하기 전에, 독립 ASP(이름 ASP_NAME)가 연결변환되어 활동할 때 다음 API를 V5R1 시스템에서 실행하십시오.

```
CALL QP0FCVT2(*ESTIMATE ASP_NAME *TYPE2)
```

주: 이 함수를 호출하기 전에 V5R1 시스템의 독립 ASP에서 RCLSTG를 실행하는 것이 좋습니다.

저장/복원 고려사항

*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.

*TYPE2 변환 준비

*TYPE2 디렉토리로 변환하기 전에 몇 가지 CL 명령 및 매개변수를 사용하는 것이 좋습니다.

- RCLSTG(기억장치 재생)

  파일 시스템을 변환하기 전에 RCLSTG SELECT(*ALL) 명령을 사용하면 디렉토리가 클린업되어 디렉토리 상태를 양호하게 유지할 수 있습니다. 이 명령을 사용한다고 해서 디렉토리 변환 중에 발생 가능한 모든 문제가 제거되는 것은 아니지만 이 명령을 사용하면 파일 시스템의 디렉토리를 읽을 수 있습니다.

  이 명령은 CVTDIR 명령 옵션을 사용하기 전에 한 번만 실행해야 합니다.
- 시스템 저장(SAVSYS)

  iSeries 서버의 전체 시스템 저장은 RCLSTG를 수행한 후, CVTDIR 명령의 *CONVERT 옵션을 사용하기 전에 수행해야 합니다. 시스템을 백업하려면 iSeries의 저장 메뉴를 사용하십시오. 저장 메뉴를 사용하려면 명령행에 GO SAVE를 입력하고 옵션 21을 선택하십시오. 자세한 정보는 백업 및 회복 을 참조하십시오.

  이 명령은 CVTDIR 명령 옵션을 사용하기 전에 한 번만 실행해야 합니다.
- CVTDIR 명령의 *ESTIMATE 옵션

  디렉토리를 변환하는 데 필요한 시간을 판별하려면 CVTDIR 명령의 *ESTIMATE 옵션을 사용하십시오.

*ESTIMATE 옵션을 사용하면 필요한 시간 및 보조 기억장치를 예상할 수 있을 뿐 아니라 몇 가지 장점이 더 있습니다. 이 옵션은 *TYPE1 디렉토리와 관련된 일부 2차 오브젝트를 빌드하므로, 변환 시 이 오브젝트를 작성할 필요가 없어지기 때문에 변환 실행 속도를 높일 수 있습니다. 2차 오브젝트는 파일 시스템을 *TYPE2 형식으로 변환하기 위해 *CONVERT 옵션을 사용할 때까지 존재합니다. 또한 *ESTIMATE 옵션은 파일 시스템의 모든 디렉토리를 읽어 디렉토리를 내재적으로 확인합니다. *ESTIMATE 옵션을 사용해도 실제 변환 중에 발생 가능한 모든 오류를 찾을 수 있는 것은 아니지만 매우 유용합니다.

*ESTIMATE 옵션을 실행한 후 작업 기록부에서 오류를 검사하고, 파일 시스템을 변환하기 전에 필요한 회복 조치를 수행하십시오. 해당 회복 조치를 수행한 후 다른 문제점이 없는지 확인하려면 다시 계산을 실행하는 것이 좋습니다(필수는 아님).

- 보조 기억장치 고려사항

  변환될 파일 시스템을 포함한 ASP에 대해 사용 가능한 보조 기억장치를 확인하십시오. CVTDIR *ESTIMATE 옵션은 CPIA090 메세지를 표시하여 ASP에 사용 가능한 보조 기억장치와 변환 중에 필요한 예상 보조 기억장치 양을 보여줍니다. 또한 CPIA091 메세지를 표시하여 변환 후 파일 시스템의 전체 *TYPE2 디렉토리 *크기*가 기존 *TYPE1 디렉토리의 *크기*보다 클지 또는 작을지를 예측합니다. ASP의 사용 가능한 기억장치 *크기*는 사용되지 않은 보조 기억장치(메세지 CPIA090에 표시)에 *TYPE1 디렉토리 *크기*와 *TYPE2 디렉토리 *크기*의 차이(메세지 CPIA091에 표시)를 더한 합계이어야 합니다.

  STRSST(시스템 서비스 툴 시작) 명령을 사용하고 디스크 장치에 대한 작업 옵션을 선택해도 사용 가능한 보조 기억장치를 알 수 있습니다.

  주: 시스템에 하나의 ASP만 정의되어 있는 경우 WRKSYSSTS(시스템 상태에 대한 작업) 명령을 사용해도 사용 가능한 보조 기억장치 정보를 표시할 수 있습니다.

  CVTDIR 명령에서 옵션을 사용하기 전에 일반적인 시스템 클린업을 수행하는 것이 좋습니다. 더 이상 필요하지 않은 디렉토리나 파일이 있는 경우 CVTDIR 명령의 옵션을 사용하기 전에 해당 디렉토리나 파일을 제거하십시오. 이렇게 하면 보조 기억장치 공간이 비워지므로 사용 가능한 보조 기억장치 공간을 더 정확하게 계산할 수 있으며, 처리할 오브젝트 수가 줄어들어 변환을 완료하는 데 걸리는 시간을 단축할 수 있습니다.

- 작업 메세지 대기행렬 가득 참 조치를 CVTDIR 명령을 발행하는 작업에 대한 *PRTWRAP로 변경해 보십시오. 이를 수행하면 결과는 다음과 같습니다.
  1. 작업 기록부가 꽉 찰 경우 작업이 비정상적으로 종료되지 않습니다.
  2. 작업 기록부가 랩핑될 경우 오버레이된 메세지를 스풀 파일에 인쇄하여 중요한 메세지가 손실될 위험이 없습니다.
- 독립 ASP가 있는 시스템의 경우, OS/400 V5R2를 실행 중인 시스템에 독립 ASP를 연결변환하기 전에 모든 독립 ASP에 V5R1의 *ESTIMATE 함수를 사용하십시오. 이렇게 하면 설치 후 독립 ASP를 처음 연결변환하는 데 걸리는 예상 시간을 알 수 있습니다. 자세한 정보는 독립 ASP(Auxiliary Storage Pool)를 참조하십시오.

변환 처리

CVTDIR 명령은 *TYPE1 디렉토리를 *TYPE2 디렉토리로 변환합니다. 변환 프로세스 동안 다음과 같은 몇 가지 사항을 고려해야 합니다.

- "루트"(/) 또는 QOpenSys 변환
- 사용자 정의 파일 시스템 변환
- 사용자 프로파일 작성
- 오브젝트 이름 변경
- 사용자 프로파일 고려사항

"루트" 또는 QOpenSys 변환

"루트" 또는 QOpenSys 파일 시스템을 변환할 때 시스템은 제한 상태에 **있어야 합니다**. 변환 중에는 어떤 파일 시스템도 사용할 수 없습니다. 모든 UDFS 및 NFS 파일 시스템은 CVTDIR 명령에 의해 마운트 해제되며 변환이 완료된 후 다시 마운트되지는 않습니다. 마운트 명령(MOUNT)을 사용하여 UDFS 또는 NFS 파일 시스템을 다시 마운트할 수 있습니다.

사용자 정의 파일 시스템 변환

ASP 1-32의 UDFS가 변환될 때 시스템 사용자는 이 파일 시스템을 사용할 수 없습니다. 이러한 각 ASP에 대해 /dev 디렉토리에 QASPxx 디렉토리가 있습니다. CVTDIR 명령이 실행되는 동안 사용자가 ASP의 UDFS를 액세스할 수 없도록 이름공간에서 QASPxx 디렉토리가 제거됩니다. CVTDIR 명령이 QASPxx 디렉토리를 포함하여 모든 오브젝트에 대한 처리를 완료하면 오브젝트가 이름공간에 다시 놓여지며 시스템 사용자가 사용할 수 있게 됩니다. ASP의 UDFS는 CVTDIR 명령에 의해 마운트 해제되며 변환이 완료된 후 다시 마운트되지는 않습니다. 마운트 명령(MOUNT)을 사용하여 UDFS를 다시 마운트할 수 있습니다.

주: QASP01 디렉토리는 모든 시스템에 존재합니다.

사용자 프로파일 작성

변환 기능은 변환 기능이 실행될 때 사용될 사용자 프로파일을 작성합니다. 이 사용자 프로파일의 이름은 QP0FCVxxxx입니다. 여기서 xxxx는 0001과 같은 숫자입니다. 이 번호는 원래 소유자가 자신의 디렉토리를 소유할 수 없을 경우 변환되는 파일 시스템의 디렉토리를 소유하기 위해 변환 기능에서 사용합니다.

이 사용자 프로파일은 변환이 완료된 후 가능하면 삭제됩니다. 디렉토리 소유권이 이 사용자 프로파일 중 하나에 주어지는 경우 메세지 CPIA08B가 송신됩니다.

오브젝트 이름 변경

*TYPE2 디렉토리의 링크명은 유효한 UTF-16 이름이어야 합니다. 이 이름은 UCS2 레벨 1 이름을 갖는 *TYPE1 디렉토리와는 다릅니다. 이러한 이유로 디렉토리 변환 시에 유효하지 않거나 중복되는 이름이 발견될 수 있습니다. 이름이 유효하지 않거나 중복되는 경우, 해당 이름이 고유하고 유효한 UTF-16 이름으로 변경되며 메세지 CPIA08A가 작업 기록부로 송신되어 원래 이름과 새 이름을 나열합니다. 이름에 포함된 결합 문자나 유효하지 않은 대체 문자쌍으로 인해 오브젝트명이 변경될 수 있습니다.

UTF-16에 대한 자세한 정보는 유니코드(Unicode) 홈페이지(http://www.unicode.org )를 참조하십시오.

결합 문자: 일부 문자는 둘 이상의 유니코드(Unicode) 문자로 구성될 수 있습니다. 예를 들어 액센트나 움라우트가 있는 문자가 있습니다. 이들 문자는 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 형식입니다. *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.

대체 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 변환이 진행되는 동안 많은 수의 디렉토리가 동시에 두 형식으로 잠시 존재하며 동일한 사용자 프로파일에 의해 소유됩니다. 사용자 프로파일에 대한 최대 기억장치 제한 때문에 *TYPE2 디렉토리를 더 이상 작성할 수 없는 경우에는 사용자 프로파일에 대한 제한값이 증가합니다. 메세지 CPIA08C가 작업 기록부로 송신되고 변환이 계속됩니다.

디렉토리 소유자 변경: *TYPE1 디렉토리를 소유한 사용자 프로파일이 작성된 *TYPE2 디렉토리를 소유할 수 없는 경우 *TYPE2 디렉토리의 소유자가 사용자 프로파일 작성에서 설명된 다른 대체 사용자 프로파일 중 하나로 설정됩니다. 메세지 CPIA08B가 작업 기록부로 송신되고 변환이 계속됩니다.

예: 모든 파일 시스템(적은 수의 오브젝트) 변환

시스템 A는 다섯 개의 ASP(Auxiliary Storage Pool)로 구성되어 있습니다. 각각은 1(시스템 ASP), 3, 5, 11 및 25입니다. 시스템의 어떤 파일 시스템도 *TYPE1 디렉토리에서 *TYPE2 디렉토리로 변환되지 않았습니다. 모든 파일 시스템을 변환하려고 합니다. 파일 시스템에 포함된 오브젝트의 수가 많지 않기 때문에 하루 동안 모든 단계를 수행합니다.

적은 수의 오브젝트를 가진 모든 파일 시스템의 디렉토리를 변환하려면 다음을 수행하십시오.

1. 시스템을 제한 상태로 설정하십시오.

2. 명령행에 RCLSTG SELECT(*ALL)를 입력하십시오.
3. 저장 메뉴를 사용하여 시스템을 저장하십시오. 명령행에 GO SAVE를 입력하고 옵션 21을 선택하십시오.
4. 명령행에 CVTDIR OPTION(*ESTIMATE) FILESYS(*ALL) FORMAT(*TYPE2)를 입력하십시오.
5. *ESTIMATE 함수에 오류 메시지가 있는지 검사하십시오.
6. 모든 ASP에 사용 가능한 보조 기억장치 공간이 충분한지 확인하십시오.
7. 명령행에 CVTDIR OPTION(*CONVERT) FILESYS(*ALL) FORMAT(*TYPE2)를 입력하십시오.

   주: 모든 파일 시스템(*ALL)을 변환할 때 나열된 파일 시스템을 변환할지 확인하는 CPAA084 메세지가 표시됩니다.
8. *CONVERT 함수에 오류 메세지가 있는지 검사하십시오.
9. 시스템을 제한 상태에서 해제하십시오.

예: 모든 파일 시스템(많은 수의 오브젝트) 변환

시스템 B는 다섯 개의 ASP(Auxiliary Storage Pool)로 구성되어 있습니다. 각각은 1(시스템 ASP), 3, 5, 11 및 25입니다. 시스템의 어떤 파일 시스템도 *TYPE1 디렉토리에서 *TYPE2 디렉토리로 변환되지 않았습니다. 모든 파일 시스템을 변환하려고 합니다. 파일 시스템에 많은 수의 오브젝트가 포함되어 있어 두 번의 주말 동안 변환 단계를 수행합니다.

첫 주말:

1. 시스템을 제한 상태로 설정하십시오.
2. 명령행에 RCLSTG SELECT(*ALL)를 입력하십시오.
3. 저장 메뉴를 사용하여 시스템을 저장하십시오. 명령행에 GO SAVE를 입력하고 옵션 21을 선택하십시오.
4. 시스템을 제한 상태에서 해제하십시오.

주 중:

5. 명령행에 CVTDIR OPTION(*ESTIMATE) FILESYS(*ALL) FORMAT(*TYPE2)를 입력하십시오.
6. *ESTIMATE 함수에 오류 메시지가 있는지 검사하십시오.
7. 모든 ASP에 사용 가능한 보조 기억장치 공간이 충분한지 확인하십시오.

두 번째 주말:

8. 시스템을 제한 상태로 설정하십시오.
9. 명령행에 CVTDIR OPTION(*CONVERT) FILESYS(*ALL) FORMAT(*TYPE2)를 입력하십시오.

   주: 모든 파일 시스템(*ALL)을 변환할 때 나열된 파일 시스템을 변환할지 확인하는 CPAA084 메세지가 표시됩니다.
10. *CONVERT 함수에 오류 메세지가 있는지 검사하십시오.
11. 시스템을 제한 상태에서 해제하십시오.

예: 특정 ASP만 변환

시스템 C는 여섯 개의 ASP(Auxiliary Storage Pool)로 구성되어 있습니다. 각각은 1(시스템 ASP), 2, 4, 8, 10 및 30입니다. 시스템의 어떤 파일 시스템도 변환되지 않았습니다. ASP 4, 10 및 30의 UDFS만 변환하려고 합니다.

특정 ASP의 UDFS에 있는 디렉토리를 변환하려면 다음을 수행하십시오.

1. 파일 시스템의 디렉토리 형식을 확인하십시오. 명령행에 CVTDIR OPTION(*CHECK)을 입력하여 다음을 수행하십시오.
2. 시스템을 제한 상태로 설정하십시오.
3. 명령행에 RCLSTG SELECT(*ALL)를 입력하십시오.
4. 저장 메뉴를 사용하여 시스템을 저장하십시오. 명령행에 GO SAVE를 입력하고 옵션 21을 선택하십시오.
5. 시스템을 제한 상태에서 해제하십시오.
6. 명령행에 CVTDIR OPTION(*ESTIMATE) FILESYS(*UDFS) ASP(4) FORMAT(*TYPE2)를 입력하십시오.
7. *ESTIMATE 함수에 오류 메시지가 있는지 검사하십시오.
8. 명령행에 CVTDIR OPTION(*ESTIMATE) FILESYS(*UDFS) ASP(10) FORMAT(*TYPE2)를 입력하십시오.
9. *ESTIMATE 함수에 오류 메시지가 있는지 검사하십시오.
10. 명령행에 CVTDIR OPTION(*ESTIMATE) FILESYS(*UDFS) ASP(30) FORMAT(*TYPE2)를 입력하십시오.
11. *ESTIMATE 함수에 오류 메시지가 있는지 검사하십시오.
12. 모든 ASP에 사용 가능한 보조 기억장치 공간이 충분한지 확인하십시오.
13. 명령행에 CVTDIR OPTION(*CONVERT) FILESYS(*UDFS) ASP(4) FORMAT(*TYPE2)를 입력하십시오.
14. *CONVERT 함수에 오류 메세지가 있는지 검사하십시오.
15. 명령행에 CVTDIR OPTION(*CONVERT) FILESYS(*UDFS) ASP(10) FORMAT(*TYPE2)를 입력하십시오.
16. *CONVERT 함수에 오류 메세지가 있는지 검사하십시오.
17. 명령행에 CVTDIR OPTION(*CONVERT) FILESYS(*UDFS) ASP(30) FORMAT(*TYPE2)를 입력하십시오.
18. *CONVERT 함수에 오류 메세지가 있는지 검사하십시오.

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 시스템 명령에서 슬래시(/) 대신 역슬래시(\)를 사용해도 됩니다.

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.

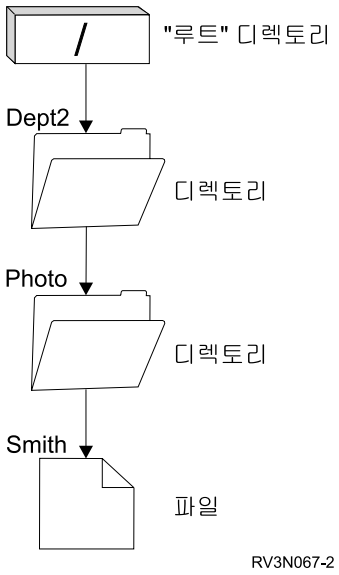


그림 6. 경로명의 구성요소

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.

링크

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

디렉토리 기반 파일 시스템을 사용하는 사용자의 경우, 오브젝트(예: 파일)를 서버에 이를 식별하는 이름을 가진 것으로 생각십시오. 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.

일단 오브젝트라기 보다는 링크에 이름이 부여된다는 생각에 익숙해지면, 전에는 보지 못했던 가능성들을 볼 수 있게 됩니다. 같은 오브젝트에 여러 링크가 있을 수 있습니다. 예를 들어, 두 사용자가 각 사용자의 홈 디렉토리부터 파일로 링크를 가짐으로써 파일을 공유할 수 있습니다(8 페이지의 『현재 디렉토리 및 홈 디렉토리』 참조). 특정 링크 유형은 파일 시스템 전체에 사용될 수 있고, 오브젝트가 없어도 존재할 수 있습니다.

링크에는 **하드 링크**와 **기호 링크**, 두 가지 유형이 있습니다

링크에 대한 자세한 정보는 다음 주제를 참조하십시오.

- 하드 링크
- 기호 링크
- 비교: 하드 링크와 기호 링크

하드 링크

링크 또는 **하드 링크**는 실제 오브젝트로 연결되지 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.

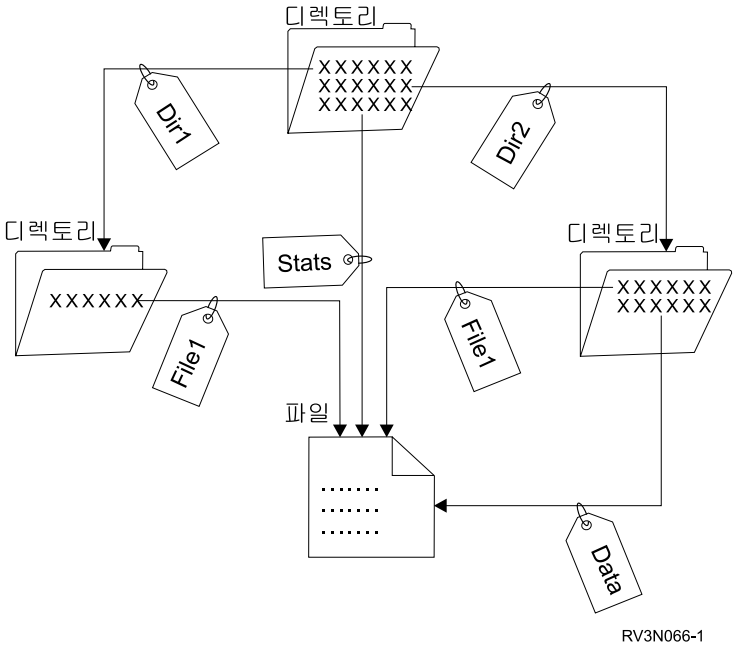


그림 7. 디렉토리 항목은 각 하드 링크를 정의합니다.

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 서버에서 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 시스템에도 적용될 수 있으나, 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 QDLS에서는 *동일한 오브젝트*에 대해 여러 개의 하드 링크가 허용되지 않습니다.

| 하드 링크는 파일 시스템간에 사용될 수 없습니다. 예를 들어, QOpenSys 파일 시스템에 있는 디렉토리는 QSYS.LIB, 독립 ASP QSYS.LIB 파일 시스템 또는 QDLS 파일 시스템의 문서로의 하드 링크를 가질 수 없습니다.

기호 링크

기호 링크는 **소프트 링크**라고도 불리며 파일에 들어 있는 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.

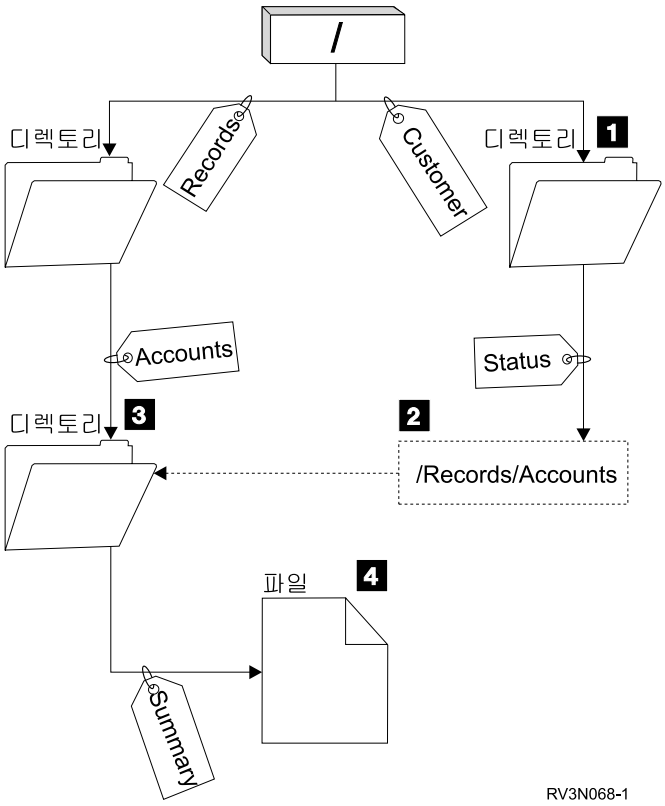


그림 8. 기호 링크 사용 예

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

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

시스템은 디렉토리 **1** 로 가는 *Customer* 링크를 따라가고, 그런 다음 *Status* 링크 뒤를 따라갑니다. 경로명 **2** 를 가지고 있는 *상태* 링크는 기호 링크입니다. 경로명이 /로 시작되기 때문에 시스템은 /(『루트』) 디렉토리로 리턴하여 차례로 *Records*와 *Accounts* 링크를 따라갑니다. 이 경로는 또 다른 디렉토리 **3** 으로 갑니다. 이제 시스템은 프로그램이 제공하는 경로명의 경로를 완료합니다. 사용자가 필요로 하는 자료가 들어 있는 파일 **4** 로 가는 *요약* 링크 다음에 옵니다.

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 QOpenSys 파일 시스템 또는 "루트"(/) 파일 시스템에 기호 링크를 작성할 수 있습니다.

- QSYS.LIB 또는 독립 ASP QSYS.LIB 파일 시스템의 데이터베이스 파일 멤버 액세스
- QDLS 파일 시스템의 문서 액세스

『비교: 하드 링크와 기호 링크』도 참조하십시오.

비교: 하드 링크와 기호 링크

프로그램 내에서 경로명을 사용할 때, 하드 링크나 기호 링크를 선택할 수 있습니다(19 페이지의 『링크』 참조). 링크 유형은 각각 장단점을 가지고 있습니다. 다음은 링크의 한 유형이 다른 유형에 비해 가지는 장점을 표시한 것입니다.

