X/Open Preliminary Specification

CD-ROM Support Component (XCDR)

The Open Group

X/Open Preliminary Specification

CD-ROM Support Component (XCDR)

Document Number: P120

Published in the U.K. by The Open Group, May 1991.

Any comments relating to the material contained in this document may be submitted to:

The Open Group
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

OGSpecs@opengroup.org

Contents

List of Figures

Preface

This Document

This document is a Preliminary Specification (see above). It describes the functionality of, and the application programming interface to, the X/Open CD-ROM Support Component (XCDR).

XCDR consists of hardware and software that accept and can read a CD-ROM disc with the format defined in this specification, that offer users and system administrators the commands defined in this specification, and that offer application programs the application programming interface defined in this specification.

XCDR supports CD-ROM discs whose physical format conforms with ISO/IEC 10149, and whose volume and file structure conform with ISO 9660, with certain restrictions, as defined herein.

The XCDR application programming interface allows an application to access CD-ROM-specific information using a set of services that are consistent across all X/Open-compliant systems that have XCDR.

This specification is based on a specification by Philips Kommunikations Industrie AG and can be implemented without obtaining a licence.

Trade Marks

Acknowledgements

X/Open gratefully acknowledges the contribution made by Philips in the development of this document, both in the provision of an original specification to serve as a basis for the X/Open specification, and in the provision of editorial assistance.

Referenced Documents

The following documents are referenced in this specification:

ISO 9660, 1988
: ISO 9660, 1988: Information Processing - Volume and file structure of CD-ROM for information interchange. 1988-09-01.

ISO / IEC 10149, 1989
: ISO / IEC 10149, 1989: Information Technology. Data Interchange on read-only 120 mm optical data discs (CD-ROM). 1989-09-01

Introduction to CD-ROM XA
: An Introduction to CD-ROM XA, First Release; Philips, Microsoft, Sony; September 1989.

XCU, Issue 3
: X / Open Portability Guide, Issue 3, Volume 1, XSI Commands and Utilities.

XSH, Issue 3
: X / Open Portability Guide, Issue 3, Volume 2, XSI System Interface and Headers.

Chapter 1

Introduction

1.1 Overview

1.1.1 Rationale

This document specifies an application programming interface (API) which allows an application to access CD-ROM-specific information using a set of services that are consistent across all X/Open-compliant systems that have the X/Open CD-ROM Support Component (XCDR).

This specification also allows information which is fixed on a CD-ROM, such as User ID and Group ID, to be adapted to the best suitable values in the target system. In addition, it allows defaults to be specified for certain information which might not be specified on an actual CD-ROM. Finally, it allows changing the interpretation of certain properties on a CD-ROM which conform to ISO 9660 , in a way that better suits the common practices in X/Open-compliant systems.

The target usages of CD-ROM in X/Open-compliant systems are:

- information retrieval
- software distribution.

The current specification is suitable for information retrieval and, to a more limited extent, software distribution. It is usable for a form of software distribution where the software is copied to magnetic disc. It is less suitable for a form of software distribution where applications are directly executed from mounted CD-ROM discs. See also Section 1.6 on page 8.

1.1.2 Contents

In this specification, the functionality and API of XCDR are described. These comprise the definition of the format on CD-ROM discs and the service of commands and library components offered to users and applications.

1.2 Conformance

1.2.1 Structure

CD-ROM support can be found in those X/Open-compliant systems that have the X/Open CD-ROM Support Component (XCDR).

The XCDR consists of hardware and software that accept and can read a CD-ROM disc with the format specified in Chapter 4 on page 25, that offer users and system administrators the commands specified in Chapter 5 on page 27, and that offer application programs the API specified in Chapter 6 on page 41.

Figure 1-1 illustrates the structure.