표 1. 하드 링크 및 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.
동적 오브젝트(속성이 변경됨)	보다 느림. 오브젝트의 많은 속성이 각 하드 링크에 저장됩니다. 따라서, 동적 오브젝트로의 변경 속도는 오브젝트로의 하드 링크 수가 증가함에 따라 느려집니다.	보다 빠름. 동적 오브젝트로의 변경은 기호 링크에 의해 영향받지 않습니다.
정적 오브젝트(속성이 변경되지 않음)	보다 빠름. 정적 오브젝트인 경우, 이름 해상도가 1차적인 성능 요건입니다. 이름 해상도는 하드 링크 사용시 보다 빨라집니다.	보다 느림. 이름 해상도는 기호 링크 사용시 보다 느려집니다.
범위	제한적. 하드 링크는 파일 시스템간에 사용될 수 없습니다.	무제한적. 기호 링크는 파일 시스템간에 사용될 수 있습니다.

확장 속성

확장 속성(Extended Attribute)은 오브젝트에 대한 추가 상세 항목을 제공하는 오브젝트와 연관된 정보입니다. EA는 이름 및 값으로 구성됩니다. 값은 텍스트, 2진 자료 또는 다른 유형의 자료가 될 수 있습니다.

오브젝트에 대한 EA는 오브젝트가 있는 한 존재합니다.

EA는 다양한 방식으로 이루어지며 다양한 정보를 포함할 수 있도록 사용될 수 있습니다. 특히, 다음 세 가지 확장 속성(EA)에 대해 알아야 합니다.

.SUBJECT

오브젝트의 내용 또는 목적에 대한 간단한 설명

.TYPE

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

.CODEPAGE

오브젝트에 대해 사용될 코드 페이지. 또한, 오브젝트에 대해 사용된 코드 페이지는 해당 오브젝트와 연관된 EA에 대해서도 사용됩니다.

이름의 첫 번째 문자로서의 마침표(.)는 EA가 시스템 사용에 대해 예약된 표준 시스템 EA(SEA)라는 의미입니다.

다양한 파일 시스템의 다양한 오브젝트에 EA가 있을 수도, 없을 수도 있습니다. QSYS.LIB 및 독립 ASP QSYS.LIB 파일 시스템은 세 가지 사전 정의된 EA 즉, .SUBJECT, .TYPE 및 .CODEPAGE를 지원합니다. 문서 라이브러리 서비스(QDLS) 파일 시스템에서, 폴더 및 문서에는 모든 종류의 EA가 있을 수 있습니다. 어떤 폴더 및 문서에는 EA가 있고, 어떤 문서 및 폴더에는 없습니다. 『루트』(/), 개방 시스템(QOpenSys), 사용자 정의 파일 시스템, 모든 디렉토리, 스트림 파일, 기호 링크는 모든 종류의 EA를 가지고 있습니다. 그러나 어떤 것은 EA를 전혀 가질 수 없습니다.

WRKLNK(오브젝트 링크에 대한 작업) 명령을 사용하여 오브젝트에 대한 .SUBJECT 확장 속성(EA)을 표시할 수 있습니다. 어플리케이션이나 사용자가 EA에 액세스하고 변경할 수 있는 다른 통합 파일 시스템 지원은 없습니다. 이 규칙에 대한 유일한 예외는 DSPUDFS 표시와 마운트된 DSPMFSINF(파일 시스템 정보 표시) CL 명령으로, 이것은 사용자에게 확장 속성을 표시해줍니다.

그러나 QDLS에 있는 일부 오브젝트와 연관된 EA는 계층 파일 시스템(HFS)이 제공하는 인터페이스를 통해 변경될 수 있습니다. 이와 같은 파일 시스템에 대한 자세한 정보는 82 페이지의 『문서 라이브러리 서비스 파일 시스템(QDLS)』 및 85 페이지의 『광 파일 시스템(QOPT)』에서 볼 수 있습니다.

클라이언트 PC가 OS/2 또는 Windows를 통해 iSeries 서버에 연결된 경우, 각 오퍼레이팅 시스템의 프로그래밍 인터페이스(예: DosQueryFileInfo 및 DosSetFileInfo)를 사용하여 파일 오브젝트의 EA를 조회하고 설정할 수 있습니다. 또한 OS/2 사용자는 설정 노트북을 사용하여, 다시 말하면, 오브젝트와 연관된 팝업 메뉴에서 설정을 선택해서 데스크탑에 있는 오브젝트의 EA를 변경할 수도 있습니다.

확장 속성을 정의하는 경우, 다음 명명 규칙을 사용하십시오.

- EA명은 최대 255자가 될 수 있습니다.
- 이름의 첫 번째 문자로서 마침표(.)를 사용하지 마십시오. 마침표로 시작하는 EA는 표준 시스템 EA로 해석됩니다.
- 이름이 중복되는 것을 피하기 위해서는 일관된 EA 명명 구조를 사용하십시오. 다음 형식을 사용하는 것이 좋습니다.

```
CompanyNameProductName.Attribute_Name
```

이름 연속성

『루트』(/), QOpenSys 및 사용자 정의 파일 시스템을 사용할 때, 오브젝트명의 문자가 동일하게 남도록 하는 시스템 지원을 이용할 수 있습니다. 이는 iSeries 서버간에 이 파일 시스템을 사용할 때와 다른 문자 코드화 체계(코드 페이지)를 갖는 연결된 장치 간에 이 파일 시스템을 사용할 때도 적용됩니다. 서버는 *TYPE1 디렉토리에 대해서는 UCS2 레벨 1(**유니코드(Unicode)**라고도 함), *TYPE2 디렉토리에 대해서는 UTF-16이라는 16비트 형식 이름으로 문자를 저장합니다. 디렉토리 형식에 대한 자세한 정보는 *TYPE2 디렉토리를 참조하십시오. UCS2 레벨 1 및 UTF-16은 ISO 10646 표준의 한 부분입니다. 이름 사용 시 시스템은 저장된 문자

형식을 사용중인 코드 페이지의 적절한 문자 표시로 변환합니다. 또한 각 오브젝트와 연관된 확장 속성명도 동일한 방식으로 처리됩니다.

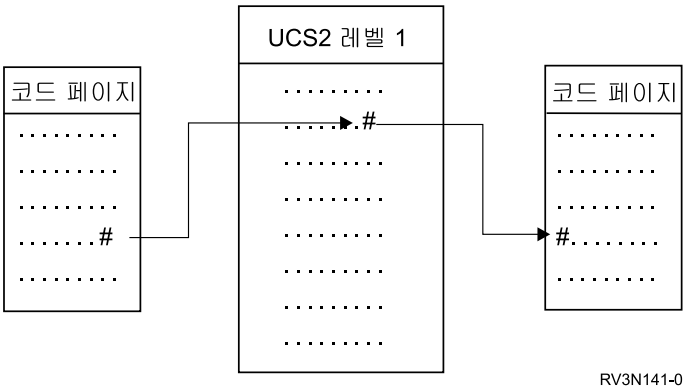


그림 9. 인코딩 체계간 문자의 동일성

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

제 3 장 기존 시스템 인터페이스를 사용하여 통합 파일 시스템 액세스

시스템의 라이브러리, 오브젝트, 데이터베이스 파일, 폴더 및 문서에 대한 작업에 사용되는 메뉴, 명령, 표시장치와 같은 모든 사용자 인터페이스는 통합 파일 시스템이 도입되기 전과 마찬가지로 작동됩니다. 그러나 이 인터페이스는 통합 파일 시스템에 의해 지원되는 스트림 파일, 디렉토리 및 기타 오브젝트에 대한 작업에 사용될 수 없습니다.

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

메뉴와 표시장치 또는 제어 언어(CL) 명령을 사용하여 서버에서 통합 파일 시스템의 디렉토리 및 오브젝트와 대화할 수 있습니다. 또한 어플리케이션 프로그램 인터페이스(API)를 사용하여 통합 파일 시스템의 스트림 파일, 디렉토리 및 기타 지원을 활용할 수 있습니다.

Windows 데스크탑에서 서버를 관리하는 데 사용되는 그래픽 사용자 인터페이스, iSeries Navigator를 통해 통합 파일 시스템과 대화할 수도 있습니다.

통합 파일 시스템과 대화하는 몇 가지 방법이 있습니다.

API 사용
통합 파일 시스템 디렉토리와 스트림 파일상에서 조작을 수행하는 API(Application Program Interfaces)는 C 언어 함수 형식입니다.

CL 명령 사용
CL 명령은 통합 파일 시스템를 통해 액세스할 수 있는 파일 시스템의 파일 및 기타 오브젝트에서 조작할 수 있습니다.

iSeries 메뉴 및 표시장치 사용
서버가 제공하는 메뉴와 표시장치 세트를 사용하여 통합 파일 시스템의 파일 및 기타 오브젝트에 대한 조작을 수행할 수 있습니다.

iSeries Navigator 사용
iSeries Navigator는 Windows 데스크탑에서 서버를 관리하기 위한 그래픽 사용자 인터페이스입니다.

PC 사용
PC가 iSeries 서버에 연결되어 있는 경우, PC에 저장되어 있는 것처럼 통합 파일 시스템의 디렉토리 및 오브젝트와 대화할 수 있습니다.

iSeries 메뉴 및 표시장치를 사용하여 조작 수행

서버가 제공하는 메뉴와 표시장치 세트를 사용하여 통합 파일 시스템의 파일 및 기타 오브젝트에 대한 조작을 수행할 수 있습니다. 통합 파일 시스템 메뉴를 표시하려면 다음을 수행하십시오.

1. 서버에 사인 온하십시오.
2. 계속하려면 **Enter**를 누르십시오.
3. iSeries 기본 메뉴에서 **파일, 라이브러리 및 폴더** 옵션을 선택하십시오.
4. 파일, 라이브러리 및 폴더 메뉴에서 **통합 파일 시스템** 옵션을 선택하십시오.

필요에 따라 통합 파일 시스템에서 디렉토리 명령, 오브젝트 명령 또는 보안 명령에 대해 작업할 수 있습니다. 그러나 사용하게 될 CL 명령을 알고 있으면, 옵션 메뉴를 바이패스하고 화면 맨 아래 명령행에 CL 명령을 입력하고 **Enter**를 누르면 됩니다.

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

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

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

통합 파일 시스템 메뉴로부터 다음 조작을 수행할 수 있는 화면을 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
- 스트림 파일 및 데이터베이스 파일 멤버간 자료 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. 특정 파일 시스템에 대한 제한사항은 4 페이지의 『통합 파일 시스템의 파일 시스템』을 참조하십시오.

통합 파일 시스템 메뉴 및 표시장치에 대한 자세한 정보는 다음 주제를 참조하십시오.

- CL 명령 및 표시장치에 대한 경로명 규칙

CL 명령을 사용한 조작 수행

통합 파일 시스템 메뉴 및 화면을 통해 할 수 있는 모든 조작(27 페이지의 『iSeries 메뉴 및 표시장치를 사용하여 조작 수행』 참조)은 제어 언어(CL) 명령을 입력하여 수행될 수 있습니다. 이들 명령은 통합 파일 시스템 인터페이스를 통해 액세스할 수 있는 파일 시스템의 파일 및 기타 오브젝트에서 조작할 수 있습니다.

표1은 통합 파일 시스템 명령을 요약한 것입니다. 특히 사용자 정의 파일 시스템, 네트워크 파일 시스템 및 일반적으로 마운트된 파일 시스템과 관련되는 CL 명령에 대한 자세한 내용은 72 페이지의 『사용자 정의 파일 시스템(UDFS)』 및 97 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 제

공됩니다.

표 2. 통합 파일 시스템 명령

명령	설명	별명
ADDLNK	링크 추가. 디렉토리와 오브젝트간의 링크를 추가합니다.	
ADDMFS	마운트된 파일 시스템 추가. 로컬 클라이언트 디렉토리에 내보낸, 리모트 서버 파일 시스템을 위치시킵니다.	MOUNT
APYJRNCHG[2]	저널링된 변경사항 적용. 저널 항목을 사용하여 저널링된 오브젝트가 저장된 이후로 발생한 변경사항을 적용하거나 지정된 시점까지의 변경사항을 적용합니다.	
CHGATR	속성 변경. 단일 오브젝트, 오브젝트 그룹 또는 디렉토리, 디렉토리 목차, 모든 서브디렉토리의 목차에 변경된 속성이 있는 디렉토리의 속성을 변경합니다.	
CHGAUD	감사 값 변경. 오브젝트에 대한 감사를 시작하거나 중지합니다.	
CHGAUT	권한 변경. 사용자 또는 사용자 그룹에게 오브젝트에 대한 특정 권한을 부여합니다.	
CHGCURDIR	현재 디렉토리 변경. 현재 디렉토리로 사용할 디렉토리를 변경합니다.	CD, CHDIR
CHGNFSEXP	네트워크 파일 시스템 내보내기 변경. NFS 클라이언트로 내보낸된 내보내기 표로 디렉토리 트리를 추가하거나 제거합니다.	EXPORTFS
CHGOWN	소유자 변경. 다른 사용자에게로 오브젝트 소유권을 이전합니다.	
CHGPGP	1차 그룹 변경. 다른 사용자로 1차 그룹을 변경합니다.	
CHKIN	체크 인. 이전에 체크 아웃된 오브젝트를 체크 인합니다.	
CHKOUT	체크 아웃. 다른 사용자가 변경하지 못하도록 오브젝트를 체크 아웃합니다.	
CPY	복사. 단일 오브젝트 또는 오브젝트 그룹을 복사합니다.	COPY
CPYFRMSTMF	스트림 파일로부터 복사. 스트림 파일로부터 데이터베이스 파일 멤버로 자료를 복사합니다.	
CPYTOSTMF	스트림 파일로 복사. 데이터베이스 파일 멤버로부터 스트림 파일로 자료를 복사합니다.	
CRTDIR	디렉토리 작성. 새로운 디렉토리를 시스템에 추가합니다.	MD, MKDIR
CRTUDFS	UDFS 작성. 사용자 정의 파일 시스템을 작성합니다.	
CVTDIR	디렉토리 변환. 통합 파일 시스템 디렉토리를 *TYPE1 형식에서 *TYPE2 형식으로 변환하는 데 필요한 정보를 제공하거나 변환을 수행합니다.	
CVTRPCSRC	RPC 소스 변환. 리모트 프로시듀어 호출(RPC) 언어로 작성된 입력 파일로부터 C 코드를 생성합니다.	RPCGEN
DLTUDFS	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.	
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.	

표 2. 통합 파일 시스템 명령 (계속)

명령	설명	별명
EDTF	스트림 파일 편집. 스트림 파일 또는 데이터베이스 파일을 편집합니다.	
ENDJRN[2]	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	이동. 오브젝트를 다른 디렉토리로 이동합니다.	MOVE
RLSIFSLCK	통합 파일 시스템 잠금 해제. 클라이언트에 의해 보류되거나 오브젝트에 있는 모든 NFS 바이트 범위 잠금을 해제합니다.	
RMVDIR	디렉토리 제거. 시스템에서 디렉토리를 제거합니다.	RD, RMDIR
RMVLNK	링크 제거. 오브젝트에 대한 링크를 제거합니다.	DEL, ERASE
RMVMFS	마운트된 파일 시스템 제거. 내보낸, 리모트 서버 파일 시스템을 로컬 클라이언트 디렉토리에서 제거합니다.	UNMOUNT
RNM	이름 변경. 디렉토리에서 오브젝트명을 변경합니다.	REN
RPCBIND	RPC 바인더 디몬 시작. 리모트 프로시듀어 호출(RPC) RPCBind 디몬을 시작합니다.	
RST	복원. 오브젝트나 오브젝트 그룹을 백업 장치에서 시스템으로 복사합니다.	
RTVCURDIR	현재 디렉토리 검색. 현재 디렉토리명을 검색하고 이를 지정된 변수(CL 프로그램에서 사용됨)에 넣습니다.	
SAV	저장. 오브젝트나 오브젝트 그룹을 시스템에서 백업 장치로 복사합니다.	
SNDJRNE[2]	저널 항목 송신. 선택적으로 저널링된 오브젝트와 연관된 사용자 저널 항목을 저널 리시버에 추가합니다.	
STRJRN[2]	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 디몬을 시작합니다.	
WRKAUT	권한에 대한 작업. 사용자 리스트 및 사용자 권한을 표시하고, 사용자 추가, 사용자 권한 변경 또는 사용자 제거 등의 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 작동되는 것은 아닙니다.
2. 자세한 정보는 iSeries Information Center의 저널 관리를 참조하십시오.

| 통합 파일 시스템 CL 명령 및 특정 파일 시스템에서 이 명령을 사용할 때의 제한사항에 대한 자세한 정보는
| 다음 주제를 참조하십시오.

| • 통합 파일 시스템의 파일 시스템

| • CL 명령 및 표시장치에 대한 경로명 규칙

| • iSeries Information Center의 CL 주제

CL 명령 및 표시장치에 대한 경로명 규칙

통합 파일 시스템 명령이나 화면을 사용하여 오브젝트에 대해 조작할 때, 경로명을 제공함으로써 오브젝트를 식별합니다. 다음은 경로명 지정시 유의해야 하는 규칙을 요약한 것입니다. 이 규칙에서 **오브젝트**는 디렉토리, 파일, 링크 또는 기타 오브젝트를 말합니다.

- 오브젝트명은 각 디렉토리 내에서 고유해야 합니다.
- 통합 파일 시스템 CL 명령에 전달된 경로명은 작업에 대해 현재 유효한 CCSID로 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.
- 경로명은 명령행에 입력시 어포스트로피(')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 명령에 의해 조작될 오브젝트명으로 끝납니다. 경로의 각 구성요소명은 슬래시(/)나 역슬래시(\)로 구분됩니다. 그 예는 다음과 같습니다.

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

  또는

  ```
  'Dir1\Dir2\Dir3\UsrFile'
  ```

- / 및 \ 문자와 널(null) 문자는 경로명의 개별 구성요소에 사용될 수 없습니다(/와 \가 분리문자로 사용되기 때문입니다). 명령어에 의해 소문자가 대문자로 변경되지 않습니다. 오브젝트가 들어 있는 파일 시스템이 대소문자를 구분하는지와 오브젝트가 작성 또는 검색중인지에 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.

  각 파일 시스템의 경로명 한계에 대해서는 통합 파일 시스템의 파일 시스템을 참조하십시오.
- 경로명 시작 부분의 / 또는 \ 문자는 경로가 최상위 디렉토리 즉, /(『루트』) 디렉토리에서 시작함을 의미합니다. 그 예는 다음과 같습니다.

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

- 경로명이 / 또는 \ 문자로 시작하지 않는 경우, 경로는 명령을 입력하는 사용자의 현재 디렉토리에서 시작한다고 간주됩니다. 그 예는 다음과 같습니다.

  ```
  'MyDir/MyFile'
  ```

  여기서 MyDir은 사용자의 현재 디렉토리의 서브디렉토리입니다.

- 경로명 맨 처음에서 슬래시(또는 역슬래시)가 따라 나오는 틸드(~) 문자는 명령을 입력하는 사용자의 홈 디렉토리에서 경로가 시작함을 의미합니다. 예를 들면, 다음과 같습니다.

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

- 경로명 맨 처음에 사용자명과 슬래시(또는 역슬래시)가 따라 나오는 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'
  ```

- iSeries 서버 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 대한 조작시, 구성요소명은 *name.object-type*의 형식이어야 하며, 예는 다음과 같습니다.

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

  보다 자세한 내용은 76 페이지의 『라이브러리 파일 시스템(QSYS.LIB)』을 참조하십시오.

- 독립 ASP QSYS.LIB 파일 시스템의 오브젝트에 대한 조작시, 구성요소명은 *name.object-type*의 형식이어야 하며, 예는 다음과 같습니다.

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

  보다 자세한 내용은 79 페이지의 『독립 ASP QSYS.LIB』을 참조하십시오

- 다음 문자들이 구성요소명에 사용되는 경우, 경로명은 작은 따옴표(') 또는 인용 부호(")안에 넣어야 합니다.
  - 별표(*)
  - 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 틀리게 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 사용되지 않는 경우, 다음과 같이 16진 표시로 입력해야 합니다.

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

  따라서, 경로명에서 16진 40이하의 코드점을 사용하지 않는 것이 좋습니다. 이 제한사항은 API가 아닌 명령어 및 연관된 화면에 대해서만 적용됩니다(54 페이지의 『API를 사용한 조작 수행』 참조).

특정 명령 사용에 대한 제한사항은 명령 도움말이나 iSeries Information Center의 CL(Control language) 주제를 참조하십시오.

PC를 사용한 조작 수행

PC가 iSeries 서버에 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, 서버 드라이브의 오브젝트를 선택하여 PC 드라이브로 해당 오브젝트를 끌면, 실제로 오브젝트를 서버에서 PC로 복사할 수 있습니다.

Windows 인터페이스를 사용하여 iSeries 서버와 PC 간에 복사된 모든 오브젝트는 EBCDIC와 ASCII 간에 자동으로 변환될 수 있습니다. EBCDIC은 확장 이진 코드화 십진 변환 코드(extended binary-coded decimal interchange code)이며, ASCII는 정보 교환을 위한 미 표준 코드(American National Standard Code for Information Interchange)입니다. 이러한 변환이 자동으로 이루어지도록 iSeries Access를 구성할 수 있으며, 또한 Client Access가 특정 확장자를 가진 파일에 대해 변환이 수행되도록 지정할 수도 있습니다. OS/400 V4R4에서, 파일에 대한 변환을 수행하기 위해 iSeries NetServer를 구성할 수도 있습니다.

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

PC를 사용하여 iSeries 서버에 연결된 경우, 통합 파일 시스템을 사용하여 PC에서 서버 디렉토리와 오브젝트를 사용할 수 있습니다. PC는 Windows 오퍼레이팅 시스템, FTP 클라이언트 또는 iSeries Navigator(iSeries Access의 일부)에 내장된 파일 공유 클라이언트를 사용하여 통합 파일 시스템의 파일에 대한 작업을 수행할 수 있습니다. PC는 Windows 파일 공유 클라이언트를 사용하여 iSeries 서버에서 실행되는 iSeries NetServer에 액세스합니다.

FTP를 사용하여 파일 전송
FTP 클라이언트를 사용하면 ″루트″(/), QSYS.LIB, 독립 ASP QSYS.LIB, QOpenSys, QOPT 및 QFileSvr.400 파일 시스템의 파일을 포함하여 iSeries 서버의 파일을 전송할 수 있습니다. 또한 문서 라이브러리 서비스(QDLS) 파일 시스템의 폴더 및 문서를 전송할 수도 있습니다.

iSeries Navigator를 사용하여 파일에 대해 작업
iSeries Access에는 iSeries 서버에 연결하여 통합 파일 시스템를 PC에서 사용할 수 있도록 하는 iSeries Navigator가 포함되어 있습니다. iSeries Navigator는 Windows 데스크탑에서 iSeries 서버를 관리하기 위한 그래픽 사용자 인터페이스입니다.

iSeries NetServer를 사용하여 파일에 대해 작업
iSeries NetServer는 Windows 클라이언트로 빌드된 파일 및 인쇄 공유가 서버에 대한 작업을 할 수 있도록 하는 OS/400의 한 부분입니다.

주: iSeries Access의 새 버전은 NetServer를 통해서만 통합 파일 시스템에 액세스합니다. NetServer 지원은 OS/400 V4R2 이상을 실행하는 iSeries 서버로의 TCP/IP 연결에만 사용할 수 있습니다.

FTP를 사용하여 파일 전송

FTP(File Transfer Protocol) 클라이언트를 사용하면 ″루트″(/), QOpenSys, QSYS.LIB, 독립 ASP QSYS.LIB, QOPT 및 QFileSvr.400 파일 시스템의 파일을 포함하여 iSeries 서버의 파일을 전송할 수 있습니다. 또한 문서 라이브러리 서비스(QDLS) 파일 시스템의 폴더 및 문서를 전송할 수도 있습니다. FTP 클라이언트는 클라이언트 파일에서 부속 명령을 읽고 이들 부속 명령에 대한 응답을 파일로 기록하는 무인 일괄처리 모드에서 대화식으로 실행할 수 있습니다. 또한 서버에서 파일을 조작하기 위한 다른 피처도 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)
- NetWare 파일 시스템(QNetWare)
- Windows NT Server 파일 시스템(QNTC)

그러나 다음과 같은 제한사항에 유의하십시오.

- 통합 파일 시스템은 FTP 지원을 파일 자료 전송으로만 제한합니다. 속성 자료를 전송하는 경우에는 FTP를 사용할 수 없습니다.
- QSYS.LIB 및 독립 ASP QSYS.LIB 파일 시스템은 FTP 지원을 실제 파일 멤버, 소스 실제 파일 멤버, 저장 파일 등으로만 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.

FTP에 대한 자세한 정보는 iSeries Information Center **네트워킹** 범주의 다음 주제를 참조하십시오.

- FTP
- FTP를 사용하여 파일 전송

iSeries NetServer를 사용하여 파일에 대해 작업

Windows Network Neighborhood용 iSeries 지원(iSeries NetServer)은 IBM Operating System/400 버전 5(OS/400)의 한 기능입니다. 이 기능을 사용하여 Windows 클라이언트는 OS/400 공유 디렉토리 경로 및 공유 출력 대기행렬에 액세스할 수 있습니다. iSeries NetServer를 사용하여 Windows 소프트웨어를 실행하는 PC가 iSeries에서 관리하는 자료 및 프린터에 액세스할 수 있습니다. 네트워크의 PC 클라이언트는 해당 오퍼레이팅 시스템에 포함된 파일 및 인쇄 공유 기능을 사용합니다. 따라서 iSeries NetServer를 사용하기 위해 추가 소프트웨어를 PC에 설치할 필요가 없습니다.

Samba 클라이언트 소프트웨어가 설치된 LINUX 클라이언트도 iSeries NetServer를 통해 자료 및 프린터에 액세스할 수 있습니다. iSeries에서 NFS 파일 시스템을 마운트하는 것과 유사한 방식으로 iSeries NetServer에서 Samba 파일 시스템(smbfs)을 마운트할 수 있습니다. 자세한 정보는 iSeries Information Center의 iSeries NetServer 주제를 참조하십시오.

iSeries NetServer 파일 공유는 iSeries NetServer가 iSeries 네트워크의 클라이언트와 공유하는 디렉토리 경로입니다. 파일 공유는 iSeries의 통합 파일 시스템 디렉토리로 구성됩니다. iSeries NetServer를 사용하여 파일 공유에 대한 작업을 하기 전에, iSeries NetServer 파일 공유를 작성하고 필요하면 iSeries Navigator를 사용하여 iSeries NetServer 파일 공유를 변경해야 합니다.

iSeries NetServer를 사용하여 통합 파일 시스템 파일 공유에 액세스하려면 다음을 수행하십시오.

1. **시작**을 마우스 오른쪽 단추로 클릭한 후 **탐색**을 선택하여 Windows PC에서 Windows 탐색기를 여십시오.
2. **도구** 메뉴를 열고 **네트워크 드라이브 연결**을 선택하십시오.
3. 사용하지 않는 드라이브 문자를 파일 공유용으로 선택하십시오(예: I:\ 드라이브).
4. iSeries NetServer 파일 공유 이름을 입력하십시오. 예를 들어 **\\QSYSTEM1\Sharename**과 같은 구문을 입력할 수 있습니다.

   주: 여기서 QSYSTEM1은 iSeries NetServer의 시스템명이며 Sharename은 사용할 파일 공유명입니다.
5. 확인을 클릭하십시오.

주: iSeries NetServer를 사용하여 연결할 때 서버명은 iSeries Access가 사용하는 이름과 다를 수 있습니다. 예를 들어 iSeries NetServer명은 QAS400X이며 파일에 대해 작업하기 위한 경로는 \\QAS400X\QDLS\MYFOLDER.FLR\MYFILE.DOC일 수 있습니다. 그러나, iSeries Access명은 AS400X이며 파일에 대해 작업하기 위한 경로는 \\AS400X\QDLS\MYFOLDER.FLR\MYFILE.DOC일 수 있습니다.

| iSeries NetServer를 사용하여 네트워크와 공유할 디렉토리를 선택하십시오. 해당 디렉토리가 서버명 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 iSeries 파일 시스템보다 탁월한 PC 파일 서비스 성능을 제공합니다. 파일을 ″루트″(/) 파일 시스템으로 이동할 수 있습니다. 자세한 정보는 다른 파일 시스템으로 오브젝트 이동시 고려사항을 참조하십시오.

| iSeries NetServer 및 파일 공유에 대한 자세한 정보는 iSeries Information Center **네트워킹** 범주의 다음 주제를 참조하십시오.

- iSeries NetServer
- iSeries NetServer file shares
- Accessing iSeries NetServer file shares with a Windows PC client

다른 파일 시스템으로 오브젝트 이동

통합 파일 시스템을 사용하기 전에 파일 시스템 간에 오브젝트을 이동하려면 37 페이지의 『다른 파일 시스템으로 오브젝트 이동시 고려사항』을(를) 검토하십시오.

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 주의깊게 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) 명령을 사용하여 선택한 파일 시스템으로 파일을 이동시키십시오.

   파일 시스템이 오브젝트 소유권 설정을 지원하는 경우, MOV가 오브젝트 소유권을 보유하므로 MOV 사용을 권장합니다. 그러나 OWNER(*KEEP) 매개변수를 사용하여 오브젝트 소유권을 보존하려면 CPY(복사) 명령을 사용할 수 있습니다. 이것은 오브젝트 소유자 설정을 지원하는 파일 시스템에 대해서만 작동한다는 점을 기억하십시오. MOV나 CPY를 사용할 때는 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.

다른 파일 시스템으로 오브젝트 이동시 고려사항

각 파일 시스템에는 고유한 특성이 있습니다. 그러나 다른 파일 시스템으로 오브젝트를 이동하면, 오브젝트가 현재 저장된 파일 시스템의 장점을 이용하지 못할 수도 있습니다. 해당 특성을 이용하기 위해 한 파일 시스템에서 다른 파일 시스템으로 오브젝트를 이동할 수 있습니다. 오브젝트를 다른 파일 시스템으로 이동하기 전에, 통합 파일 시스템의 파일 시스템 및 그 특성을 잘 알고 있어야 합니다. 자세한 내용은 4 페이지의 『통합 파일 시스템의 파일 시스템』을 참조하십시오.

또한 다음 사항을 고려해야 합니다.

- 현재 오브젝트가 들어 있는 파일 시스템의 장점을 사용하는 어플리케이션을 사용하고 있습니까?

  어떤 파일 시스템은 통합 파일 시스템 지원의 일부가 아닌 인터페이스를 지원합니다. 이러한 인터페이스를 사용하는 어플리케이션은 다른 파일 시스템으로 이동하는 오브젝트로 액세스할 수 없습니다. 예를 들어, QDLS나 QOPT 파일 시스템은 계층 파일 시스템(HFS)을 지원합니다. API와 명령으로 문서와 폴더 오브젝트에 대한 작업을 수행합니다. 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 속성의 저장 및 검색을 지원합니다. 따라서, QSYS.LIB 및 독립 ASP QSYS.LIB는 확장 속성을 가진 오브젝트를 저장할 만한 좋은 파일은 아닙니다. QDLS는 많은 『오피스』 속성을 지원하지만, 다른 파일 시스템은 그렇지 않습니다. 따라서, QDLS는 사용자의 오피스 문서를 저장하기 위한 좋은 장소입니다.

  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, QNetWare 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.

통합 파일 시스템에서 제공하는 디렉토리

통합 파일 시스템은 다음의 디렉토리가 존재하지 않는 경우, 시스템이 재시작될 때 디렉토리를 작성합니다.

/tmp /tmp 디렉토리는 어플리케이션이 임시 파일을 저장할 수 있는 장소를 제공합니다. 이 디렉토리는 /(『루트』) 디렉토리의 서브디렉토리이므로, 그 경로명은 /tmp입니다.

어플리케이션이 /tmp 디렉토리에 파일을 기록하면, 파일은 사용자나 해당 어플리케이션이 제거할 때까지 존재합니다. 시스템은 /tmp로부터 자동으로 파일을 제거하거나, /tmp에 있는 파일에 대해 다른 특별한 처리를 수행하지 않습니다.

사용자 화면 및 명령을 사용하여 /tmp 디렉토리 및 파일을 관리하기 위한 통합 파일 시스템을 지원할 수 있습니다. 예를 들어, 오브젝트 링크에 대한 작업 표시 화면이나 WRKLNK 명령을 사용하여 디렉토리의 파일 또는 /tmp 디렉토리를 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 /tmp 및 그 파일들을 관리하기 위하여 통합 파일 시스템을 지원하는 어플리케이션 프로그램 인터페이스(API)를 사용할 수 있습니다(54 페이지의 『API를 사용한 조작 수행』 참조). 예를 들어, 어플리케이션 프로그램은 unlink() API를 사용하여 /tmp 내의 파일을 제거할 수 있습니다.

/tmp가 제거되면 다음에 시스템이 재시작되는 동안 자동으로 다시 작성됩니다.

/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. 더 자세한 정보는 8 페이지의 『현재 디렉토리 및 홈 디렉토리』를 참조하십시오.

/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 사용권 프로그램 제품 자료용으로 사용되는 시스템 디렉토리입니다.

/QIBM/UserData

/QIBM/UserData 디렉토리는 구성 파일과 같은 사용권 프로그램 사용자 자료용으로 사용되는 시스템 디렉토리입니다.

/QOpenSys/QIBM

/QOpenSys/QIBM 디렉토리는 QOpenSys 파일 시스템용으로 사용되는 시스템 디렉토리입니다.

/QOpenSys/QIBM/ProdData

/QOpenSys/QIBM/ProdData 디렉토리는 QOpenSys 파일 시스템용 시스템 디렉토리이며 사용권 프로그램 제품 자료용으로 사용됩니다.

/QOpenSys/QIBM/UserData

/QOpenSys/QIBM/UserData 디렉토리는 QOpenSys 파일 시스템용 시스템 디렉토리이며 구성 파일과 같은 사용권 프로그램 사용자 자료용으로 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 사용권 프로그램 사용자 자료용으로 사용되는 시스템 디렉토리입니다. 여기서 asp_name은 독립 ASP 이름입니다.

제 4 장 iSeries Navigator를 사용한 통합 파일 시스템 액세스

iSeries Navigator는 Windows 데스크탑에서 시스템을 관리하기 위한 그래픽 사용자 인터페이스입니다. iSeries Navigator는 시스템을 보다 수월하게 운영 및 관리하여 생산성을 높일 수 있도록 합니다. 예를 들어, 한 iSeries 서버에서 다른 iSeries 서버로 사용자 프로파일을 끌면 사용자 프로파일을 다른 시스템에 복사할 수 있습니다. 마법사는 보안과 TCP/IP 서비스 및 어플리케이션 설정을 안내합니다.

iSeries Navigator를 사용하여 많은 타스크를 수행할 수 있습니다. 다음에는 시작하는 데 도움이 되는 몇 가지 일반적인 파일 시스템 타스크가 나열되어 있습니다.

파일 및 폴더에 대한 작업

- 44 페이지의『폴더 작성』
- 45 페이지의『폴더 제거』
- 42 페이지의『파일 체크 인』
- 42 페이지의『파일 체크 아웃』
- 42 페이지의『파일이나 폴더에 대한 권한 설정』
- 43 페이지의『텍스트 변환 설정』
- 43 페이지의『다른 시스템으로 파일 또는 폴더 송신』
- 44 페이지의『패키지 정의 옵션 변경』
- 44 페이지의『파일 또는 폴더 송신 날짜와 시간 스케줄』

파일 공유에 대한 작업

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

사용자 정의 파일 시스템에 대한 작업

- 46 페이지의『신규 사용자 정의 파일 시스템 작성』
- 46 페이지의『사용자 정의 파일 시스템 마운트』
- 47 페이지의『사용자 정의 파일 시스템 마운트 해제』

저널 오브젝트

- 47 페이지의『저널링 시작』
- 48 페이지의『저널링 종료』

파일 체크 인

파일을 체크 인하려면 다음 단계를 수행하십시오.

1. **iSeries Navigator**에서 체크 인할 파일을 오른쪽 마우스 버튼으로 클릭하십시오.
2. **등록 정보**를 선택하십시오.
3. **파일 등록 정보 -> 페이지 사용**을 선택하십시오.
4. **체크 인**을 클릭하십시오.

파일 체크 아웃

파일을 체크 아웃하려면 다음 단계를 수행하십시오.

1. **iSeries Navigator**에서 체크 아웃할 파일을 오른쪽 마우스 버튼으로 클릭하십시오.
2. **등록 정보**를 선택하십시오.
3. **파일 등록 정보 -> 페이지 사용**을 선택하십시오.
4. **체크 아웃**을 클릭하십시오.

파일이나 폴더에 대한 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.

파일이나 폴더에 대한 권한을 설정하려면 다음 단계를 수행하십시오.

1. **iSeries Navigator** 창에서 사용할 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. **통합 파일 시스템**을 확장하십시오. 권한이 추가될 오브젝트가 나타날 때까지 계속 확장하십시오.
4. 권한을 추가할 오브젝트를 오른쪽 마우스 버튼으로 클릭하고 **권한**을 선택하십시오.
5. **권한** 대화상자에서 **추가**를 클릭하십시오.
6. **추가** 대화상자의 사용자 또는 그룹명 필드에서 하나 이상의 사용자와 그룹을 선택하거나 사용자 또는 그룹명을 입력하십시오.
7. **확인**을 클릭하십시오. 사용자 또는 그룹이 리스트 맨 위에 추가됩니다.
8. 자세한 권한을 구현하려면 **세부사항** 버튼을 클릭하십시오.
9. 해당 선택란의 상자를 체크하여 사용자에게 원하는 권한을 적용하십시오.
10. **확인**을 클릭하십시오.

텍스트 변환 설정

iSeries Navigator에서 자동 텍스트 파일 변환을 설정할 수 있습니다. 자동 텍스트 파일 변환을 사용하면 파일 자료 변환용 파일 확장자를 사용할 수 있습니다. 통합 파일 시스템은 iSeries와 PC 사이에 자료 파일이 전송될 때 이 파일을 변환할 수 있습니다. PC에서 자료 파일에 액세스할 경우 이 파일은 ASCII 형식처럼 처리됩니다.

텍스트 변환을 설정하려면 다음 단계를 수행하십시오.

1. **iSeries Navigator**에서 사용할 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. **통합 파일 시스템**을 오른쪽 마우스 버튼으로 클릭하고 **등록 정보**를 선택하십시오.
4. **자동 텍스트 파일 변환을 위한 파일 확장자** 텍스트 상자에서 자동으로 변환할 파일 확장자를 입력하고 **추가**를 클릭하십시오.
5. 자동으로 변환할 모든 파일 확장자에 대해 4단계를 반복하십시오.
6. **확인**을 클릭하십시오.

다른 시스템으로 파일 또는 폴더 송신

다른 시스템으로 파일 또는 폴더를 송신하려면 다음 단계를 수행하십시오.

1. **iSeries Navigator**에서 사용할 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. **통합 파일 시스템**을 확장하십시오. 송신할 파일 또는 폴더가 나타날 때까지 계속 확장하십시오.
4. 파일 또는 폴더를 오른쪽 마우스 버튼으로 클릭하고 **송신**을 선택하십시오. 파일 또는 폴더가 **파일 송신 위치** 대화상자의 선택된 파일 및 폴더 리스트에 나타납니다.
5. 사용할 수 있는 시스템 및 그룹 리스트를 확장하십시오.
6. 시스템을 선택하고 **추가**를 선택하여 시스템을 **목표 시스템 및 그룹** 리스트에 추가하십시오. 이 파일 또는 폴더를 송신할 모든 파일 시스템에 대해 이 단계를 반복하십시오.
7. 현재 디폴트 패키지 정의 및 스케줄 정보와 함께 파일 또는 폴더를 송신하려면 **확인**을 클릭하십시오.

44 페이지의 『패키지 정의 옵션 변경』 또는 44 페이지의 『파일 또는 폴더 송신 날짜와 시간 스케줄』도 가능합니다.

패키지 정의를 작성하여 저장한 후, 정의된 일련의 파일 및 폴더를 복수 종료점 시스템이나 시스템 그룹으로 송신하기 위해 언제나 다시 사용할 수 있습니다. 파일의 스냅샷을 작성하면 동일한 파일 집합 사본에 대해 둘 이상의 버전을 보유할 수 있습니다. 스냅샷을 송신하면 분배하는 동안 파일에 대한 갱신이 이루어지지 않으므로 마지막 목표 시스템이 처음 목표 시스템과 동일한 오브젝트를 수신하게 됩니다.

패키지 정의 옵션 변경

패키지 정의를 사용하면 일련의 OS/400 오브젝트나 통합 파일 시스템 파일을 그룹화할 수 있습니다. 또한 패키지 정의를 사용하면 이후 분배를 위한 보존용으로 파일의 스냅샷을 작성하여 이 동일한 파일 그룹을 하나의 논리 집합이나 실제 집합으로 볼 수 있습니다.

패키지 정의 옵션을 변경하려면 다음 단계를 수행하십시오.

1. 43 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. 파일을 송신하려면 **확인**을 클릭하고 파일 송신 시간을 설정하려면 **스케줄**을 클릭하십시오.

관련 주제

- 『파일 또는 폴더 송신 날짜와 시간 스케줄』

파일 또는 폴더 송신 날짜와 시간 스케줄

스케줄러 기능을 사용하면 원하는 시간에 작업을 수행할 수 있습니다. 파일 또는 폴더 송신 날짜와 시간을 스케줄하려면 다음 단계를 수행하십시오.

1. 43 페이지의 『다른 시스템으로 파일 또는 폴더 송신』단계를 완료하십시오.
2. **스케줄**을 클릭하십시오.
3. 파일 또는 폴더 송신 시기 옵션을 선택하십시오.

폴더 작성

폴더를 작성하려면 다음 단계를 수행하십시오.

1. **iSeries Navigator**에서 사용할 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. **통합 파일 시스템**을 확장하십시오.
4. 새로운 폴더를 추가할 파일 시스템을 오른쪽 마우스 버튼으로 클릭하고 **신규 폴더**를 선택하십시오.
5. **신규 폴더** 대화상자에서 신규 오브젝트명을 입력하십시오.
6. **확인**을 클릭하십시오.

iSeries 서버에서 폴더를 작성할 경우 저널 관리를 사용하여 새 폴더(또는 오브젝트)를 보호할지 여부를 고려해야 합니다. 자세한 정보는 저널 관리 주제를 참조하십시오.

| 관련 주제

| • 저널링 시작

| • 저널링 종료

폴더 제거

폴더를 제거하려면 다음 단계를 수행하십시오.

1. **iSeries Navigator**에서 사용할 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. **통합 파일 시스템**을 확장하십시오. 제거할 파일 또는 폴더가 나타날 때까지 계속 확장하십시오.
4. 파일 또는 폴더를 오른쪽 마우스 버튼으로 클릭하고 **삭제**를 선택하십시오.

파일 공유 작성

| 파일 공유는 iSeries NetServer가 iSeries 네트워크의 PC 클라이언트와 공유하는 디렉토리 경로입니다. 파일
| 공유는 iSeries의 모든 통합 파일 시스템 디렉토리로 구성할 수 있습니다.

파일 공유를 작성하려면 다음 단계를 수행하십시오.

1. **iSeries Navigator**에서 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. **통합 파일 시스템**을 확장하십시오.
4. 공유를 작성할 폴더가 들어 있는 파일 시스템을 확장하십시오.
5. 공유를 작성할 폴더를 오른쪽 마우스 버튼으로 클릭하고 **공유**를 선택하십시오.
6. **신규 공유**를 선택하십시오.

파일 공유 변경

| 파일 공유는 iSeries NetServer가 iSeries 네트워크의 PC 클라이언트와 공유하는 디렉토리 경로입니다. 파일
| 공유는 iSeries의 모든 통합 파일 시스템 디렉토리로 구성할 수 있습니다.

파일 공유를 변경하려면 다음 단계를 수행하십시오.

1. **iSeries Navigator**에서 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. **통합 파일 시스템**을 확장하십시오.
4. 공유가 정의된 변경할 폴더를 확장하십시오.
5. **공유**가 정의된 폴더를 오른쪽 마우스 버튼으로 클릭하십시오.
6. **신규 공유**를 선택하십시오.

신규 사용자 정의 파일 시스템 작성

사용자 정의 파일 시스템(UDFS)은 사용자가 속성을 작성하고 정의하는 파일 시스템입니다. UDFS는 시스템 상의 보조 기억장치 풀에 상주합니다.

신규 사용자 정의 파일 시스템(UDFS)을 작성하려면 다음 단계를 수행하십시오.

1. **iSeries Navigator**에서 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. **통합 파일 시스템**을 확장하십시오.
4. **루트**를 확장하십시오.
5. **Dev**를 확장하십시오.
6. 신규 UDFS를 포함시킬 ASP(Auxiliary Storage Pool)를 클릭하십시오.
7. **파일** 메뉴에서 **신규 UDFS**를 선택하십시오.
8. **신규 사용자 정의 파일 시스템** 대화상자에서 UDFS 이름, 설명(선택적), 감사 값, 디폴트 파일 형식, 신규 UDFS에 있는 파일에 대소문자 구분 파일명이 있는지 여부를 지정하십시오.

사용자 정의 파일 시스템 마운트

사용자 정의 파일 시스템(UDFS)은 사용자가 속성을 작성하고 정의하는 파일 시스템입니다. UDFS는 시스템 상의 보조 기억장치 풀에 상주합니다. UDFS에 저장된 자료에 액세스하거나 자료를 보려면 UDFS를 마운트해야 합니다.

UDFS를 마운트하면 폴더 계층에서 마운트 지점 아래에 있는 모든 파일 시스템, 디렉토리 또는 오브젝트가 포함됩니다. 따라서 UDFS를 마운트 해제할 때까지 해당 파일 시스템, 디렉토리 또는 오브젝트에 액세스할 수 없습니다. 통합 파일 시스템의 모든 자료에 대한 액세스를 유지하려면 빈 폴더에 UDFS를 마운트하십시오. UDFS가 마운트된 후 UDFS 내의 파일은 해당 폴더 내에서 액세스할 수 있습니다. 폴더에서 작성된 변경사항은 포함된 폴더가 아닌 UDFS에 대한 변경사항이 됩니다.

주: 독립 ASP가 마운트될 수 없는 UDFS.

사용자 정의 파일 시스템(UDFS)을 마운트하려면 다음 단계를 수행하십시오.

1. **iSeries Navigator**에서 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. **통합 파일 시스템**을 확장하십시오.
4. **루트**를 확장하십시오.
5. **Dev**를 확장하십시오.
6. 마운트할 UDFS가 들어 있는 ASP(Auxiliary Storage Pool)를 클릭하십시오.
7. Operations Navigator의 오른쪽 분할 창에 있는 **UDFS 이름** 열에서 마운트할 UDFS를 오른쪽 마우스 버튼으로 클릭하십시오.

8. **마운트**를 선택하십시오.

끌어서 놓기를 사용하려면, 동일한 서버의 통합 파일 시스템 내의 폴더로 끌면 UDFS를 마운트할 수 있습니다. /dev, /dev/QASPxx, /dev/asp_name, 다른 시스템 또는 데스크탑에 UDFS를 드롭(drop)할 수는 없습니다.

사용자 정의 파일 시스템 마운트 해제

UDFS를 마운트하면 폴더 계층에서 마운트 지점 아래에 있는 모든 파일 시스템, 디렉토리 또는 오브젝트가 포함됩니다. 따라서 UDFS를 마운트 해제할 때까지 해당 파일 시스템, 디렉토리 또는 오브젝트에 액세스할 수 없습니다.

사용자 정의 파일 시스템(UDFS)을 마운트 해제하려면 다음 단계를 수행하십시오.

1. **Operations Navigator**에서 시스템을 확장하십시오.
2. **파일 시스템**을 확장하십시오.
3. **통합 파일 시스템**을 확장하십시오.
4. **루트**를 확장하십시오.
5. **Dev**를 확장하십시오.
6. 마운트 해제할 UDFS가 들어 있는 ASP(Auxiliary Storage Pool)를 클릭하십시오.
7. iSeries Navigator의 오른쪽 분할 창에 있는 **UDFS 이름** 열에서 마운트 해제할 UDFS를 오른쪽 마우스 버튼으로 클릭하십시오.
8. **마운트 해제**를 선택하십시오.

저널링 시작

저널링의 1차 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 회복할 수 있도록 하는 것입니다.

오브젝트에서 저널링을 시작하려면 다음을 수행하십시오.

1. **iSeries 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.

통합 파일 시스템 오브젝트 저널링에 대한 자세한 내용은 iSeries Information Center에서 저널 관리를 참조하십시오.

저널링 종료

저널링의 1차 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 회복할 수 있도록 하는 것입니다. 오브젝트에서 저널링을 시작하는 방법에 대한 자세한 정보는 저널링 시작을 참조하십시오. 오브젝트에서 저널링을 시작한 이후, 이 오브젝트의 저널링을 종료하려는 경우가 있을 수 있습니다.

오브젝트에서 저널링을 종료하려면 다음을 수행하십시오.

1. **iSeries 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.

통합 파일 시스템 오브젝트 저널링에 대한 자세한 내용은 iSeries Information Center에서 저널 관리를 참조하십시오.

제 5 장 통합 파일 시스템에 대한 프로그래밍 지원

통합 파일 시스템을 iSeries 서버에 추가해도 기존 iSeries 서버 어플리케이션에 영향을 주지 않습니다. 프로그래밍 언어, 유틸리티 및 자료 설명 스펙과 같은 시스템 지원은 통합 파일 시스템이 추가되기 전과 마찬가지로 작동됩니다.

그러나 스트림 파일, 디렉토리 및 기타 통합 파일 시스템의 지원을 이용하기 위해서는 통합 파일 시스템 기능에 액세스하기 위해 제공되는 C 언어 어플리케이션 프로그램 인터페이스(API) 세트를 사용해야 합니다.

또한 통합 파일 시스템 추가를 통해 실제 데이터베이스 파일과 스트림 파일 사이에 자료를 복사할 수 있습니다. CL 명령, iSeries Access의 자료 전송 기능 또는 API를 사용하여 이러한 복사를 수행할 수 있습니다.

다음 주제에서는 통합 파일 시스템 스트림 파일에서 복사 기능을 사용하는 방법을 설명하며, 통합 파일 시스템 함수에 액세스하는 한 방법으로 API를 소개합니다.

- 스트림 파일과 데이터베이스 파일 사이의 자료 복사
- 스트림 파일과 저장 파일 간에 자료 복사
- API를 사용한 자료 복사
- 소켓 지원
- 명명 및 국제적 지원
- 자료 변환

스트림 파일과 데이터베이스 파일 사이의 자료 복사

자료 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 자료에 액세스하기 위해서는 바이트 오프셋과 길이를 표시합니다. 사용자는 데이터베이스 파일내 자료에 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 시스템으로부터 도움을 거의 받지 못하거나 전혀 받지 못합니다. 즉, 스트림 파일은 일부 프로그래밍 상황에 보다 적합한 반면, 레코드 지향 파일은 다른 프로그래밍 상황에 보다 적합합니다.

통합 파일 시스템에서 스트림 파일과 데이터베이스 파일 간에 자료를 복사하는 몇 가지 방법이 있습니다.

- CL 명령을 사용한 자료 복사
- API를 사용한 자료 복사

- 자료 전송 기능을 사용한 자료 복사

CL 명령을 사용한 자료 복사

스트림 파일과 데이터베이스 파일 멤버 간에 자료를 복사할 수 있도록 하는 두 개의 CL 명령 세트가 있습니다.

- CPYTOSTMF 및 CPYFRMSTMF
- CPYTOIMPF 및 CPYFRMIMPF

CPYTOSTMF 및 CPYFRMSTMF 명령

CPYFRMSTMF(스트림 파일로부터 복사)와 CPYTOSTMF(스트림 파일로 복사) 명령을 사용하여 스트림 파일 및 데이터베이스 파일 멤버 사이에 자료를 복사할 수 있습니다. CPYTOSTMF 명령을 사용하여 데이터베이스 파일 멤버로부터 스트림 파일을 작성할 수 있습니다. 또한, CPYFRMSTMF 명령을 사용하여 스트림 파일로부터 데이터베이스 파일 멤버를 작성할 수도 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 자료를 변환 및 재형식화하기 위한 다양한 옵션을 제공합니다.

CPYTOSTMF 및 CPYFRMSTMF 명령을 사용하여 스트림 파일과 저장 파일 간에 자료를 복사할 수도 있습니다.

CPYTOIMPF 및 CPYFRMIMPF 명령

CPYTOIMPF(가져오기 파일로 복사)와 CPYFRMIMPF(가져오기 파일에서 복사) 명령을 사용하여 스트림 파일과 데이터베이스 파일 멤버 간에 자료를 복사할 수도 있습니다. CPYTOSTMF 및 CPYFRMSTMF 명령을 사용하여 복잡한 외부 설명(DDS 설명) 데이터베이스 파일에서 자료를 이동할 수 없습니다. *가져오기 파일*이란 단어는 스트림 유형 파일을 말하며, 이 용어는 일반적으로 이종 데이터베이스 간에 자료를 복사하려는 목적으로 작성되는 파일을 말합니다.

스트림(또는 가져오기) 파일에서 복사할 때, CPYFRMIMPF 명령을 사용하여 스트림 파일의 자료를 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 자료 유형을 변환하기 위한 옵션도 제공됩니다.

목표 스트림 파일이나 데이터베이스 멤버가 이미 존재하는 경우 이들 명령에 자료 변환이 제공됩니다. 파일이 존재하지 않는 경우, 다음 두 단계 메소드를 사용하여 자료를 변환할 수 있습니다.

1. CPYTOIMPF 및 CPYFRMIMPF 명령을 사용하여 외부 설명 파일과 소스 실제 파일 사이에서 자료를 복사하십시오.
2. CPYTOSTMF 및 CPYFRMSTMF 명령(목표 파일의 존재 여부에 관계없이 전체 자료 변환 제공)을 사용하여 소스 실제 파일과 스트림 파일 사이에서 자료를 복사하십시오.

예는 다음과 같습니다.

```
CPYTOIMPF  FROMFILE(DB2FILE) TOFILE(EXPFILE) DTAFMT(*DLM)
        FLDDLM(';') RCDDLM(X'07') STRDLM('"') 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 서버에서 실행되기 때문에 유용합니다. 그러나 인터페이스가 복잡합니다.

자세한 정보는 명령 도움말이나 iSeries Information Center의 CL(Command language) 주제를 참조하십시오.

API를 사용한 자료 복사

어플리케이션에서 데이터베이스 파일 멤버를 스트림 파일에 복사하는 경우, 통합 파일 시스템 open(), read() 및 write() 함수를 사용하여 멤버를 열어 자료를 읽고 기록할 수 있습니다. 자세한 정보는 iSeries Information Center의 통합 파일 시스템 API 주제를 참조하십시오.

자료 전송 기능을 사용한 자료 복사

iSeries Access 자료 전송 어플리케이션은 따라하기 쉬운 그래픽 인터페이스, 자동 숫자 및 문자 자료 변환의 장점이 있습니다. 그러나 자료 전송을 하려면 iSeries Access 제품을 설치해야 하며, PC와 iSeries 서버 자원 모두를 사용하고 이들 사이의 통신이 필요합니다.

PC와 서버에 iSeries Access를 설치한 경우, 자료 전송 어플리케이션을 사용하여 스트림 파일과 데이터베이스 파일 간에 자료를 전송할 수 있습니다. 기존 데이터베이스 파일을 기반으로 신규 데이터베이스 파일이나 외부 서술 데이터베이스 파일로 자료를 이동시키거나 신규 데이터베이스 파일 정의와 파일로 전송할 수도 있습니다.

| 다음 타스크를 통해 자료 전송 어플리케이션으로 자료를 전송하거나 복사할 수 있습니다.

- 데이터베이스 파일에서 스트림 파일로 자료 전송
- 스트림 파일에서 데이터베이스 파일로 자료 전송
- 새로 작성된 데이터베이스 파일 정의와 파일로 자료 전송
- 형식 설명 파일 작성

데이터베이스 파일에서 스트림 파일로 자료 전송

데이터베이스 파일에서 서버의 스트림 파일로 파일을 전송하려면 다음의 단계를 수행하십시오.

1. 서버로의 연결을 설정하십시오.
2. iSeries 파일 시스템의 해당 경로에 네트워크 드라이브를 맵핑시키십시오.
3. Windows용 iSeries Access 창에서 **iSeries 서버에서 자료 전송**을 선택하십시오.

4. 전송 서버를 선택하십시오.
5. iSeries 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. 자료 전송은 ASCII 텍스트, BIFF3, CSV, DIF, 탭 분리 텍스트 또는 WK4와 같은 일반 PC 파일 유형을 지원합니다.
6. 파일 전송을 실행하려면 **iSeries에서 자료 전송**을 클릭하십시오.

또한 자료 전송 어플리케이션을 사용하여 일괄처리 작업으로 자료 이동을 수행할 수 있습니다. 위와 같이 진행하되, **파일** 메뉴 옵션을 선택하여 전송 요구를 저장하십시오. iSeries 서버로의 자료 전송 어플리케이션은 .DTT 또는 .TFR 파일을 작성합니다. iSeries 서버에서 자료 전송 어플리케이션은 .DTF 또는 .TTO 파일을 작성합니다. iSeries Access 디렉토리에서, 다음과 같이 명령행에서 두 프로그램을 일괄처리로 실행할 수 있습니다.

- RTOPCB는 .DTF 또는 .TTO 파일을 매개변수로서 갖습니다.
- RFROMPCB는 .DTT 또는 .TFR 파일을 매개변수로서 갖습니다.

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 자료 전송

스트림 파일에서 서버의 데이터베이스 파일로 자료를 전송하려면 다음을 수행하십시오.

1. 서버로의 연결을 설정하십시오.
2. iSeries 파일 시스템의 해당 경로에 네트워크 드라이브를 맵핑시키십시오.
3. Windows용 iSeries Access 창에서 **iSeries 서버로 자료 전송**을 선택하십시오.
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 서버를 선택하십시오.
6. 파일 전송을 실행하려면 **iSeries 서버로 자료 전송**을 클릭하십시오.

   주: 자료를 서버의 기존 데이터베이스 파일 정의로 이동시키는 경우, iSeries 서버어플리케이션으로 자료전송은 연관된 형식 설명 파일(FDF)을 사용할 것을 요구합니다. FDF 파일은 스트림 파일의 형식을 기술하며, 자료가 데이터베이스 파일에서 스트림 파일로 전송될 때 iSeries 서버로의 자료 전송 어플리케이션 에 의해 작성됩니다. 스트림 파일에서 데이터베이스 파일로 자료 전송을 완료하려면, **iSeries으로 자료 전송**을 클릭하십시오. 기존 .FDF 파일을 사용할 수 없는 경우, 신속하게 .FDF 파일 작성을 할 수 있습니다.

또한 자료 전송 어플리케이션을 사용하여 일괄처리 작업으로 자료 이동을 수행할 수 있습니다. 위와 같이 진행하되, **파일** 메뉴 옵션을 선택하여 전송 요구를 저장하십시오. iSeries 서버로의 자료 전송 어플리케이션은 .DTT 또는 .TFR 파일을 작성합니다. iSeries 서버에서 자료 전송 어플리케이션은 .DTF 또는 .TTO 파일을 작성합니다. iSeries Access 디렉토리에서, 다음과 같이 명령행에서 두 프로그램을 일괄처리로 실행할 수 있습니다.

- RTOPCB는 .DTF 또는 .TTO 파일을 매개변수로서 갖습니다.
- RFROMPCB는 .DTT 또는 .TFR 파일을 매개변수로서 갖습니다.

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.

새로 작성된 데이터베이스 파일 정의와 파일로 자료 전송

새로 작성된 데이터베이스 파일 정의와 파일로 자료를 전송하려면 다음 단계를 수행하십시오.

1. 서버로의 연결을 설정하십시오.
2. iSeries 파일 시스템의 해당 경로에 네트워크 드라이브를 맵핑시키십시오.
3. Windows용 iSeries Access 창에서 **iSeries 서버로 자료 전송**을 선택하십시오.
4. iSeries 서버로의 자료 전송 어플리케이션의 툴 메뉴를 여십시오.
5. **iSeries 데이터베이스 파일 작성**을 선택하십시오.

   기존 PC 파일로부터 신규 iSeries 데이터베이스 파일을 작성할 수 있는 마법사가 나타납니다. iSeries 파일이 기반을 두게 될 PC 파일명, 작성할 iSeries 파일명, 기타 몇 가지 필수 세부사항을 지정해야 합니다. 이 툴은 주어진 스트림 파일을 분석하여 결과 데이터베이스 파일에서 필수적인 필드의 수, 유형 및 크기를 판별합니다. 그러면 툴이 서버에 데이터베이스 파일 정의를 작성할 수 있게 됩니다.

형식 설명 파일 작성

자료를 서버의 기존 데이터베이스 파일 정의로 이동시키는 경우, iSeries 서버로의 자료 전송 어플리케이션은 연관된 형식 설명 파일(FDF)을 사용할 것을 요구합니다. FDF 파일은 스트림 파일의 형식을 기술하며, 자료가 데이터베이스 파일에서 스트림 파일로 전송될 때 iSeries 서버에서 자료 전송 어플리케이션에 의해 작성됩니다.

.FDF 파일을 작성하려면 다음 단계를 수행하십시오.

1. 소스 스트림 파일과 대응하는 형식(필드의 수, 자료의 유형)으로 외부 설명 데이터베이스 파일을 작성하십시오.
2. 데이터베이스 파일 내에 하나의 임시 자료 레코드를 작성하십시오.
3. 데이터베이스 파일에서 스트림 파일 및 연관된 .FDF 파일을 작성하려면 iSeries 서버에서 자료 전송 기능을 사용하십시오.
4. 이제, iSeries 서버로 자료 전송 기능을 사용할 수 있습니다. 전송하려는 소스 스트림 파일로 .FDF 파일을 지정할 수 있습니다.

스트림 파일과 저장 파일 간에 자료 복사

저장 파일은 그 밖의 테이프나 디스켓에 작성될 자료를 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 명령을 사용하여 스트림 파일 간에 저장 파일을 복사할 수 있습니다. 그러나 다시 저장 파일 오브젝트로 스트림 파일을 복사할 때, 자료는 유효한 저장 파일 자료(저장 파일에서 비롯되어 스트림 파일로 복사된 자료)여야 합니다.

또한 PC 클라이언트를 사용하여 저장 파일에 액세스하고 PC 기억장치나 LAN으로 자료를 복사할 수 있습니다. 그러나 네트워크 파일 시스템(NFS)을 통해 저장 파일의 자료에 액세스할 수 없다는 점을 유의하십시오.

API를 사용한 조작 수행

통합 파일 시스템 디렉토리와 스트림 파일상에서 조작을 수행하는 API(Application Program Interfaces)는 C 언어 함수 형식입니다. 다음 두 세트의 함수 중 하나를 선택하여 통합 언어 환경(ILE) C/400으로 작성되는 프로그램에서 사용할 수 있습니다.

- OS/400에 포함되어 있는 통합 파일 시스템 C 함수
- ILE C/400 사용권 프로그램이 제공하는 C 함수

통합 파일 시스템 기능은 통합 파일 시스템 스트림 I/O 지원을 통해서만 작동합니다. 다음 API가 지원됩니다.

표 3. 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(큰 파일 작동가능)
DosSetFileLocks()	파일의 바이트 범위 잠그기 및 잠금 해제
DosSetFileLocks64()	파일의 바이트 범위 잠그기 및 잠금 해제(큰 파일 작동가능)
DosSetRelMaxFH()	파일 설명자의 최대 수 변경
dup()	개방 파일 설명자 복사
dup2()	다른 설명자로 개방 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
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(큰 파일 작동가능)
fstatvfs()	설명자를 사용한 정보 가져오기
fstatvfs64()	설명자를 사용한 정보 가져오기(64비트 작동가능)
fsync()	파일에 대한 변경 동기화
ftruncate()	파일 절단
ftruncate64()	파일 절단(큰 파일 작동가능)

표 3. 통합 파일 시스템 API (계속)

함수	설명
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
getgrnam()	그룹명을 사용한 그룹 정보 가져오기
getgroups()	그룹 ID 가져오기
getwpnam()	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(큰 파일 작동가능)
lstat()	파일 또는 링크 정보 가져오기
lstat64()	파일 또는 링크 정보 가져오기(큰 파일 작동가능)
mmap()	메모리 맵 작성
mmap64()	메모리 맵 작성(큰 파일 작동가능)
mprotect()	메모리 맵 보호 변경
msync()	메모리 맵 동기화
munmap()	메모리 맵 제거
mkdir()	디렉토리 작성
mkfifo()	FIFO 특수 파일 작성
open()	파일 열기
open64()	파일 열기(큰 파일 작동가능)
opendir()	디렉토리 열기
pathconf()	구성가능 경로명 변수 가져오기
pipe()	소켓을 사용한 프로세스간 채널 작성
pread()	오프셋을 사용하여 설명자에서 읽기
pread64()	오프셋을 사용하여 설명자에서 읽기(큰 파일 사용)
pwrite()	오프셋을 사용하여 설명자에 쓰기
pwrite64()	오프셋을 사용하여 설명자에 쓰기(큰 파일 사용)
QjoEndJournal()	저널링 종료
QjoRetrieveJournal Information()	저널 정보 검색
QJORJIDI()	저널 ID 정보 검색
QJOSJRNE()	저널 항목 송신
QjoStartJournal()	저널링 시작
QlgAccess()	파일 액세스 가능성 판별(NLS 작동가능 경로명 사용)

표 3. 통합 파일 시스템 API (계속)

함수	설명
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)
QlgCreat64()	신규 파일 작성 또는 기존 파일 다시쓰기(큰 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(큰 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(큰 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 자료 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)

표 3. 통합 파일 시스템 API (계속)

함수	설명
QlgStat64()	파일 정보 가져오기(큰 파일 작동가능 및 NLS 작동가능 경로명 사용)
QlgStatvfs()	파일 시스템 정보 가져오기(NLS 작동가능 경로명 사용)
QlgStatvfs64()	파일 시스템 정보 가져오기(큰 파일 작동가능 및 NLS 작동가능 경로명 사용)
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
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
Qp0lSaveStgFree()	오브젝트 자료 저장 및 기억장치 비우기
Qp0lSetAttr()	오브젝트에 대한 시스템 속성 설정
Qp0lUnlink()	NLS 작동가능 경로명이 있는 파일 링크 해제
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(큰 파일 작동가능)
statvfs()	파일 시스템 정보 가져오기

표 3. 통합 파일 시스템 API (계속)

함수	설명
statvfs64()	파일 시스템 정보 가져오기(큰 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 OS/400 소켓에 대해서도 사용됩니다. 특정 파일 시스템에 대한 이러한 함수 사용의 제한사항은 4 페이지의 『통합 파일 시스템의 파일 시스템』을 참조하십시오. 통합 파일 시스템 C 기능을 사용하는 예제 프로그램은 113 페이지의 부록 B 『통합 파일 시스템 C 함수를 사용한 프로그램 예』를 참조하십시오.

통합 파일 시스템 API에 대한 자세한 정보는 다음 주제를 참조하십시오.

- ILE C/400 함수
- API에 대한 큰 파일 지원
- API에 대한 경로명 규칙
- 파일 설명자
- 보안
- iSeries Information Center에서 API(Application programming interfaces) 주제

ILE C/400 함수

ILE C/400은 미국 표준 협회(ANSI)가 정의하는 표준 C 함수를 제공합니다. 이들 함수는 C 프로그램 작성시 사용자 지정에 따라, 자료 관리 I/O 지원이나 통합 파일 시스템 스트림 I/O 지원을 통해 작동할 수 있습니다. 컴파일러는 사용자가 다르게 지시하지 않는 한, 자료 관리 I/O를 사용합니다.

컴파일러가 통합 파일 시스템 스트림 I/O를 사용하도록 지시하려면, CRTCMOD(ILE C/400 모듈 작성) 또는 CRTBNDC(바인드된 C 프로그램 작성) 명령의 SYSINCOPT(시스템 인터페이스 옵션) 매개변수에 대하여 *IFSIO를 지정해야 합니다. *IFSIO를 지정할 때, 통합 파일 시스템 I/O 함수는 자료 관리 I/O 함수 대신 바인드됩니다. 사실상 ILE C/400 C 함수는 I/O를 수행하기 위해 통합 파일 시스템 함수를 사용합니다.

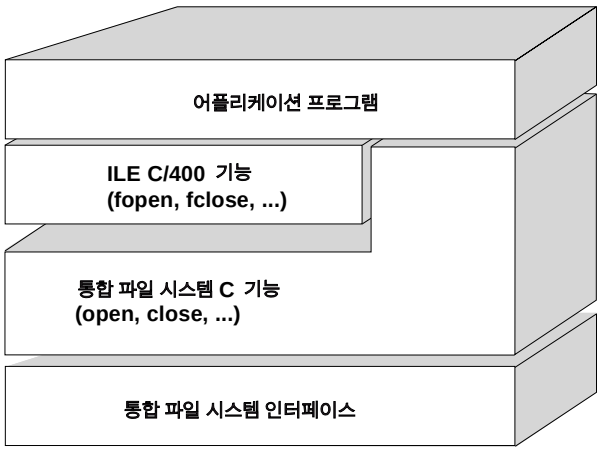


그림 10. ILE C/400 함수는 통합 파일 시스템 스트림 I/O 함수를 사용합니다.

통합 파일 시스템 스트림 I/O로 ILE C/400 함수를 사용하는 것에 대한 자세한 정보는 WebSphere Development Studio: ILE C/C++ 프로그래머 안내서 를 참조하십시오. 각 ILE C/400 C 함수에 대한 자세한 내용은 WebSphere Development Studio: C/C++ 언어 참조서 를 참조하십시오.

API에 대한 더 큰 파일 지원

| 어플리케이션이 매우 큰 파일을 저장 및 처리할 수 있도록 통합 파일 시스템 API가 향상되었습니다. 통합 파
| 일 시스템은 "루트"(/), 개방 시스템 파일 시스템(QOpenSys) 및 사용자 정의 파일 시스템에서 최대 256GB의
| 스트림 파일 크기를 허용합니다.

통합 파일 시스템은 64비트 UNIX 유형 API 세트를 제공하며 8바이트 정수 인수를 사용하여 큰 파일 크기 및 오프셋에 액세스할 수 있는 64비트 API 대 기존 32비트 API의 맵핑을 쉽게 할 수 있게 합니다. 각각의 64비트 API에 대한 세부사항은 iSeries Information Center의 통합 파일 시스템 API 주제를 참조하십시오.

어플리케이션이 큰 파일 지원을 사용할 수 있도록 다음이 제공됩니다.

1. 매크로 레이블 _LARGE_FILE_API가 컴파일 시간에 정의되는 경우, 어플리케이션은 64비트를 사용할 수 있는 API 및 자료 구조에 액세스합니다. 예를 들어, stat64() API 및 stat64 구조를 사용하려는 어플리케이션은 컴파일 시간에 _LARGE_FILE_API를 정의해야 합니다.
2. 어플리케이션이 매크로 레이블 _LARGE_FILES를 컴파일 시간에 정의하는 경우, 기존 API 및 자료 구조는 해당 64비트 버전에 맵핑됩니다. 예를 들어, 어플리케이션이 컴파일 시간에 _LARGE_FILES를 정의하는 경우, stat() API에 대한 호출은 stat64() API에 맵핑되며 stat() 구조에 대한 호출은 stat64() 구조에 맵핑됩니다.

큰 파일 지원을 사용하려는 어플리케이션은 컴파일 시간에 _LARGE_FILE_API를 정의하여 코드가 64비트 API에 직접 코드화하거나 컴파일 시간에 _LARGE_FILES를 정의할 수 있습니다. 그런 다음, 모든 적절한 API 및 자료 구조는 64비트 버전에 자동으로 맵핑됩니다.

큰 파일 지원을 사용하지 않으려는 어플리케이션은 영향을 받지 않으며 변경 내용없이 통합 파일 시스템 API를 계속 사용할 수 있습니다.

API에 대한 경로명 규칙

오브젝트상에서 조작하기 위해 통합 파일 시스템 또는 ILE C/400 API 사용시, 디렉토리 경로를 제공함으로써 오브젝트를 식별합니다. 다음은 API에 경로명을 지정할 때 유의해야 할 규칙들을 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
  ```

  역슬래시(\)는 분리자로서 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. 각 파일 시스템의 한계는 65 페이지의 『파일 시스템 비교』를 참조하십시오.
- 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.
- iSeries 서버 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 적용된다는 점을 유의하십시오.
- QSYS.LIB 파일 시스템의 오브젝트에 대한 조작시, 구성요소명은 *name.object-type*의 형식이어야 하며, 예는 다음과 같습니다.

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

  보다 자세한 내용은 76 페이지의 『라이브러리 파일 시스템(QSYS.LIB)』을 참조하십시오.
- 독립 ASP QSYS.LIB 파일 시스템의 오브젝트에 대한 조작시, 구성요소명은 *name.object-type*의 형식이어야 하며, 예는 다음과 같습니다.

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

  보다 자세한 내용은 79 페이지의 『독립 ASP QSYS.LIB』을 참조하십시오
- 경로명에 콜론(:)을 사용하지 마십시오. 콜론은 시스템 내에서 특별한 의미가 있습니다.

- 통합 파일 시스템 명령의 경로명(31 페이지의 『CL 명령 및 표시장치에 대한 경로명 규칙』 참조)과 달리 별표(*), 물음표(?), 작은 따옴표('), 인용 부호(") 및 틸드(~)에는 특별한 의미가 없습니다. 이들은 단지 이름에서 다른 여러 문자처럼 처리됩니다. QjoEndJournal() 및 QjoStartJournal API만 이 규칙이 적용되지 않습니다.

파일 설명자

파일상에서 조작을 수행하기 위해 미국 표준 협회(ANSI)가 지정한 대로 ILE C/400 스트림 I/O 함수를 사용하면, 포인터를 사용하여 파일을 인식하게 됩니다. 통합 파일 시스템 C 함수를 사용할 때는 **파일 설명자**를 지정함으로써 파일을 식별합니다. 파일 설명자는 각 작업에 고유한 양의 정수입니다. 이 작업은 파일에서 조작을 수행할 때 개방 파일을 식별하기 위해 파일 설명자를 사용합니다. 파일 설명자는 통합 파일 시스템상에서 작동되는 C 함수의 변수 *fildes*와 소켓상에서 작동되는 C 함수의 변수 *descriptor*에 의해 표시됩니다.

각 파일 설명자는 파일 오프셋, 파일 상태, 파일에 대한 액세스 모드와 같은 정보가 들어 있는 **개방 파일 설명**을 말합니다. 동일한 개방 파일 설명은 하나 이상의 파일 설명자에 의해 참조될 수 있으나, 파일 설명자는 하나의 개방 파일 설명만을 참조할 수 있습니다.

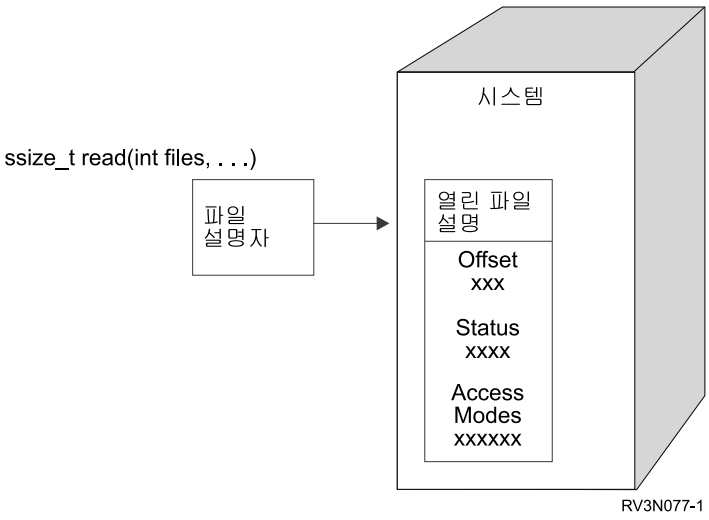


그림 11. 파일 설명자 및 개방 파일 설명

통합 파일 시스템과 함께 ILE C/400 스트림 I/O 함수가 사용되는 경우, ILE C/400 실행시 지원은 파일 포인터를 파일 설명자로 변환합니다.

『루트』(/), QOpenSys 파일 시스템, 사용자 정의 파일 시스템을 사용할 때, 작업이 파일에 액세스할 수 있도록 한 작업에서 다른 작업으로 개방 파일 설명에 액세스를 전달할 수 있습니다. 이는 작업간의 파일 설명자를 전달하기 위해 givedescriptor() 또는 takedescriptor() 함수를 사용함으로써 이루어집니다. 이 함수들에 대한 설명은 소켓 프로그래밍 또는 iSeries Information Center의 소켓 API 주제를 참조하십시오.

보안

통합 파일 시스템 API 사용시 자료 관리 인터페이스와 마찬가지로 오브젝트에 대한 액세스를 제한할 수 있습니다. 그러나 채택 권한은 지원되지 않는 것에 유의하십시오. 통합 파일 시스템 API는 작업이 수행중인 사용자 프로파일의 권한을 사용합니다.

각 파일 시스템에는 각각의 특수 권한 조건이 있을 수 있습니다. NFS 서버 작업은 이 규칙에 대한 유일한 예외입니다. 네트워크 파일 시스템 서버 요구는 사용자의 ID(UID) 번호가 요구 NFS 서버에 의해 수신되었던 사용자의 프로파일에 수행됩니다.

서버에 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 모드』를 구성하는 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 수 있습니다.

자료 보안 및 권한에 대한 세부사항은 보안 -- 참조서  를 참조하십시오.

소켓 지원

사용자의 어플리케이션이 『루트』(/), QOpenSys, 사용자 정의 파일 시스템을 사용하고 있는 경우, 통합 파일 시스템 **로컬 소켓** 지원의 장점을 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 bind() C 언어 함수를 사용함으로써 연결 지점을 설치하여 로컬 소켓 오브젝트를 작성하는 것입니다. 다른 작업은 connect(), sendto() 또는 sendmsg() 함수상에서 로컬 소켓 오브젝트명을 지정하는 것입니다. 이 함수들과 일반적인 소켓에 대한 개념은 iSeries Information Center의 소켓 프로그래밍 주제에서 설명합니다.

연결된 후에 write() 및 read()와 같은 통합 파일 시스템 함수를 사용하여 서로간의 자료 전송 및 수신 작업이 이루어질 수 있습니다. 전송된 자료 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() 함수나 RMVLNK(링크 제거) 명령을 사용하여 제거될 때까지 시스템에 남아 있습니다.

로컬 소켓 오브젝트는 저장될 수 없습니다.

명명 및 국제적 지원

오브젝트명의 문자는 『루트』(/) 및 QOpenSys 파일 시스템 지원으로 인해 서로 다른 자국어 지원 제품 및 장치에 사용되는 인코딩 체계와는 상관없이 상수로 남아 있게 됩니다. 오브젝트명이 시스템으로 전달될 때, 오브젝트명의 각 문자는 모든 문자가 표준 코드 표시가 되도록 16비트 형식으로 변환됩니다(24 페이지의 『이름 연속성』 참조). 이름을 사용하는 경우, 코드 페이지에 적합한 코드 형식으로 변환됩니다.

이름이 변환되는 코드 페이지에 이름에 사용된 문자가 들어 있지 않은 경우, 그 이름은 유효하지 않은 것으로 거부됩니다.

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.

코드 페이지에 대한 자세한 정보는 iSeries Information Center의 글로벌화 주제를 참조하십시오.

자료 변환

통합 파일 시스템를 통해 파일에 액세스할 때, 파일의 자료는 파일이 열릴 때 요청되는 열린 모드에 따라 변환될 수도 있고 변환되지 않을 수도 있습니다

개방 파일은 다음 두 가지 열림 모드 중 하나가 될 수 있습니다.

2진 변환되지 않고 파일로부터 자료가 읽히고 파일로 자료가 기록됩니다. 어플리케이션은 자료 처리를 담당합니다.

텍스트 자료는 텍스트 양식으로 가정되어 파일에서 읽히고 파일에 기록됩니다. 파일에서 자료를 읽을 때 자료는 파일의 코드화 문자 세트 ID(CCSID)에서 자료를 받는 시스템, 어플리케이션 또는 작업의 CCSID로 변환됩니다. 파일에 자료를 기록할 때 자료는 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 사용중인 레코드 파일로부터 읽을 때, 행 종료 문자(캐리지 리턴 및 라인 피드)는 각 레코드의 자료 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(자료 실제 파일 멤버인 경우)로 채워집니다.

열기 요청시 다음 중 하나가 지정될 수 있습니다.

2진, 강제

실제 파일 내용에 관계없이 자료는 2진으로 처리됩니다. 어플리케이션이 자료 처리 방법을 알아냅니다.

텍스트, 강제

자료는 텍스트로 간주됩니다. 자료는 파일의 CCSID에서 어플리케이션의 CCSID로 변환됩니다.

2진, 강제 디폴트가 통합 파일 시스템 open() 함수에 대해 사용됩니다

제 6 장 통합 파일 시스템의 파일 시스템

통합 파일 시스템의 파일 시스템은 다음과 같습니다.

- 『루트』
- QOpenSys
- UDFS
- QSYS.LIB
- 독립 ASP QSYS.LIB
- QDLS
- QOPT
- QNetWare
- QNTC
- QFileSvr.400
- NFS

각 파일 시스템에 대한 개요는 파일 시스템 비교를 참조하십시오.

파일 시스템 비교

표 4 및 67 페이지의 표 5에는 각 파일 시스템의 피처 및 제한사항이 요약되어 있습니다.

표 4. 파일 시스템 요약(1/2)

성능	/(“루트”)	QOpenSys	QSYS.LIB [16]	QDLS	QNTC
OS/400의 기본 부분	예	예	예	예	예
파일 유형	스트림	스트림	레코드 [12]	스트림	스트림
OfficeVision과의 통합(예. 파일을 메일로 전송)	아니오	아니오	아니오	예	아니오
OS/400 파일 서버를 통한 액세스	예	예	예	예	예
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	255자	10.6자 [6]	8.3자 [7]	255자
경로명 최대 길이 [8]	16MB	16MB	55 - 66 char[4]	82자	255자
오브젝트에 대한 확장 속성의 최대 길이	2GB	2GB	가변 [9]	32KB	64KB
파일 시스템 내 디렉토리 계층의 최대 레벨	제한 없음 [10]	제한 없음 [10]	3	32	127
오브젝트 당 최대 링크 [11]	Varies [15]	Varies [15]	1	1	1
기호 링크 지원	예	예	아니오	아니오	아니오
오브젝트/파일의 소유자 여부	예	예	예	예	아니오

표 4. 파일 시스템 요약(1/2) (계속)

성능	/("루트")	QOpenSys	QSYS.LIB [16]	QDLS	QNTC
통합 파일 시스템 명령 지원	예	예	예	예	예
통합 파일 시스템 API 지원	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

주:

1. 파일 서버 I/O 프로세서는 LAN 서버에 의해 사용되는 하드웨어입니다.
2. OS/400 파일 서버로 액세스할 때
3. LAN 서버 클라이언트 PC를 통하여 액세스할 때. iSeries API를 사용한 액세스는 상대적으로 느립니다
4. QSYS.LIB 파일 시스템의 최대 경로명 길이는 55자입니다. 자세한 내용은 76 페이지의 『라이브러리 파일 시스템(QSYS.LIB)』을 참조하십시오. 독립 ASP QSYS.LIB 파일 시스템의 최대 경로 길이는 66자입니다. 자세한 내용은 79 페이지의 『독립 ASP QSYS.LIB』을 참조하십시오.
5. 자세한 내용은 82 페이지의 『문서 라이브러리 서비스 파일 시스템(QDLS)』을 참조하십시오.
6. 오브젝트명에는 최대 10자, 오브젝트 유형에는 최대 6자가 허용됩니다. 보다 자세한 내용은 76 페이지의 『라이브러리 파일 시스템(QSYS.LIB)』을 참조하십시오.
7. 파일명에는 최고 8자, 파일 유형 확장자에는 최대 1-3자가 허용됩니다. 자세한 내용은 82 페이지의 『문서 라이브러리 서비스 파일 시스템(QDLS)』을 참조하십시오.
8. / 다음에 파일 시스템명으로 시작하는 절대 경로명(예: `/QDLS...`)을 가정합니다.
9. QSYS.LIB 및 독립 ASP QSYS.LIB 파일 시스템은 세 가지 사전 정의된 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 입력과 출력을 지원합니다.
13. 통합 파일 시스템 API는 스레드세이프 파일 시스템에 있는 오브젝트로 조작이 지시되는 경우 스레드세이프입니다. 다중 스레드가 작업에서 실행되고 있을 때 API가 스레드세이프이 아닌 파일 시스템의 오브젝트에 대하여 조작하는 경우, API는 실패합니다.
14. QSYS.LIB 및 독립 ASP QSYS.LIB 파일 시스템은 루트, UDFS, QOpenSys 파일 시스템 이외의 다른 오브젝트 유형 저널링을 지원합니다. QSYS.LIB 또는 독립 ASP QSYS.LIB 파일 시스템에 상주하는 오브젝트 저널링에 대한 자세한 정보는 iSeries Information Center의 저널 관리 주제를 참조하십시오.
15. *TYPE2 디렉토리는 오브젝트당 백만 개의 링크만 허용합니다. *TYPE1 디렉토리는 오브젝트당 32, 767개의 링크만 허용합니다. 자세한 정보는 *TYPE2 디렉토리를 참조하십시오.
16. 이 열의 자료는 QSYS.LIB 파일 시스템 및 독립 ASP QSYS.LIB 파일 시스템을 말합니다.

약어

char = 문자

B = 바이트 KB = 킬로바이트 MB = 메가바이트 GB = 기가바이트

표 5. 파일 시스템 요약(2/2)

성능	QOPT	QFileSvr.400	UDFS	NFS	QNetWare
OS/400의 기본 부분	예	예	예	예	아니오
파일 유형	스트림	스트림	스트림	스트림	스트림
OfficeVision과의 통합(예. 파일을 메일로 전송)	아니오	아니오	아니오	아니오	아니오
OS/400 파일 서버를 통한 액세스	예	예	예	예	예
통합 PC 서버(FSIOP)를 통한 직접 액세스 [1]	아니오	아니오	아니오	아니오	예
열기/닫기의 비교 속도	저속	저속 [2]	보통 [2]	보통 [2]	고속 [11]
대소문자로 구분되는 탐색	아니오	아니오 [2]	예 [12]	가변 [2]	아니오
경로명에서 각 구성요소의 최대 길이	가변 [4]	가변 [2]	255자	가변 [2]	255자 [13]
경로명의 최대 길이	294자	제한 없음 [2]	16MB	제한 없음 [2]	255자
오브젝트에 대한 확장 속성의 최대 길이	8MB	0 [6]	2GB [10]	0 [6]	64KB
파일 시스템 내 디렉토리 계층의 최대 레벨	제한 없음 [7]	제한 없음 [2]	제한 없음 [7]	제한 없음 [2]	100
오브젝트 당 최대 링크 [7]	1	1	가변 [15]	가변 [2]	1
기호 링크 지원	아니오	아니오	예	예 [2]	아니오
오브젝트/파일의 소유자 여부	아니오	아니오 [9]	예	예 [2]	예
통합 파일 시스템 명령 지원	예	예	예	예	예
통합 파일 시스템 API 지원	예	예	예	예	예
계층 파일 시스템(HFS) API 지원	예	아니오	아니오	아니오 [2]	아니오
스레드세이프(Threadsafe) [14]	예	아니오	예	아니오	아니오
오브젝트 저널링 지원	아니오	아니오	예	아니오	아니오

표 5. 파일 시스템 요약(2/2) (계속)

성능	QOPT	QFileSvr.400	UDFS	NFS	QNetWare

주:

1. 파일 서버 I/O 프로세서는 LAN 서버에 의해 사용되는 하드웨어입니다.
2. 액세스되는 리모트 파일 시스템에 따라 다릅니다.
3. OS/400 파일 서버를 통해 액세스될 때
4. 자세한 내용은 85 페이지의 『광 파일 시스템(QOPT)』을 참조하십시오.
5. / 다음에 파일 시스템명으로 시작하는 절대 경로명을 가정합니다.
6. QFileSvr.400 파일 시스템은 액세스중인 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. Novell NetWare 클라이언트 PC를 통해 액세스될 때. iSeries API를 사용한 액세스는 상대적으로 느립니다
12. UDFS를 작성할 때, 대소문자 구분이 지정될 수 있습니다. UDFS를 작성할 때 *MIXED 매개변수가 사용되면, 대소문자 탐색이 허용됩니다.
13. NetWare 디렉토리 서비스 오브젝트는 최대 255자입니다. 파일과 디렉토리는 DOS 8.3 형식으로 제한됩니다.
14. 통합 파일 시스템 API가 다중 스레드 프로세스에서 액세스될 때 이들은 스레드세이프 방식으로 처리됩니다. 파일 시스템은 스레드세이프가 아닌 파일 시스템에 대해 액세스를 허용하지 않습니다.
15. *TYPE2 디렉토리는 오브젝트당 백만 개의 링크만 허용합니다. *TYPE1 디렉토리는 오브젝트당 32, 767개의 링크만 허용합니다. 자세한 정보는 *TYPE2 디렉토리를 참조하십시오.

약어

char = 문자

B = 바이트 KB = 킬로바이트 MB = 메가바이트 GB = 기가바이트

"루트"(/) 파일 시스템

"루트"(/) 파일 시스템은 통합 파일 시스템의 스트림 파일 지원 및 계층적 디렉토리 구조의 장점을 갖습니다. "루트"(/) 파일 시스템은 DOS 및 OS/2 파일 시스템의 특징을 갖습니다.

또한,

- 스트림 파일 입출력(I/O)에 대해 최적화되었습니다.
- 복수 하드 링크 및 기호 링크를 지원합니다.
- 로컬 소켓을 지원합니다.
- *OOPOOL 오브젝트를 지원합니다.
- 스레드세이프 어플리케이션 프로그램 인터페이스를 지원합니다.
- *FIFO 오브젝트를 지원합니다.
- 기타 *CHRSF 오브젝트 및 /dev/null, /dev/zero *CHRSF 오브젝트를 지원합니다.

| • 오브젝트 변경사항의 저널링을 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 있으 | 며, /dev/null 또는 /dev/zero에 기록된 다른 데이터는 삭제됩니다. /dev/null 및 /dev/zero 파일의 오 | 브젝트 유형은 *CHRSF이며 일반 파일처럼 사용될 수 있지만, /dev/null 파일에서 자료를 읽을 수 없으며 | /dev/zero 파일은 0으로 지워진 자료를 항상 리턴합니다.

"루트"(/) 파일 시스템에 대한 자세한 정보는 "루트"(/) 파일 시스템 사용을 참조하십시오.

"루트"(/) 파일 시스템 사용

『루트』(/) 파일 시스템은 OS/400 파일 서버 또는 통합 파일 시스템 명령, 사용자 화면, C 언어 API를 사용하여 통합 파일 시스템 인터페이스로 액세스할 수 있습니다.

"루트"(/) 파일 시스템에서 대소문자 구분

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

"루트"(/) 파일 시스템의 경로명

- 경로명은 다음과 같은 형식으로 되어 있습니다.

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

- 공유명 뒤의 경로명의 구성요소는 Windows NT 경로명 규칙을 따릅니다. 조합된 디렉토리명과 오브젝트명이 255자의 총 경로 길이를 채울 수 있습니다.
- 프로그램 및 서버 공간 제한을 제외하고 디렉토리 계층 깊이에는 제한이 없습니다.
| • 이름 저장시 문자는 UCS2 레벨 1 형식(*TYPE1 디렉토리) 및 UTF-16(*TYPE2 디렉토리)으로 변환됩니 | 다(24 페이지의 『이름 연속성』 참조). 디렉토리 형식에 대한 자세한 정보는 *TYPE2 디렉토리를 참조하십 | 시오.

"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 시스템의 오브젝트와의 링크에 사용됩니다.

링크 설명에 대해서는 19 페이지의 『링크』를 참조하십시오

"루트"(/) 파일 시스템에서 통합 파일 시스템 명령 사용

28 페이지의 『CL 명령을 사용한 조작 수행』에 나열된 모든 명령어와 27 페이지의 『iSeries 메뉴 및 표시장치를 사용하여 조작 수행』에 설명된 표시장치는 『루트』(/) 파일 시스템상에서 작동할 수 있습니다. 그러나 다중 스레드가 가능한 프로세스에서 이 명령을 사용하는 것은 안전하지 않을 수도 있습니다.

″루트″(/) 파일 시스템에서 통합 파일 시스템 API 사용

54 페이지의 『API를 사용한 조작 수행』에 나열된 모든 C 언어 API는 스레드세이프 방식으로 『루트』(/) 파일 시스템에서 작동할 수 있습니다.

″루트″(/) 파일 시스템에서 오브젝트 변경사항 저널

″루트″(/) 파일 시스템의 오브젝트는 저널됩니다. 저널 관리의 기본 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 회복할 수 있도록 하는 것입니다. ″루트″(/) 파일 시스템에서 오브젝트 변경사항 저널링에 대한 자세한 정보는 101 페이지의 제 7 장 『통합 파일 시스템 오브젝트에 대한 저널링 지원』을 참조하십시오.

개방 시스템 파일 시스템(QOpenSys)

QOpenSys 파일 시스템은 POSIX와 XPG와 같은 UNIX용 개방 시스템 표준과 호환될 수 있습니다. ″루트″(/) 파일 시스템과 마찬가지로, 이 파일 시스템은 통합 파일 시스템이 제공하는 스트림 파일 및 디렉토리 지원의 장점을 갖습니다.

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.
- *OOPOOL 오브젝트를 지원합니다.
- 스레드세이프 API를 지원합니다.
- *FIFO 오브젝트를 지원합니다.
- 오브젝트 변경사항의 저널링을 지원합니다.

QOpenSys 파일 시스템은 UNIX용 개방 시스템 표준 지원이 가능한 대소문자 구분 외에 『루트』(/) 파일 시스템과 같은 특성을 지닙니다.

QOpenSys에 대한 자세한 정보는 QOpenSys 사용을 참조하십시오.

*TYPE 1에서 *TYPE2 디렉토리로의 변환 및 QOpenSys 파일 시스템 제한사항에 대한 정보는 *TYPE2 디렉토리를 참조하십시오.

QOpenSys 사용

QOpenSys는 OS/400 파일 서버 또는 통합 파일 시스템 명령어, 사용자 화면, C 언어 API를 사용하여 통합 파일 시스템 인터페이스로 액세스할 수 있습니다.

QOpenSys 파일 시스템에서 대소문자 구분

『루트』(/) 파일 시스템과 달리 QOpenSys 파일 시스템은 오브젝트명 탐색시 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`, `PAYROLL` 오브젝트를 모두 사용할 수 있습니다.

QOpenSys 파일 시스템에서 경로명

- 경로명은 다음과 같은 형식으로 되어 있습니다.

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

- 경로명의 각각 구성요소는 최대 255자가 될 수 있습니다. 완전한 경로명은 최대 16MB가 될 수 있습니다.
- 프로그램 및 서버 공간 제한을 제외하고 디렉토리 계층 깊이에는 제한이 없습니다.
- 이름 저장시 문자는 UCS2 레벨 1 형식(*TYPE1 디렉토리) 및 UTF-16(*TYPE2 디렉토리)으로 변환됩니다(24 페이지의 『이름 연속성』 참조). 디렉토리 형식에 대한 자세한 정보는 *TYPE2 디렉토리를 참조하십시오.

QOpenSys 파일 시스템에서 링크

QOpenSys 파일 시스템 내에서 동일한 오브젝트에 여러 개의 하드 링크가 허용됩니다. 기호 링크가 전면 지원됩니다. 기호 링크는 QOpenSys 파일 시스템에서 다른 파일 시스템의 오브젝트와의 링크에 사용됩니다.

링크 설명에 대해서는 19 페이지의 『링크』를 참조하십시오

QOpenSys 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

28 페이지의 『CL 명령을 사용한 조작 수행』에 나열된 모든 명령과 27 페이지의 『iSeries 메뉴 및 표시장치를 사용하여 조작 수행』에서 설명된 화면들은 QOpenSys 파일 시스템에서 사용할 수 있습니다. 그러나 다중 스레드가 가능한 프로세스에서 이 명령을 사용하는 것은 안전하지 않을 수도 있습니다.

QOpenSys 파일 시스템에서 통합 파일 시스템 API 사용

54 페이지의 『API를 사용한 조작 수행』에 나열된 모든 C 언어 함수는 스레드세이프 방식으로 QOpenSys 파일 시스템상에서 사용할 수 있습니다.

QOpenSys 파일 시스템에서 오브젝트 변경사항 저널

QOpenSys 파일 시스템의 오브젝트는 저널됩니다. 저널 관리의 기본 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 회복할 수 있도록 하는 것입니다. QOpenSys 파일 시스템에서 오브젝트 변경사항 저널링에 대한 자세한 정보는 101 페이지의 제 7 장 『통합 파일 시스템 오브젝트에 대한 저널링 지원』을 참조하십시오.

사용자 정의 파일 시스템(UDFS)

UDFS 파일 시스템은 사용자가 선택한 ASP(Auxiliary Storage Pool)나 독립 ASP(Auxiliary Storage Pool)에 상주합니다. 사용자가 이들 파일 시스템을 작성하고 관리합니다.

또한,

- DOS 및 OS/2와 같은 PC 오퍼레이팅 시스템과 유사한 계층적 디렉토리 구조를 제공합니다.
- 스트림 파일 입출력(I/O)에 대해 최적화되었습니다.
- 복수 하드 링크 및 기호 링크를 지원합니다.
- 로컬 소켓을 지원합니다.
- 스레드세이프 API를 지원합니다.
- *FIFO 오브젝트를 지원합니다.
- 오브젝트 변경사항의 저널링을 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

사용자 정의 파일 시스템에 대한 자세한 정보는 다음 주제를 참조하십시오.

- UDFS 개념
- 통합 파일 시스템 인터페이스를 통한 UDFS 사용

UDFS 개념

UDFS에서, 『루트』(/) 및 QOpenSys 파일 시스템에서와 같이 디렉토리, 스트림 파일, 기호 링크, 로컬 소켓 및 SOM 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, QFileSvr.4001 인터페이스를 통해서만 사용자에 액세스될 수 있습니다.

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 디렉토리의 원래 내용은 액세스가 불가능합니다. 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 해제시키고 난 후, /home/JON/payroll 파일의 액세스가 다시 가능해지며 UDFS에 있는 3개 디렉토리의 액세스가 불가능해집니다.

| **주:** 독립 ASP가 마운트될 수 없는 UDFS.

파일 시스템 마운트에 대해 자세히 알려면 OS/400 네트워크 파일 시스템 지원  을 참조하십시오.

통합 파일 시스템 인터페이스를 통한 UDFS 사용

OS/400 파일 서버나 통합 파일 시스템 명령, 사용자 표시장치 및 API를 사용하여 통합 파일 시스템 인터페이스를 통해 UDFS에 액세스할 수 있습니다. 통합 파일 시스템 인터페이스를 사용할 경우 다음 고려사항 및 제한사항을 알고 있어야 합니다.

통합 파일 시스템 UDFS에서 대소문자 구분

UDFS를 작성할 때 사용자는 UDFS에 있는 오브젝트 이름의 대소문자 구분 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. 대소문자를 구분하는 UDFS를 작성하기 위해서는 CRTUDFS 명령을 사용할 때 CASE 매개변수에 *MIXED를 지정할 수 있습니다.

대소문자 구분 안함을 선택하면 서버에서 이름을 탐색하는 동안 대소문자를 구분하지 않습니다. 따라서, 서버는 /home/CAYCE와 /HOME/cayce를 분리된 디렉토리가 아닌 동일한 디렉토리로 인식합니다. 대소문자를 무시하는 UDFS를 작성하기 위해서는 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 서버를 통해 이름을 탐색하는 방법에만 적용됩니다.

통합 파일 시스템 UDFS에서 경로명

| 블록 특수 파일(*BLKSF)은 전체 UDFS와 UDFS 내의 모든 오브젝트를 조작해야 할 때의 UDFS를 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 UDFS
| 의 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. 완전한 경로명은 최대 16MB가 될 수 있습니다.
| • 프로그램 및 서버 공간 제한을 제외하고 디렉토리 계층 깊이에는 제한이 없습니다.
| • 이름 저장시 문자는 UCS2 레벨 1 형식(*TYPE1 디렉토리) 및 UTF-16(*TYPE2 디렉토리)으로 변환됩니
| 다(24 페이지의 『이름 연속성』 참조). 디렉토리 형식에 대한 자세한 정보는 *TYPE2 디렉토리를 참조하십
| 시오.

통합 파일 시스템 UDFS에서 링크

UDFS에 있는 오브젝트는 다수의 하드 링크가 동일한 오브젝트에 액세스할 수 있도록 하며, 기호 링크를 완전하게 지원합니다. 기호 링크는 UDFS로부터 다른 파일 시스템의 오브젝트로 링크를 작성할 수 있습니다.

링크 설명에 대해서는 19 페이지의 『링크』를 참조하십시오.

UDFS에서 통합 파일 시스템 명령 사용

28 페이지의 『CL 명령을 사용한 조작 수행』에 나열된 모든 명령어들과 27 페이지의 『iSeries 메뉴 및 표시장치를 사용하여 조작 수행』에 표시된 모든 화면들은 사용자 정의 파일 시스템에서 조작할 수 있습니다. 사용자 정의 파일 시스템과 일반적으로 마운트되어 있는 기타 파일 시스템에 고유한 몇 가지 CL 명령이 있습니다. 다음은 이에 관한 설명입니다.

표 6. 사용자 정의 파일 시스템 CL 명령

명령	설명
ADDMFS	마운트된 파일 시스템 추가는 내보낸 리모트 서버 파일 시스템을 로컬 클라이언트 디렉토리에 위치시킵니다.
CRTUDFS	UDFS 작성은 사용자 정의 파일 시스템을 작성합니다.
DLTUDFS	UDFS 삭제는 사용자 정의 파일 시스템을 삭제합니다.
DSPMFSINF	마운트된 파일 시스템 정보 표시는 마운트 파일 시스템에 대한 정보를 표시합니다.
DSPUDFS	사용자 정의 파일 시스템에 대한 UDFS 표시 화면 정보를 표시합니다.
MOUNT	파일 시스템 마운트는 내보낸 리모트 서버 파일 시스템을 로컬 클라이언트 디렉토리상에 위치시킵니다. 이 명령은 ADDMFS 명령에 대한 별명입니다.
RMVMFS	마운트된 파일 시스템 제거는 로컬 클라이언트 이름공간에서 내보낸 리모트 서버 파일 시스템을 제거합니다.
UNMOUNT	파일 시스템 마운트 해제는 내보낸 리모트 서버 파일 시스템을 로컬 클라이언트 이름공간으로부터 제거합니다. 이 명령은 RMVMFS 명령에 대한 별명입니다.

주: UDFS에 저장된 오브젝트에서 통합 파일 시스템 명령을 사용하기 전에 반드시 UDFS를 마운트시켜야 합니다.

UDFS에서 통합 파일 시스템 API 사용

54 페이지의 『API를 사용한 조작 수행』에 나열된 모든 C 언어는 사용자 정의 파일 시스템에서 조작됩니다

주: UDFS에 저장된 오브젝트에서 통합 파일 시스템 명령을 사용하기 전에 반드시 UDFS를 마운트시켜야 합니다.

UDFS에 대한 그래픽 사용자 인터페이스

iSeries Navigator에서, 사용자 PC상의 그래픽 사용자 인터페이스는 UDFS에 대한 쉽고도 편리한 액세스를 제공합니다. 이 인터페이스를 사용하여 Windows 클라이언트로부터 UDFS를 작성, 삭제, 표시, 마운트 및 마운트 해제할 수 있습니다.

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

- 46 페이지의 『신규 사용자 정의 파일 시스템 작성』
- 46 페이지의 『사용자 정의 파일 시스템 마운트』
- 47 페이지의 『사용자 정의 파일 시스템 마운트 해제』

통합 파일 시스템 UDFS 작성

CRTUDFS(사용자 정의 파일 시스템 작성) 명령은 통합 파일 시스템 이름공간, API 및 CL 명령을 통해 볼 수 있는 파일 시스템을 작성합니다. ADDMFS나 MOUNT 명령은 기존 로컬 디렉토리의 『맨 위에』 UDFS를 위치시킵니다. 사용자가 선택한 ASP나 독립 ASP에서 UDFS를 작성할 수 있습니다. 또한 대소문자 구분을 지정할 수 있습니다.

통합 파일 시스템 UDFS 삭제

DLTUDFS(사용자 정의 파일 시스템 삭제) 명령은 기존의 마운트 해제된 UDFS와 UDFS 내의 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.

통합 파일 시스템 UDFS 표시

DSPUDFS(사용자 정의 파일 시스템 표시) 명령은 기존 UDFS의 속성(마운트 또는 마운트 해제 여부)을 나타냅니다. DSPMFSINF(마운트된 파일 시스템 정보 표시) 명령은 마운트된 파일 시스템 뿐만 아니라, 마운트된 UDFS에 관한 정보도 표시합니다.

통합 파일 시스템 UDFS 마운트

ADDMFS(마운트된 파일 시스템 추가) 명령과 MOUNT 명령은 파일 시스템 내의 오브젝트가 통합 파일 시스템 이름공간에 액세스할 수 있도록 합니다. UDFS를 마운트하기 위해서 ADDMFS의 TYPE 매개변수에 *UDFS를 지정해야 합니다.

주: 독립 ASP가 마운트될 수 없는 UDFS.

통합 파일 시스템 UDFS 마운트 해제

마운트 해제 명령은 UDFS의 내용을 통합 파일 시스템 인터페이스로 액세스할 수 없도록 합니다. UDFS안에 있는 오브젝트는 UDFS가 일단 마운트 해제되면, 개별적으로 액세스가 불가능합니다. RMVMFS(마운트된 파일 시스템 제거) 명령 또는 UNMOUNT 명령은 마운트된 파일 시스템이 통합 파일 시스템 이름공간으로 액세스하지 못하게 합니다. 명령을 사용할 때 파일 시스템의 임의의 오브젝트가 사용 중이면(예를 들어, 파일이 열려있는 경우), 오류 메세지를 수신합니다. 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.udfs를 통합 파일 시스템 이름공간의 /home/judy에 마운트합니다. 그런 다음, 다른 파일 시스템 /pubs를 /home/judy에 마운트하면 jenn.udfs의 내용에 액세스할 수 없게 됩니다. 또한, /home/judy로부터 두 번째 파일 시스템을 제거할 때까지는 jenn.udfs를 마운트 해제할 수 없습니다.

주: 독립 ASP가 마운트될 수 없는 UDFS.

통합 파일 시스템 UDFS 저장 및 복원

모든 UDFS의 연관된 권한뿐 아니라 모든 UDFS 오브젝트를 저장 및 복원할 수 있습니다. 저장 명령(SAV)을 사용하여 UDFS에 오브젝트를 저장할 수 있으며, 복원 명령(RST)을 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.

UDFS 파일 시스템에서 오브젝트 변경사항 저널

사용자 정의 파일 시스템에서 오브젝트는 저널됩니다. 저널 관리의 기본 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 회복할 수 있도록 하는 것입니다. UDFS 파일 시스템에서 오브젝트 변경사항 저널에 대한 자세한 정보는 101 페이지의 제 7 장 『통합 파일 시스템 오브젝트에 대한 저널링 지원』을 참조하십시오.

라이브러리 파일 시스템(QSYS.LIB)

QSYS.LIB 파일 시스템은 iSeries 서버 라이브러리 구조를 지원합니다. 이 파일 시스템은 시스템 및 기본 사용자 ASP에서 데이터베이스 파일과 라이브러리 지원이 관리하는 다른 모든 iSeries 서버 오브젝트 유형에 대한 액세스를 제공합니다.

또한,

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

OS/400 버전 3 이전에, QSYS.LIB 파일 시스템은 iSeries 서버 파일 시스템으로 알려져 있습니다. 어플리케이션 개발을 위해 RPG 또는 COBOL과 같은 언어와 DDS와 같은 기능을 사용한 프로그래머는 QSYS.LIB

파일 시스템을 사용했던 것입니다. 출력 대기행렬을 조작하기 위해 명령어, 메뉴 및 화면을 사용한 시스템 오퍼레이터는 사용자 프로파일을 작성 및 변경한 시스템 관리자와 같이 QSYS.LIB 파일 시스템을 사용했던 것입니다.

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

QSYS.LIB에 대한 자세한 정보는 통합 파일 시스템 인터페이스를 통한 QSYS.LIB 사용을 참조하십시오.

통합 파일 시스템 인터페이스를 통한 QSYS.LIB 사용

QSYS.LIB 파일 시스템은 OS/400 파일 서버 또는 통합 파일 시스템 명령, 사용자 화면 및 C 언어 API를 사용하여 통합 파일 시스템 인터페이스를 통해 액세스할 수 있습니다. 통합 파일 시스템 인터페이스 사용에서는 다음 고려사항 및 제한사항을 참조하십시오.

QSYS.LIB 파일 시스템에서 QPWFSERVER 권한 부여 리스트

QPWFSERVER는 리모트 클라이언트를 통해 액세스되는 QSYS.LIB 파일 시스템의 모든 오브젝트에 대해 추가 액세스 요구를 제공하는 권한 부여 리스트입니다. 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)이나 WRKAUTL(권한 부여 리스트에 대한 작업) 명령을 사용하여 권한의 값을 변경할 수 있습니다. 관리자는 권한 부여 리스트에 PUBLIC *EXCLUDE 권한을 할당하여, 일반 사용자들이 리모트 클라이언트에서 QSYS.LIB 오브젝트에 액세스할 수 없게 만듭니다.

QSYS.LIB 파일 시스템에서 파일 처리 제한사항

- 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 iSeries Information Center의 fcntl() 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 I/O 조작을 지원합니다. 예를 들어, 프로그램은 스트림 자료를 사용자 공간에 쓰고 사용자 공간으로부터 자료를 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 파일에 대한 지원

QSYS.LIB 파일 시스템은 저장 파일 오브젝트에 스트림 I/O 조작을 지원합니다. 예를 들면, 기존의 다른 빈 저장 파일 오브젝트에 자료를 배치할 필요가 있을 때까지 기존 저장 파일에는 다른 파일로 복사되거나 읽을 수 있는 자료가 있습니다. 쓰기에 대해 저장 파일이 열려 있을 때, 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 자료에 스트림 I/O 조작은 작업에서 다중 스레드가 실행 중인 경우 허용되지 않습니다.

저장 파일에 대한 스트림 I/O 조작은 저장 파일이나 디렉토리가 네트워크 파일 시스템 서버를 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 오브젝트명의 대소문자를 구분하지 않습니다. 오브젝트명에 대한 탐색 시 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 오브젝트 유형에 따라 2 내지 3 레벨(경로명의 경우 2 내지 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"(/)와 QSYS.LIB가 처음 두 레벨로 포함된 경우, QSYS.LIB의 디렉토리 계층을 최대 5 레벨까지 만들 수 있습니다.
- 이름 저장시 이름의 문자들은 CCSID 37로 변환됩니다. 그러나 인용된 이름은 작업의 CCSID를 사용하여 저장됩니다.

  CCSID에 대한 자세한 정보는 iSeries Information Center의 글로벌화 주제를 참조하십시오.

QSYS.LIB 파일 시스템에서 링크

기호 링크는 QSYS.LIB 파일 시스템에서 작성 또는 저장될 수 없습니다.

라이브러리와 라이브러리 내 오브젝트간의 관계는 라이브러리와 라이브러리 내의 각 오브젝트 사이의 하드 링크와 같습니다. 통합 파일 시스템은 하나의 링크로서 라이브러리 오브젝트 관계를 핸들합니다. 따라서, 기호 링크를 지원하는 파일 시스템에서 QSYS.LIB 파일 시스템의 오브젝트와의 링크가 가능합니다.

링크 설명에 대해서는 19 페이지의 『링크』를 참조하십시오

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

다음 명령을 제외하고 28 페이지의 『CL 명령을 사용한 조작 수행』에 나열된 명령어는 QSYS.LIB 파일 시스템에서 사용할 수 있습니다.

- QSYS.LIB의 오브젝트*에* 기호 링크를 작성하는 경우에만 ADDLNK 명령을 사용할 수 있습니다.
- 파일 조작은 프로그램 설명 실제 파일 및 소스 실제 파일에서만 수행될 수 있습니다.
- STRJRN 및 ENDJRN 명령은 데이터베이스 실제 파일에서 사용할 수 없습니다.

동일한 제한사항이 27 페이지의 『iSeries 메뉴 및 표시장치를 사용하여 조작 수행』에 설명된 사용자 화면에 적용됩니다.

QSYS.LIB 파일 시스템에서 통합 파일 시스템 API 사용

다음을 제외하고 54 페이지의 『API를 사용한 조작 수행』에 나열된 C 언어 함수는 QSYS.LIB 파일 시스템상에서 작동됩니다.

- 파일 조작은 프로그램 설명 실제 파일 및 소스 실제 파일에서만 수행될 수 있습니다.
- symlink() 함수는 기호 링크를 지원하는 다음 파일 시스템에서 QSYS.LIB에 있는 오브젝트와의 링크에만 사용됩니다.
- QjoStartJournal() 및 QjoEndJournal() API는 데이터베이스 실제 파일에서 사용할 수 없습니다.

독립 ASP QSYS.LIB

독립 ASP QSYS.LIB 파일 시스템은 사용자가 작성하고 정의한 독립 ASP(Auxiliary Storage Pool)의 iSeries 서버 라이브러리 구조를 지원합니다. 이 파일 시스템은 독립 ASP에서 데이터베이스 파일과 라이브러리 지원이 관리하는 다른 모든 iSeries 서버 오브젝트 유형에 대한 액세스를 제공합니다.

또한,

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

독립 ASP QSYS.LIB 파일 시스템에 대한 자세한 정보는 통합 파일 시스템 인터페이스를 통한 독립 ASP QSYS.LIB 사용을 참조하십시오.

통합 파일 시스템 인터페이스를 통한 독립 ASP QSYS.LIB 사용

독립 ASP QSYS.LIB 파일 시스템은 OS/400 파일 서버 또는 통합 파일 시스템 명령, 사용자 화면 및 C 언어 API를 사용하여 통합 파일 시스템 인터페이스를 통해 액세스할 수 있습니다. 통합 파일 시스템 인터페이스 사용에서는 다음 고려사항 및 제한사항을 참조하십시오.

독립 ASP QSYS.LIB 파일 시스템에서 QPWFSERVER 권한 부여 리스트

QPWFSERVER는 리모트 클라이언트를 통해 액세스되는 독립 ASP QSYS.LIB 파일 시스템의 모든 오브젝트에 대해 추가 액세스 요구를 제공하는 권한 부여 리스트입니다. 이 권한 부여 리스트에 지정된 권한들은 독립 ASP QSYS.LIB 파일 시스템 내의 모든 오브젝트에 적용됩니다.

이 오브젝트에 대한 디폴트 권한은 PUBLIC *USE 권한입니다. 관리자는 EDTAUTL(권한 부여 리스트 편집)이나 WRKAUTL(권한 부여 리스트에 대한 작업) 명령을 사용하여 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 만듭니다.

독립 ASP QSYS.LIB 파일 시스템에서 파일 처리 제한사항

- 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 iSeries Information Center의 fcntl() 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 I/O 조작을 지원합니다. 예를 들어, 프로그램은 스트림 자료를 사용자 공간에 쓰고 사용자 공간으로부터 자료를 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.

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 자료를 배치할 필요가 있을 때까지 기존 저장 파일에는 다른 파일로 복사되거나 읽을 수 있는 자료가 있습니다. 쓰기에 대해 저장 파일이 열려 있을 때, 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 자료에 스트림 I/O 조작은 작업에서 다중 스레드가 실행 중인 경우 허용되지 않습니다.

저장 파일에 대한 스트림 I/O 조작은 저장 파일이나 디렉토리가 네트워크 파일 시스템 서버를 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

일반적으로, 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 오브젝트 유형에 따라 2 내지 3 레벨(경로명의 경우 2 내지 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 계층을 최대 6 레벨까지 만들 수 있습니다.
- 이름 저장시 이름의 문자들은 CCSID 37로 변환됩니다. 그러나 인용된 이름은 작업의 CCSID를 사용하여 저장됩니다.

  CCSID에 대한 자세한 정보는 iSeries Information Center의 글로벌화 주제를 참조하십시오.

독립 ASP QSYS.LIB 파일 시스템에서 링크

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

라이브러리와 라이브러리 내 오브젝트간의 관계는 라이브러리와 라이브러리 내의 각 오브젝트 사이의 하드 링크와 같습니다. 통합 파일 시스템은 하나의 링크로서 라이브러리 오브젝트 관계를 핸들합니다. 따라서, 기호 링크를 지원하는 파일 시스템에서 독립 ASP QSYS.LIB 파일 시스템의 오브젝트와의 링크가 가능합니다.

링크 설명에 대해서는 19 페이지의 『링크』를 참조하십시오.

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

다음 명령을 제외하고 28 페이지의 『CL 명령을 사용한 조작 수행』에 나열된 명령어는 독립 ASP QSYS.LIB 파일 시스템에서 작동됩니다.

- 독립 ASP QSYS.LIB의 오브젝트*에* 기호 링크를 작성하는 경우에만 ADDLNK 명령을 사용할 수 있습니다.
- 파일 조작은 프로그램 설명 실제 파일 및 소스 실제 파일에서만 수행될 수 있습니다.
- STRJRN 및 ENDJRN 명령은 데이터베이스 실제 파일에서 사용할 수 없습니다.
- MOV 명령을 사용하여 독립 ASP QSYS.LIB 파일 시스템의 라이브러리를 기본 ASP(Auxiliary Storage Pool)로 이동할 수 없습니다. 그러나 독립 ASP QSYS.LIB의 라이브러리를 시스템 ASP나 다른 독립 ASP로 이동할 수는 있습니다.
- SAV나 RST를 사용하여 독립 ASP의 라이브러리 오브젝트를 저장하거나 복원하려면 해당 독립 ASP를 SAV나 RST를 수행하는 작업에 연관시키거나 `ASPDEV` 매개변수에 독립 ASP를 지정해야 합니다. `/asp_name/QSYS.LIB/object.type`의 경로명 명명 규칙은 SAV와 RST에서 지원되지 않습니다.

동일한 제한사항이 27 페이지의 『iSeries 메뉴 및 표시장치를 사용하여 조작 수행』에 설명된 사용자 화면에 적용됩니다.

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

다음을 제외하고 54 페이지의 『API를 사용한 조작 수행』에 나열된 C 언어 함수는 독립 ASP QSYS.LIB 파일 시스템상에서 작동됩니다.

- 파일 조작은 프로그램 설명 실제 파일 및 소스 실제 파일에서만 수행될 수 있습니다.
- symlink() 함수는 기호 링크를 지원하는 다음 파일 시스템에서 독립 ASP QSYS.LIB에 있는 오브젝트와의 링크에만 사용됩니다.
- QjoStartJournal() 및 QjoEndJournal() API는 데이터베이스 실제 파일에서 사용할 수 없습니다.

문서 라이브러리 서비스 파일 시스템(QDLS)

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

또한,

- iSeries 서버 폴더 및 문서 라이브러리 오브젝트(DLO)를 지원합니다.
- 스트림 파일에 저장된 자료를 지원합니다.

QDLS에 대한 자세한 정보는 통합 파일 시스템 인터페이스를 통한 QDLS 사용을 참조하십시오.

통합 파일 시스템 인터페이스를 통한 QDLS 사용

QDLS 파일 시스템은 OS/400 파일 서버 또는 통합 파일 시스템 명령, 사용자 화면 및 C 언어 API를 사용하여 통합 파일 시스템 인터페이스를 통해 액세스할 수 있습니다. 통합 파일 시스템 인터페이스 사용에서는 다음 고려사항 및 제한사항을 참조하십시오.

QDLS 파일 시스템에서 통합 파일 시스템 및 HFS

문서 라이브러리 오브젝트(DLO) CL 명령 뿐만 아니라, HFS라는 계층 파일 시스템이 제공하는 API나 통합 파일 시스템 인터페이스를 통해 QDLS 파일 시스템의 오브젝트에 대한 조작을 수행할 수 있습니다. 통합 파일 시스템이 통합 언어 환경(ILE) 프로그램 모델을 기반으로 하는 반면, HFS는 원래 iSeries 서버 프로그램 모델을 기반으로 합니다.

HFS API를 이용하여 통합 파일 시스템이 지원하지 않는 몇 개의 추가 연산을 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 사용한 API의 명명 규칙과는 다르다는 점에 유의하십시오.

HFS에 대한 자세한 정보는 iSeries Information Center의 계층 파일 시스템 API 주제를 참조하십시오.

QDLS 파일 시스템에서 사용자 등록

QDLS에서 오브젝트에 대한 작업을 할 때 사용자는 시스템 분배 디렉토리에 등록되어야 합니다.

QDLS 파일 시스템에서 대소문자 구분

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

다른 모든 문자는 QDLS에서 대소문자로 구분됩니다.

자세한 내용은 iSeries Information Center의 폴더 및 문서명 주제를 참조하십시오.

QDLS 파일 시스템에서 경로명

- 경로명의 각 구성요소는 다음과 같이 이름만으로 구성될 수 있습니다.

  ```
  /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.
- QDLS 내의 디렉토리 계층은 최대 32 레벨이 될 수 있습니다. /와 QDLS가 처음의 두 레벨로 포함된 경우, 디렉토리 계층은 34 레벨이 될 수 있습니다.
- 이름 문자는 자료 영역 Q0DEC500이 작성되지 않는 한, 이름이 저장될 때 작업의 코드 페이지로 변환됩니다. 자료 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 iSeries Information Center의 글로벌화 주제를 참조하십시오.

QDLS 파일 시스템에서 링크

기호 링크는 QDLS 파일 시스템에서 작성 또는 저장될 수 없습니다.

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

링크 설명에 대해서는 19 페이지의 『링크』를 참조하십시오

QDLS 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

다음 명령을 제외하고, 28 페이지의 『CL 명령을 사용한 조작 수행』에 나열된 명령어는 QDLS 파일 시스템에서 사용할 수 있습니다.

- 기호 링크를 지원하는 다른 파일 시스템에서 QDLS의 오브젝트로 링크하기 위해서만 ADDLNK 명령을 사용할 수 있습니다.
- 파일에 대해서 CHKIN 및 CHKOUT 명령이 지원되지만 디렉토리에 대해서는 지원되지 않습니다.
- APYJRNCHG, ENDJRN, SNDJRNE 및 STRJRN 명령은 지원되지 않습니다.

동일한 제한사항이 27 페이지의 『iSeries 메뉴 및 표시장치를 사용하여 조작 수행』에 설명된 사용자 화면에 적용됩니다.

QDLS 파일 시스템에서 통합 파일 시스템 API 사용

다음을 제외하고 54 페이지의 『API를 사용한 조작 수행』에 나열된 C 언어 함수는 QDLS 파일 시스템상에서 작동됩니다.

- symlink() 함수는 기호 링크를 지원하는 다른 파일 시스템에서 QDLS에 있는 오브젝트와의 링크에만 사용됩니다.
- 다음 함수는 지원되지 않습니다.

  givedescriptor()

  ioctl()

  link()

  QjoEndJournal()

  QjoRetrieveJournalInformation()

  QJORJIDI()

  QJOSJRNE()

  QjoStartJournal()

  Qp0lGetPathFromFileID()

  readlink()

  takedescriptor()

광 파일 시스템(QOPT)

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

또한,

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

QOPT에 대한 자세한 정보는 통합 파일 시스템 인터페이스를 통한 QOPT 사용을 참조하십시오.

통합 파일 시스템 인터페이스를 통한 QOPT 사용

QOPT 파일 시스템은 OS/400 파일 서버 또는 통합 파일 시스템 명령, 사용자 화면 및 API를 사용하여 통합 파일 시스템 인터페이스를 통해 액세스될 수 있습니다. 통합 파일 시스템 인터페이스를 사용할 때, 다음의 고려사항 및 제한사항을 참조하십시오.

자세한 내용은 광 지원 서적을 참조하십시오.

QOPT 파일 시스템에서 통합 파일 시스템 및 HFS

HFS로 알려진 계층적 파일 시스템이 제공하는 API 또는 통합 파일 시스템 인터페이스를 통해 QOPT 파일 시스템의 오브젝트상에서 수행될 수 있습니다. 통합 파일 시스템이 통합 언어 환경(ILE) 프로그램 모델을 기반으로 하는 반면, HFS는 원래 iSeries 서버 프로그램 모델을 기반으로 합니다.

HFS API를 이용하여 통합 파일 시스템이 지원하지 않는 몇 개의 추가 연산을 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 사용한 API의 명명 규칙과는 다르다는 점에 유의하십시오.

HFS API에 대한 자세한 정보는 iSeries Information Center의 계층 파일 시스템 API 주제 또는

광 지원 서적을 참조하십시오.

QOPT 파일 시스템에서 대소문자 구분

QOPT에서 파일이나 디렉토리를 작성할 때 광 매체 형식에 따라 문자가 보존될 수도 있고 보존되지 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 디렉토리나 서브디렉토리를 경로명에 포함시킬 수 있습니다. 디렉토리 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 광 지원 서적의 『경로명 규칙』 설명을 참조하십시오.

QOPT 파일 시스템에서 링크

QOPT 파일 시스템은 오브젝트에 하나의 링크만을 지원합니다. QOPT에서 기호 링크는 작성되거나 저장될 수 없습니다. 그러나 QOPT의 파일은 『루트』(/) 또는 QOpenSys 파일 시스템에서 기호 링크를 사용하여 액세스할 수 있습니다.

링크 설명에 대해서는 19 페이지의 『링크』를 참조하십시오

QOPT 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

28 페이지의 『CL 명령을 사용한 조작 수행』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 27 페이지의 『iSeries 메뉴 및 표시장치를 사용하여 조작 수행』에 설명된 사용자 화면에 적용됩니다.

다음 통합 파일 시스템 명령은 QOPT 파일 시스템에서 지원되지 않습니다.

- ADDLNK
- APYJRNCHG
- CHKIN
- CHKOUT
- ENDJRN
- SNDJRNE
- STRJRN
- WRKOBJOWN
- WRKOBJPGP

QOPT 파일 시스템에서 통합 파일 시스템 API 사용

54 페이지의 『API를 사용한 조작 수행』에 나열된 모든 C 언어 API는 다음을 제외한 스레드세이프 방식으로 『루트』(/) 파일 시스템에서 작동할 수 있습니다.

- QjoEndJournal()
- QjoRetrieveJournalInformation()

| • QJORJIDI()
| • QJOSJRNE()
| • QjoStartJournal()

NetWare 파일 시스템(QNetWare)

QNetWare 파일 시스템은 Novell Netware 4.10 또는 4.11를 실행하는 로컬 또는 리모트 iSeries용 통합 xSeries 서버에 저장된 자료 또는 Novell NetWare 3.12, 4.10, 4.11 또는 5.0을 실행중인 독립형 PC 서버에 대한 액세스를 제공합니다.

또한,

- NetWare 디렉토리 서비스(NDS) 오브젝트에 대한 액세스를 제공합니다.
- 스트림 파일에 저장된 자료를 지원합니다.
- 로컬 이름공간으로의 Netware 파일 시스템의 동적 마운트를 제공합니다.

주: QNetWare 파일 시스템은 iSeries 400용 NetWare Enhanced Integration, BOSS 옵션 25가 시스템에 설치된 경우에만 사용할 수 있습니다. 설치 후 IPL을 수행하면 /QNetWare 디렉토리 및 서브디렉토리가 통합 파일 시스템 디렉토리 구조의 일부로 나타납니다.

QNetWare 파일 시스템에 대한 자세한 정보는 다음 주제를 참조하십시오.

- NetWare 파일 시스템 마운트
- QNetWare 디렉토리 구조
- 통합 파일 시스템 인터페이스를 통한 QNetWare 사용

NetWare 파일 시스템 마운트

Novell NetWare 서버에 있는 NetWare 파일 시스템은 『루트』(/), QOpenSys와 기타 파일 시스템에 마운트되어 보다 쉬운 액세스가 이루어지도록 하며 /QNetWare 디렉토리에서보다 우수한 성능을 나타냅니다. NetWare 파일 시스템을 마운트하면, 읽기-쓰기 파일 시스템을 읽기 전용으로 마운트하는 것과 같은 마운트된 파일 시스템 추가(ADDMFS) 명령의 장점을 이용할 수 있습니다.

NDS 경로를 사용하거나 SERVER/VOLUME:directory/directory의 형식으로 NetWare 경로를 지정하려 NetWare 파일 시스템을 마운트될 수 있습니다. 예를 들어, 서버 Dreyfuss의 볼륨 Nest에 있는 디렉토리 doorway를 마운트하려면, 다음 구문을 사용할 수 있습니다.

```
DREYFUSS/NEST:doorway
```

이 경로 구문은 NetWare MAP 명령 구문과 매우 유사합니다. NDS 경로는 NetWare 볼륨에 대한 경로를 지정하는 데 사용될 수 있으나, 그 자체를 마운트할 수는 없습니다.

QNetWare 디렉토리 구조

/QNetWare 디렉토리 구조는 다수의 독특한 파일 시스템을 나타냅니다.

- 그 구조는 다음과 같은 형식으로 네트워크에서 Novell NetWare 서버와 볼륨을 나타냅니다.

  ```
  /QNetWare/SERVER.SVR/VOLUME
  ```

  확장자 .SVR은 Novell NetWare 서버를 나타내는 데 사용됩니다.
- 서버 아래의 볼륨이 통합 파일 시스템 메뉴, 명령, API를 통해 액세스될 때, NetWare 볼륨의 루트 디렉토리는 /QNetWare 아래의 VOLUME 디렉토리에 자동으로 마운트됩니다.
- QNetWare는 다음의 형식으로 네트워크상에서 NDS 트리를 나타냅니다.

  ```
  /QNetWare/CORP_TREE.TRE/USA.C/ORG.O/ORG_UNIT.OU/SVR1_VOL.CN
  ```

  확장자 .TRE, .C, .O, .OU, .CN은 각각 NDS 트리, 국가, 조직, 조직 단위, 공통 이름을 나타냅니다. Novell NetWare 볼륨이 오브젝트나 별명을 사용한 NDS를 통해 볼륨 오브젝트에 액세스되는 경우, 루트 디렉토리도 자동으로 NDS 오브젝트에 마운트됩니다.

통합 파일 시스템 인터페이스를 통한 QNetWare 사용

QNetWare 파일 시스템은 OS/400 파일 서버 또는 통합 파일 시스템 명령, 사용자 표시 화면 및 API를 사용하여 통합 파일 시스템 인터페이스를 통해 액세스될 수 있습니다. 다음과 같은 고려사항, 제한사항, 의존성 등에 유의해야 합니다.

QNetWare 파일 시스템에서 권한 및 소유권

QNetWare의 파일과 디렉토리는 Novell NetWare 서버에 의해 저장 및 관리됩니다. 소유자나 사용자의 권한을 검색하고 설정하기 위해 명령과 API를 사용할 때, QNetWare는 사용자명을 기준으로 NetWare 사용자를 iSeries 서버 사용자에게 맵핑합니다. NetWare 이름이 10자를 초과하거나 대응하는 iSeries 서버 사용자가 없으면, 권한이 맵핑되지 않습니다. 맵핑되지 않는 소유자는 자동으로 사용자 프로파일 QDFTOWN에 맵핑됩니다. WRKAUT와 CHGAUT 명령을 사용하면 사용자의 권한을 표시하고 변경시킬 수 있습니다. 서버간에 권한이 전송될 때, 권한은 iSeries 서버 권한에 맵핑됩니다.

QNetWare 파일 시스템에서 감사

Novell NetWare가 파일 및 디렉토리의 감사를 지원한다고 할지라도 QNetWare 파일 시스템은 이들 오브젝트의 감사 값을 변경시킬 수 없습니다. 따라서, CHGAUD 명령은 지원되지 않습니다.

QNetWare 파일 시스템에서 파일 및 디렉토리

QNetWare 파일 시스템은 파일이나 디렉토리가 명령이나 API로 입력된 대소문자를 유지하지 않습니다. 모든 이름들은 NetWare 서버로의 전송시 대문자로 설정됩니다. 또한 Novell NetWare는 DOS, OS/2, Apple Macintosh 및 NFS와 같은 복수 플랫폼의 이름공간을 지원합니다. QNetWare 파일 시스템은 DOS 공간만을 지원합니다. DOS 공간이 모든 Novell NetWare 볼륨에 필요하기 때문에, 모든 파일과 디렉토리는 QNetWare 파일 시스템에서 나타납니다.

QNetWare 파일 시스템에서 NDS 오브젝트

QNetWare 파일 시스템은 대소문자로된 NDS 이름의 표시를 지원합니다.

QNetWare 파일 시스템에서 링크

QNetWare 파일 시스템은 오브젝트에 대한 하나의 링크만을 지원합니다. 기호 링크는 QNetWare에 작성되거나 저장될 수 없습니다. 그러나 기호 링크는 QNetWare 파일이나 디렉토리를 가리키는 『루트』(/)나 QOpenSys 디렉토리에서 작성될 수 있습니다.

QNetWare 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

다음 명령을 제외하고 28 페이지의 『CL 명령을 사용한 조작 수행』에 나열된 명령어는 QNetWare 파일 시스템에서 작동됩니다.

ADDLINK
| APYJRNCHG
CHGAUD
CHGPGP
CHKIN
CHKOUT
| ENDJRN
| SNDJRN
| STRJRN
WRKOBJOWN
WRKOBJPGP

앞의 명령에 추가하여 다음은 NDS 오브젝트, 서버, 볼륨에 사용될 수 없습니다.

CHGOWN
CPYFRMSTMF
CPYTOSTMF
CRTDIR

QNetWare 파일 시스템에서 통합 파일 시스템 API 사용

54 페이지의 『API를 사용한 조작 수행』에 나열된 C 언어 함수는 다음의 API를 제외하고 QNetWare 파일 시스템상에서 작동합니다.

givedescriptor()
link()
| QjoEndJournal()
| QjoRetrieveJournalInformation()
| QJORJIDI()
| QJOSJRNE()
| QjoStartJournal()
readlink()

symlink()

takedescriptor()

앞의 API에 추가하여 다음 API는 NDS 오브젝트, 서버, 볼륨에 사용될 수 없습니다.

chmod()

chown()

create()

fchmod()

fchown()

fcntl()

ftruncate()

lseek()

mkdir()

read()

readv()

unmask()

write()

writev()

Windows NT 서버 파일 시스템(QNTC)

QNTC 파일 시스템은 Windows NT 4.0 Server 이상을 실행중인 로컬 또는 리모트 iSeries용 통합 xSeries 서버 또는 또는 독립형 서버에 저장된 자료 및 오브젝트에 대한 액세스를 제공합니다. iSeries 서버 어플리케이션이 Windows NT 클라이언트와 동일한 자료를 사용할 수 있도록 합니다. 스트림 파일에 자료를 저장합니다.

QNTC 파일 시스템은 기본 OS/400 오퍼레이팅 시스템의 일부입니다. QNTC 파일 시스템을 사용하려면 iSeries 400용 TCP/IP Connectivity Utilities(부품 번호: 5769-TC1)가 설치되어 있어야 합니다. /QNTC에 액세스하기 위해 iSeries 400 Integration with Windows NT Server, 오퍼레이팅 시스템의 옵션 29를 설치할 필요는 없습니다.

QNTC에 대한 자세한 정보는 통합 파일 시스템 인터페이스를 통한 QNTC 사용을 참조하십시오.

통합 파일 시스템 인터페이스를 통한 QNTC 사용

OS/400 파일 서버, 통합 파일 시스템 명령, 사용자 화면 또는 API를 사용하여 통합 파일 시스템 인터페이스를 통해 QNTC 파일 시스템에 액세스할 수 있습니다. 이 경우 다음 고려사항 및 제한사항을 참조하십시오.

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 WRKAUT와 CHGAUT 명령을 지원하지 않습니다.

QNTC 파일 시스템에서 대소문자 구분

QNTC 파일 시스템은 입력된 오브젝트명의 대소문자를 동일하게 보존하지만 이름의 대소문자를 구분하지는 않습니다. 오브젝트명에 대한 탐색시 이름의 문자가 대문자인지 소문자인지에 관계없이 동일한 것으로 인식합니다.

QNTC 파일 시스템에서 경로명

- 경로명은 반드시 슬래시(/)로 시작하여 최고 255자까지 사용이 가능합니다.
- 경로명은 대소문자를 구분합니다.
- 경로는 파일 시스템명, Windows NT 서버명, 공유명, 디렉토리 및 서브디렉토리명, 오브젝트명으로 구성됩니다. 경로명은 다음과 같은 형식으로 되어 있습니다.

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

- 서버명은 최고 15자까지 사용이 가능합니다. 이것은 반드시 경로에 포함되어야 합니다.
- 공유명(sharename)은 최고 12자까지 사용이 가능합니다.
- 공유명 뒤에 오는 경로명의 각 구성요소는 최대 255자까지 사용이 가능합니다.
- 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 Unicode CCSID로 저장됩니다.
- 로컬 서브네트 내에서 기능하는 각 Windows NT 서버는 /QNTC 아래의 디렉토리로서 자동으로 표시됩니다. 로컬 서브네트 외부에 Windowns NT 서버를 추가하려면 MKDIR(디렉토리 작성) 명령(29 페이지의 표 2 참조)이나 mkdir() API(54 페이지의 『API를 사용한 조작 수행』 참조)를 사용하십시오.

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 자료에 액세스할 수 있습니다.

링크 설명에 대해서는 19 페이지의 『링크』를 참조하십시오

QNTC 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

| 다음 명령을 제외하고 28 페이지의 『CL 명령을 사용한 조작 수행』에 나열된 명령어는 QNTC 파일 시스템에
| 서 작동됩니다.

ADDLNK

APYJRNCHG

CHGOWN

CHGAUT

CHGPGP

CHKIN

CHKOUT

DSPAUT

ENDJRN

RST

SAV

SNDJRNE

STRJRN

WRKAUT

WRKOBJOWN

WRKOBJPGP

동일한 제한사항이 27 페이지의 『iSeries 메뉴 및 표시장치를 사용하여 조작 수행』에 설명된 사용자 화면에 적용됩니다.

QNTC 파일 시스템에서 MKDIR 명령 사용

/QNTC 디렉토리에 서버 디렉토리를 추가하려면 MKDIR(디렉토리 작성) 명령을 사용하십시오. 로컬 서브네트 내에서 기능하는 모든 Windows NT 서버들은 자동으로 작성됩니다. 로컬 서브네트 외부의 해당 Windows NT 서버들은 MKDIR 명령이나 mkdir() API를 사용하여 추가시켜야 합니다. 예를 들면, 다음과 같습니다.

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

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

또한 TCP/IP 주소를 사용하여 디렉토리 구조에 새로운 서버를 추가할 수도 있습니다. 예를 들면, 다음과 같습니다.

```
MKDIR '/QNTC/9.130.67.24'
```

위의 명령은 QNTC 파일 시스템 디렉토리 구조에 서버를 추가합니다.

주: mkdir() API 또는 MKDIR CL 명령을 사용하여 디렉토리 구조에 디렉토리를 추가하는 경우, 디렉토리는 IPL 간에 시각적으로 남아있지 않습니다. MKDIR 명령이나 mkdir() API는 모든 시스템에 IPL이 이루어진 후 다시 발행되어야 합니다.

QNTC 파일 시스템에서 통합 파일 시스템 API 사용

다음을 제외하고 54 페이지의 『API를 사용한 조작 수행』에 나열된 C 언어 함수는 QNTC 파일 시스템상에서 작동됩니다.

- chmod(), fchmod(), utime(), umask() 함수는 QNTC 내의 오브젝트에 영향을 주지 않지만 사용하는 것이 오류를 일으키지는 않습니다.
- QNTC 파일 시스템은 다음 함수를 지원하지 않습니다.

  chown()

  fchown()

  givedescriptor()

  link()

  QjoEndJournal()

  QjoRetrieveJournalInformation()

  QJORJIDI()

  QJOSJRNE()

  QjoStartJournal()

  Qp0lGetPathFromFileID()

  readlink()

  symlink()

  takedescriptor()

OS/400 파일 서버 파일 시스템(QFileSvr.400)

OS/400 파일 서버 파일 시스템은 리모트 iSeries 서버에 상주하는 기타 파일 시스템에 대한 투명한 액세스를 제공합니다. 이는 계층적 디렉토리 구조를 통해 액세스됩니다.

QFileSvr.400 파일 시스템은 사용자를 대신하여 파일 요청을 수행하는 클라이언트로 간주될 수 있습니다. QFileSvr.400은 목표 시스템의 OS/400 파일 서버와 상호 작용하여 실제 파일 조작을 수행합니다.

QFileSvr.400에 대한 자세한 정보는 통합 파일 시스템 인터페이스를 통한 QFileSvr.400 사용을 참조하십시오.

통합 파일 시스템 인터페이스를 통한 QFileSvr.400 사용

QFileSvr.400 파일 시스템은 OS/400 파일 서버 또는 통합 파일 시스템 명령, 사용자 화면 및 API를 사용하여 통합 파일 시스템 인터페이스를 통하여 액세스할 수 있습니다. 통합 파일 시스템 인터페이스 사용에서는 다음 고려사항 및 제한사항을 참조하십시오.

주: QFileSvr.400 파일 시스템의 특성은 목표 서버에서 액세스 중인 파일 시스템의 특성에 의해 판별됩니다.

OS/400 파일 서버 파일 시스템에서 대소문자 구분

실제로 목표 시스템의 『루트』(/) 디렉토리를 대표하는 제1 레벨 디렉토리의 경우, QFileSvr.400 파일 시스템은 오브젝트명이 입력되는 형식과 같은 대소문자입니다. 그러나 QFileSvr.400이 이름을 탐색할 때 대소문자를 구분하여 인식하지 않습니다.

다른 모든 디렉토리의 경우, 대소문자 구분은 액세스중인 특정 파일 시스템에 따라 다릅니다. QFileSvr.400은 파일 요청이 OS/400 파일 서버로 전송될 때 입력된 오브젝트명과 같은 대소문자 형식을 유지합니다.

OS/400 파일 서버 파일 시스템에서 경로명

- 경로명은 다음과 같은 형식으로 되어 있습니다.

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

  제1 레벨 디렉토리(즉, 위의 예에 있는 RemoteLocationName)는 다음 두 항목 모두를 나타냅니다.

  – 통신 연결 설정에 사용될 목표 서버명. 목표 서버명은 다음 중 하나입니다.

    - TCP/IP 호스트명(예: beowulf.newyork.corp.com)
    - SNA LU 6.2명(예: appn.newyork)

  – 목표 서버의 『루트』(/) 디렉토리

  따라서, 제1 레벨 디렉토리가 통합 파일 시스템 인터페이스를 사용하여 작성되는 경우, 지정된 모든 속성은 무시됩니다.

  주: 제1 레벨 디렉토리는 다음 IPL시에 계속되지 않습니다. 다시 말해, 제1 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.400에 의해 허용되는 최대값보다 작게 제한할 수 있습니다.

- 프로그램 및 시스템 제한과 액세스중인 파일 시스템에 의해 부과되는 제한을 제외하고 디렉토리 계층 깊이에는 제한이 없습니다.
- 이름 저장시 문자는 UCS2 레벨 1 형식으로 변환됩니다(24 페이지의 『이름 연속성』 참조).

OS/400 파일 서버 파일 시스템에서 통신

- 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 연결시 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 프로그래밍  을 참조하십시오.

- 통신 프로토콜로 TCP를 사용하는 파일 서버 요청은 요청하는 작업의 문맥(context) 내에서 수행됩니다. SNA를 통신 프로토콜로 사용하는 파일 서버 요청은 OS/400 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.
  2. 해결된 IP 주소를 사용하여 잘 알려진 포트인 449상의 호스트 서버의 서버 맵퍼로 연결하십시오. 그런 후, 서버 맵퍼에 서비스명『as-file』에 대한 조회를 보내십시오. 조회 결과에 따라 다음 중 하나가 발생합니다.
     - 『as-file』이 목표 서버의 서비스 표에 있는 경우, 서버 맵퍼는 OS/400 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 OS/400 파일 서버 디먼과 TCP 연결 설정을 시도합니다. 연결시 QFileSvr.400은 요청을 교환하고 파일 서버로 응답합니다. QSERVER 서브시스템에서 QPWFSERVSO 사전시작 요청은 연결을 제어합니다. 사전시작 작업은 각자의 사용자 프로파일 아래서 수행됩니다.
  3. 리모트 위치명이 IP 주소로 해결되지 않는 경우, 제1 레벨 디렉토리가 SNA LU 6.2명으로 간주됩니다. 따라서, APPC를 OS/400 파일 서버로 연결하려는 시도가 이루어집니다.
- QFileSvr.400 파일 시스템은 정기적으로(2시간 마다) 사용하지 않는 연결이 있는지(예를 들면, 연결과 연관된 열린 파일이 없는 경우) 검사하고, 2시간 동안 연결에 어떤 활동도 없었는지 검사합니다. 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 지정될 때, QFileSvr.400 파일 시스템은 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.

OS/400 파일 서버 파일 시스템에서 보안 및 오브젝트 권한

두 시스템에 Kerberos가 구성되어 있고 사용자가 Kerberos에 대해 인증된 경우 Kerberos를 사용하여 목표 iSeries 서버에 상주하는 파일 시스템을 인증할 수 있습니다. Kerberos 인증이 실패하면 사용자 ID와 암호를 사용하여 액세스를 확인합니다.

주: 목표 서버가 사용자의 액세스를 확인한 후 티켓 허가 티켓이나 서버 티켓이 만기되는 경우, 목표 서버와의 연결이 종료될 때까지는 만기가 유효하지 않습니다. Kerberos에 대한 자세한 정보는 iSeries Information Center의 네트워크 인증 서비스 주제를 참조하십시오.

- 목표 iSeries 서버에 상주하는 파일 시스템에 액세스하려면 사용자는 Kerberos가 인증에 사용되지 않을 경우, 로컬 서버의 사용자 ID 및 암호와 일치하는 사용자 ID 및 암호를 목표 서버에 갖고 있어야 합니다.

  주: 목표 서버가 사용자의 액세스를 확인한 후, 로컬 서버나 목표 서버의 사용자 암호가 변경되면, 변경사항은 목표 서버와의 연결이 종료될 때까지 반영되지 않습니다. 그러나 로컬 서버의 사용자 프로파일이 삭제되고, 동일한 사용자 ID를 사용하여 다른 사용자 프로파일이 작성되는 경우에는 지연되지 않습니다. 이 경우, QFileSvr.400 파일 시스템은 사용자에게 목표 서버에 대한 액세스가 있는지 확인합니다.
- 오브젝트 권한은 목표 서버에 상주하는 사용자 프로파일을 기반으로 합니다. 즉, 목표 서버의 사용자 프로파일이 오브젝트에 대한 적절한 권한을 기지고 있는 경우에만 목표 서버의 파일시스템에 있는 오브젝트에 액세스할 수 있습니다.

OS/400 파일 서버 파일 시스템에서 링크

QFileSvr.400 파일 시스템은 오브젝트와 하나의 링크만을 지원합니다. QFileSvr.400에서 기호 링크는 작성되거나 저장될 수 없습니다. 그러나 QFileSvr.400에 있는 파일은 『루트』(/), QOpenSys, 사용자 정의 파일 시스템에서 기호 링크를 사용하여 액세스할 수 있습니다.

링크 설명에 대해서는 19 페이지의 『링크』를 참조하십시오

OS/400 파일 서버 파일 시스템에서 통합 파일 시스템 명령 및 표시 화면 사용

다음 명령을 제외하고 28 페이지의 『CL 명령을 사용한 조작 수행』에 나열된 명령어는 QFileSvr.400 파일 시스템에서 작동됩니다.

ADDLNK
APYJRNCHG
CHGAUT
CHGOWN
DSPAUT
ENDJRN
RST
SAV
SNDJRNE
STRJRN
WRKOBJOWN
WRKOBJPGP

동일한 제한사항이 27 페이지의 『iSeries 메뉴 및 표시장치를 사용하여 조작 수행』에 설명된 사용자 화면에 적용됩니다.

OS/400 파일 서버 파일 시스템에서 통합 파일 시스템 API 사용

다음을 제외하고 54 페이지의 『API를 사용한 조작 수행』에 나열된 C 언어 함수는 QFileSvr.400 파일 시스템상에서 작동됩니다.

chown()

fchown()

givedescriptor()

link()

| QjoEndJournal()

| QjoRetrieveJournalInformation()

| QJORJIDI()

| QJOSJRNE

| QjoStartJournal

Qp0lGetPathFromFileID()

symlink()

takedescriptor()

네트워크 파일 시스템(NFS)

NFS 파일 시스템은 리모트 NFS 서버에 저장된 오브젝트 및 자료에 대한 액세스를 제공합니다. 따라서, NFS 서버는 NFS 클라이언트가 동적으로 마운트시키는 네트워크 파일 시스템을 내보낼 수 있습니다.

또한 네트워크 파일 시스템을 통해 로컬로 마운트되는 모든 파일 시스템은 리모트 서버에서 마운트되었던 디렉토리나 파일 시스템의 피처, 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.

NFS에 대한 자세한 정보는 통합 파일 시스템 인터페이스를 통한 NFS 파일 시스템 사용을 참조하십시오.

통합 파일 시스템 인터페이스를 통한 NFS 파일 시스템 사용

네트워크 파일 시스템은 통합 파일 시스템 인터페이스를 통해 액세스될 수 있으며, 여기에는 다음과 같은 고려사항과 제한사항이 있습니다.

네트워크 파일 시스템의 특성

NFS를 통해 마운트된 파일 시스템의 특성은 서버로부터 마운트되었던 파일 시스템의 유형에 의존한다는 것입니다. 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 시스템에서 서버 및 클라이언트 변화

클라이언트/서버 연결에는 네트워크 파일 시스템이 어떻게 기능하고 어떤 특성을 갖게 될지에 관해 영향을 주는 세 가지 주요한 가능성이 있습니다.

1. 사용자가 클라이언트에 있는 iSeries 서버로부터 파일 시스템을 마운트하는 경우
2. 사용자가 클라이언트에 있는 UNIX 서버로부터 파일 시스템을 마운트하는 경우
3. 사용자가 클라이언트에 있는 그 밖의 iSeries, UNIX 서버로부터 파일 시스템을 마운트하는 경우

첫 번째 시나리오에서는 마운트된 파일 시스템이 iSeries 서버에서 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 제한으로 작동하게 됩니다. 예를 들어, QDLS 파일 시스템에서 경로명 구성요소는 3자의 확장자를 제외하고 8자로 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 iSeries 서버 QOpenSys 파일 시스템과 매우 유사하게 작동한다는 사실을 알아두어야 합니다. QOpenSys 파일 시스템에 관한 자세한 정보는 70 페이지의 『개방 시스템 파일 시스템(QOpenSys)』을 참조하십시오.

세 번째 시나리오에서는 서버의 오퍼레이팅 시스템과 연관된 파일 시스템의 자료를 검토해야 합니다.

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.

링크 설명에 대해서는 19 페이지의 『링크』를 참조하십시오

네트워크 파일 시스템에서 통합 파일 시스템 명령 사용

28 페이지의 『CL 명령을 사용한 조작 수행』에 나열된 모든 명령어들과 27 페이지의 『iSeries 메뉴 및 표시장치를 사용하여 조작 수행』에 표시된 모든 화면들은 다음 명령을 제외한 네트워크 파일 시스템에서 조작할 수 있습니다.

- APYJRNCHG
- CHGAUD
- CHGATR
- CHGAUT

| • CHGOWN
| • CHGPGP
| • CHKIN
| • CHKOUT
| • ENDJRN
| • SNDJRNE
| • STRJRN

네트워크 파일 시스템과 일반적으로 마운트되어 있는 기타 파일 시스템에 고유한 몇 가지 CL 명령이 있습니다. 그러나 다중 스레드가 가능한 프로세스에서 이 명령을 사용하는 것은 안전하지 않을 수도 있습니다. 다음은 이와 같은 명령에 관한 설명입니다. 특히 네트워크 파일 시스템과 관련된 명령 및 표시장치에 대한 자세한 설명은 OS/400 네트워크 파일 시스템 지원 을 참조하십시오.

표 7. 네트워크 파일 시스템 CL 명령

명령	설명
ADDMFS	마운트된 파일 시스템 추가는 내보낸 리모트 서버 파일 시스템을 로컬 클라이언트 디렉토리에 위치시킵니다.
CHGNFSEXP	네트워크 파일 시스템 내보내기 변경은 네트워크 파일 시스템 클라이언트로 내보낸 파일 시스템의 내보내기 표에 디렉토리 트리를 추가하거나 제거합니다.
DSPMFSINF	마운트된 파일 시스템 정보 표시는 마운트 파일 시스템에 대한 정보를 표시합니다.
ENDNFSSVR	네트워크 파일 시스템 서버 종료는 서버상의 하나 또는 모든 네트워크 파일 시스템 디먼을 종료합니다.
EXPORTFS	파일 시스템 내보내기는 네트워크 파일 시스템 클라이언트로 내보낸 파일 시스템의 내보내기 표에 디렉토리 트리를 추가하거나 제거합니다.
MOUNT	파일 시스템 마운트는 내보낸 리모트 서버 파일 시스템을 로컬 클라이언트 디렉토리상에 위치시킵니다. 이 명령은 ADDMFS 명령에 대한 별명입니다.
RLSIFSLCK	통합 파일 시스템 잠금 해제는 클라이언트 또는 오브젝트상에서 유지되고 있는 모든 네트워크 파일 시스템 바이트 범위의 잠금을 해제합니다.
RMVMFS	마운트된 파일 시스템 제거는 로컬 클라이언트 이름공간에서 내보낸 리모트 서버 파일 시스템을 제거합니다.
STRNFSSVR	네트워크 파일 시스템 서버 시작은 서버상의 하나 또는 모든 네트워크 파일 시스템 디먼을 시작합니다.
UNMOUNT	파일 시스템 마운트 해제는 내보낸 리모트 서버 파일 시스템을 로컬 클라이언트 이름공간으로부터 제거합니다. 이 명령은 RMVMFS 명령에 대한 별명입니다.

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

네트워크 파일 시스템에서 통합 파일 시스템 API 사용

| 54 페이지의 『API를 사용한 조작 수행』에 나열된 모든 C 언어 함수는 다음 명령을 제외한 네트워크 파일 시
| 스템상에서 작동될 수 있습니다.

| • QjoEndJournal()
| • QjoRetrieveJournalInformation()

| • QJORJIDI()
| • QJOSJRNE()
| • QjoStartJournal()

특히 네트워크 파일 시스템과 관련된 C 언어 함수에 대한 자세한 설명은 OS/400 네트워크 파일 시스템 지원  을 참조하십시오.

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

제 7 장 통합 파일 시스템 오브젝트에 대한 저널링 지원

저널링의 1차 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 회복할 수 있도록 하는 것입니다.

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

다음 주제에서는 통합 파일 시스템 오브젝트에 대한 저널링 지원을 설명합니다.

- 『저널 관리』
- 『저널링해야 할 오브젝트』
- 102 페이지의 『저널링된 통합 파일 시스템 오브젝트』
- 103 페이지의 『저널링된 조작』
- 104 페이지의 『저널 항목에 대한 특수 고려사항』

| 통합 파일 시스템 오브젝트 저널링에 대한 자세한 정보는 iSeries Information Center에서 저널 관리를 참조
| 하십시오.

저널 관리

저널 관리의 기본 목적은 오브젝트가 마지막으로 저장된 이후에 발생한 오브젝트에 대한 변경사항을 회복할 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 회복
- 어플리케이션 프로그램 테스트 지원

저널을 사용하여 저널 관리로 보호하려는 오브젝트를 정의할 수 있습니다. 오브젝트 저널링에 대한 자세한 고려사항은 『저널링해야 할 오브젝트』를 참조하십시오. 통합 파일 시스템에서, 스트림 파일, 디렉토리, 기호 링크를 저널링할 수 있습니다. "루트"(/), QOpenSys, UDFS 파일 시스템의 오브젝트만 지원됩니다.

저널링해야 할 오브젝트

통합 파일 시스템 오브젝트 저널링 여부를 결정할 때 다음 질문을 고려하십시오.

- 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 서버의 기타 오브젝트와 얼마나 관련이 있습니까? 특정 오브젝트의 자료가 자주 변경되지 않더라도, 해당 오브젝트의 자료가 서버에 있는 더욱 동적인 다른 오브젝트에 비해 중요할 수 있습니다. 예를 들면, 고객의 마스터 파일에 의존하는 오브젝트가 많습니다. 주문을 재구성하는 경우, 고객 마스터 파일에는 이전에 저장된 이후에 발생한 신용 한도 변경사항 또는 새로운 고객이 포함되어야 합니다.

저널링된 통합 파일 시스템 오브젝트

OS/400 저널링 지원을 사용하여 저널링할 수 있는 일부 통합 파일 시스템 오브젝트 유형이 있습니다. 지원되는 오브젝트 유형은 스트림 파일, 디렉토리, 기호 링크입니다. ″루트″(/), QOpenSys, UDFS 파일 시스템만 이런 오브젝트 유형을 지원합니다. 일반적인 시스템 인터페이스(CL 명령 또는 API)나 iSeries Navigator를 사용하여 통합 파일 시스템 오브젝트를 저널링할 수 있습니다. iSeries Navigator 뿐만 아니라 화면 저널링 정보를 통해 저널링을 시작하고 저널링을 종료할 수 있습니다.

주: 스트림 파일에 맵핑된 메모리와 가상 드라이브 기억장치 공간용 the iSeries용 통합 xSeries 서버 (IXS)에 사용되는 스트림 파일은 저널될 수 없습니다.

다음은 통합 파일 시스템의 저널링 지원을 요약한 리스트입니다.

- 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 오브젝트 전체 서브트리에 대해 저널링 시작, 저널링 종료, 저널링된 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 들면, 저널링 시작 명령을 사용하여 `"*.data"` 패턴과 일치하는 `"/MyCompany"` 트리의 모든 오브젝트에 대해 저널링을 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: 오브젝트에 대한 저널링을 끝내고 현재 같은 디렉토리에 상주하는 오브젝트명을 변경하면, 디렉토리가 상속의 저널링 속성을 갖는다 하더라도 오브젝트에 대해 저널링을 시작할 수 없습니다.
- 오브젝트명과 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) 회복이 제공됩니다.

- write() 및 writev() API에서 지원하는 최대 쓰기 한계는 2GB-1입니다. RCVSIZOPT(*MAXOPT2)를 지정한 경우, 최대 저널 항목 크기는 4,000,000,000입니다. 지정하지 않은 경우, 최대 저널 항목 크기는 15,761,440바이트입니다. 스트림 파일을 저널링하고 15,761,440바이트를 초과하는 쓰기를 갖고 있는 경우, *MAXOPT2 지원을 사용하여 오류가 발생하지 않도록 할 수 있습니다.

통합 파일 시스템 오브젝트 저널링에 대한 자세한 내용은 iSeries Information Center에서 저널 관리를 참조하십시오.

다양한 저널 항목 배치에 대한 자세한 정보는 QSYSINC/H(QP0LJRNL) 멤버에 제공된 C 언어 포함 파일 qp0ljrnl.h에 있으며, 통합 파일 시스템 저널 항목별 자료 내용과 형식 세부사항이 들어 있습니다.

| 통합 파일 시스템 오브젝트에 배치되는 모든 저널 항목의 전체 리스트는 iSeries Information Center의 Journal
| code finder를 참조하십시오.

저널링된 조작

다음 조작은 조작이 사용되고 있는 오브젝트나 링크 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.

- 자료 쓰기
- 파일 절단/확장
- 파일 자료 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
- 저널링 종료
- APYJRNCHG(저널링된 변경사항 적용) 명령 시작
- APYJRNCHG(저널링된 변경사항 적용) 명령 종료
- 저장

- 복원

통합 파일 시스템 오브젝트 저널링에 대한 자세한 내용은 iSeries Information Center에서 저널 관리를 참조하십시오. 통합 파일 시스템 오브젝트에 배치된 모든 저널 항목의 전체 리스트는 **시스템 관리** 주제의 저널 코드 파인더를 참조하십시오.

저널 항목에 대한 특수 고려사항

다수의 저널링된 통합 파일 시스템 조작이 내부적으로 확약 제어를 사용하여 조작 시 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)
- FA -- 속성 변경
- JT -- 저널 시작(상속 저널링 속성이 '예'인 디렉토리에서의 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 자료 필드가 있습니다. 요약 항목 유형을 송신하는 조작은 두 가지의 동일한 유형의 항목을 저널로 송신합니다. 첫 번째 항목에는 입력 항목별 자료 서브세트가 들어 있습니다. 두 번째 항목에는 전체 입력 항목별 자료가 들어 있어 요약 항목임을 표시합니다. 오브젝트 복제 및/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. 이런 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 회복이 수행되었습니다.

부록 A. 전송 독립 리모트 프로시듀어 호출(TI-RPC)

Sun Microsystems사에서 개발한 RPC(리모트 프로시듀어 호출)는 클라이언트 어플리케이션을 서버 구조로부터 쉽게 분산시킵니다. 여기에는 여러 유형의 기계에서 자료 전송을 가능케 하는 자료 표시, 호출된 외부 자료 표시 또는 XDR이 포함됩니다. TI-RPC는 RPC의 최신 버전입니다. 이 버전은 한 프로토콜에서 다른 프로토콜로 무리없는 전환을 제공하면서, 네트워크 계층에 사용되는 기본 프로토콜을 분리하는 방법을 제공합니다. iSeries 서버에서는 현재 TCP와 UDP 프로토콜만 사용할 수 있습니다.

네트워크 간에 걸친 분산 어플리케이션의 개발은 RPC를 사용할 때 무리없는 타스크가 됩니다. 1차적인 목표는 사용자 인터페이스나 자료 검색을 분산시키는 데 중점을 두고 있는 어플리케이션입니다.

네트워크 선택

다음 API에서는 어플리케이션이 실행되어야 하는 전송 선택 방법을 제공합니다.

이 API를 위해서는 시스템에 *STMF /etc/netconfig 파일이 있어야 합니다. netconfig 파일이 /etc 디렉토리에 없는 경우, /QIBM/ProdData/OS400/RPC 디렉토리에서 해당 파일을 복사하여야 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 리턴시킵니다.

이름 대 주소 변환

다음 API에서는 어플리케이션이 전송 독립 방식으로 서비스나 지정 호스트를 구할 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.
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.

API	설명
taddr2uaddr()	전송 고유(로컬) 주소를 전송-독립(범용) 주소로 변환합니다.
uaddr2taddr()	전송 독립(범용) 주소를 전송 고유(로컬) 주소(netbuf 구조)로 변환합니다.

외부 자료 표시(XDR)

다음 API는 리모트 프로시듀어 호출(RPC) 어플리케이션들이 다른 호스트의 바이트 순서나 배치 규약과 관계없이 임의 자료 구조를 핸들할 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 자료를 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 2바이트 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 자료와 해당 외부 표시간에 변환시키는 필터 원어
xdr_opaque()	고정 크기 opaque 자료와 해당 외부 표시간에 변환시키는 필터 원어
xdr_pointer()	구조 안에서 추적하기 위한 포인터를 제공하고 널 포인터를 차례로 나열합니다. 2진 트리나 연결 리스트 등과 같은 순환 자료 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. 차례로 나열되는 동안 xdr_reference() 함수에서는 널 포인터로 특별한 의미를 추가하지 않지만, 널 포인터의 주소를 전달하는 것이 메모리 오류의 원인이 될 수 있습니다. 따라서, 프로그래머는 반드시 두 면의 구분 집합으로 자료를 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
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

API	설명
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 자료를 전송하지는 않습니다.
xdr_wrapstring()	xdr_string(xdr, sp, maxuint) API를 호출하는 원어로서 maxuint는 부호없는 정수의 최대값입니다. xdr_wrapstring()는 RPC 패키지가 매개변수로서 최대 두 개의 XDR 함수를 전달하므로, 유용한 반면에 xdr_string() 함수에서는 세 개를 필요로 합니다.

인증

다음 API에서는 전송 독립 리모트 프로시듀어 호출(TI-RPC) 어플리케이션으로 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.

전송 독립 RPC(TI-RPC)

다음 API는 특정 전송 피처로부터 어플리케이션을 분리시켜 분산 어플리케이션 개발 환경을 제공합니다. 이것은 전송에 사용상의 편리함을 추가한 것입니다.

TI-RPC 단순 API

다음의 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.

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()	총칭 클라이언트 핸들을 작성합니다. clnt_destroy() 클라이언트의 RPC 핸들을 파기합니다.
clnt_destroy()	클라이언트의 RPC 핸들을 파기합니다.
svc_create()	서버 핸들을 작성합니다.

API	설명
svc_destroy()	RPC 서비스 전송 핸들을 파기합니다.

TI-RPC 중간 레벨 API

다음 API는 최고 레벨 API와 유사하지만, 사용자 어플리케이션에서 네트워크 선택 API를 사용해 전송 고유 정보를 선택합니다.

API	설명
clnt_tp_create()	클라이언트 핸들을 작성합니다.
svc_tp_create()	서버 핸들을 작성합니다.

TI-RPC 전문가 레벨 API

다음 API에서는 어플리케이션이 사용할 전송을 선택할 수 있도록 합니다. 또한 CLIENT와 SVCXPRT 핸들의 세부사항에 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.

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 자료를 해제시킵니다.
clnt_geterr()	클라이언트 핸들로부터 오류 구조를 가져옵니다.
svc_freeargs()	RPC나 XDR 시스템에서 할당한 자료를 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.

API	설명
svcerr_systemerr()	시스템 오류에 대해 클라이언트로 정보를 송신합니다.

부록 B. 통합 파일 시스템 C 함수를 사용한 프로그램 예

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.

이 면책사항 정보는 코드 예에 관한 사항입니다.

```
#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;
}
```

부록 C. 통합 파일 시스템 RPG 코드 예

Code Snippets 에는 통합 파일 시스템 RPG 코드 예가 포함되어 있습니다. 이 예를 보려면, 다음 단계를 수행하십시오.

1. 탐색 범주의 드롭 다운 리스트에서 **ILE RPG 소스**를 선택하십시오.
2. **탐색**을 누르십시오.
3. **RPG에서 IFS 사용**이 나타날 때까지 리스트를 아래로 화면이동하십시오.
4. **RPG에서 IFS를 사용하기 위한 코드**를 클릭하십시오.

이 면책사항 정보는 코드 예에 관한 사항입니다.

참고 문헌

도서 목록에는 이 서적에서 논의하는 정보에 대한 백그라운드 정보나 추가 정보가 들어 있는 iSeries 서버 정보가 나열되어 있습니다.

- iSeries Information Center의 **프로그래밍** 범주에 있는 제어 언어 주제는 iSeries 서버 제어 언어(CL)와 명령에 대해 설명합니다. 각 명령 설명에는 구문 다이어그램, 매개변수, 디폴트 값, 키워드 및 예가 포함됩니다.
- iSeries Information Center의 Globalization 주제에서는 문자 세트 및 코드 페이지와 같은 자국어 지원(NLS) 개념에 대해서 설명하고, iSeries 서버 NLS 및 복수 언어 기능을 평가, 계획 및 사용하는 데 필요한 정보를 제공합니다.
- iSeries Information Center의 **프로그래밍** 범주에 있는 API 주제에서는 통합 파일 시스템 API를 포함하여 각 OS/400 API에 대해 설명합니다.
- iSeries Information Center **시스템 관리** 범주의 저널 관리 주제에서는 iSeries 서버의 시스템 관리 액세스 경로 보호(SMAPP), 로컬 저널 및 리모트 저널에 대한 설정, 관리 및 문제 해결 방법에 대한 정보를 제공합니다.
- iSeries Information Center **데이터베이스** 범주의 확약 제어 주제에서는 데이터베이스 파일이나 통합 파일 시스템 파일 등의 자원에 대한 일련의 변경 사항을 논리적 작업 단위로 정의하고 처리하는 방법에 대해 설명합니다.

- OS/400 네트워크 파일 시스템 지원
  이 서적은 일련의 실제 어플리케이션을 통한 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.

- Optical Support
  이 서적은 OS/400상에서 IBM Optical Support 사용자 안내서 및 참조서입니다. 이 서적에 포함되어 있는 정보를 참조로 하여 사용자는 광 라이브러리 자료 서버 개념 이해, 광 라이브러리 계획, 광 라이브러리 자료 서버 관리 및 조작, 광 자료 서버 문제 해결등을 쉽게 할 수 있습니다.
- WebSphere Development Studio: ILE C/C++ Programmers Guide
  이 서적은 iSeries 서버에서 ILE C/400 프로그램을 설계, 편집, 컴파일, 실행 및 디버그하는 데 필요한 정보를 제공합니다.
- WebSphere Development Studio: C/C++ Language Reference
  이 서적은 ILE C/400 프로그램의 구조에 대한 정보를 제공하고, 라이브러리 기능에 대한 상세한 내용이 들어 있으며, 헤더 파일이 포함됩니다.

- 보안 -- 참조
  이 서적은 OS/400 보안에 관한 상세한 기술적 정보를 제공합니다.

- APPC 프로그래밍
  이 서적은 iSeries 서버에 대한 APPC(advanced program-to-program communication) 지원에 대해 설명합니다. APPC를 사용하는 어플리케이션 프로그램 개발 및 APPC용 통신 환경 정의 안내가 나와 있습니다.

- 백업 및 회복
  이 서적은 IBM iSeries 서버의 회복 및 가용성 옵션에 대한 일반 정보를 제공합니다.

색인

[가]

[나]

[다]

[라]

[마]

[바]

[사]

[아]

[자]

[카]

[타]

[파]

W

Printed in U.S.A