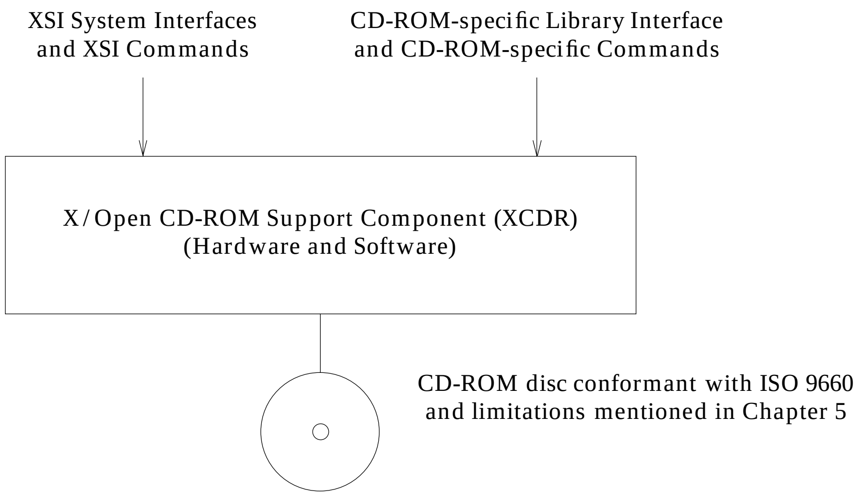
XSI System Interfaces and XSI Commands
CD-ROM-specific Library Interface and CD-ROM-specific Commands
X/Open CD-ROM Support Component (XCDR) (Hardware and Software)
CD-ROM disc conformant with ISO 9660 and limitations mentioned in Chapter 5

Figure 1-1 Structure of XCDR

1.2.2 Mandatory

All interfaces and disc formats are mandatory.

1.2.3 Optional

There are no optional interfaces and disc format elements specified. The CD-ROM may, however, have recorded additional structures (as defined by ISO 9660) which may or may not be exposed to the application by a specific implementation. It is further left free whether an XCDR implementation has less restrictions or more interfaces than implied by this specification. Some information is given for the behaviour when some of the restrictions would not apply. Areas where XCDR implementations could have more functionality are, for example:

- supporting transparent access to multi-volume sets
- removing the limitation on length of File/Directory Identifiers.

1.3 Use and Implementation of Interfaces

Each of the following statements applies unless explicitly stated otherwise in the detailed descriptions that follow. If an argument to a function has an invalid value (such as a value outside the domain of the function, or a pointer outside the address space of the program or a null pointer), the behaviour is undefined. Any function declared in a header may be implemented as a macro defined in the header, so a library function should not be declared explicitly if its header is included. Any macro definition of a function can be suppressed locally by enclosing the name of the function in parentheses, because the name is then not followed by the left parenthesis that indicates expansion of a macro function name. For the same syntactic reason, it is permitted to take the address of a library function even if it is also defined as a macro. The use of **#undef** to remove any macro definition will also ensure that an actual function is referred to. Any invocation of a library function that is implemented as a macro will expand to code that evaluates each of its arguments exactly once, fully protected by parentheses where necessary, so it is generally safe to use arbitrary expressions as arguments. Likewise, those function-like macros described in the following sections may be invoked in an expression anywhere a function with a compatible return type could be called.

Provided that a library function can be declared without reference to any type defined in a header, it is also permissible to declare the function, either explicitly or implicitly, and use it without including its associated header. If a function that accepts a variable number of arguments is not declared (explicitly or by including its associated header), the behaviour is undefined.

1.4 Relationship to Standards

This X/Open CD-ROM specification specifies a subset of ISO 9660 for the volume and file structure of CD-ROM discs. The physical characteristics and the recorded format of the CD-ROM discs conform to the specification in ISO/IEC 10149. This X/Open CD-ROM specification also references ISO 9660 requirements for conformant systems. See Chapter 4 on page 25 for further information.

1.5 Format of Entries

The entries in Chapter 5 on page 27 and Chapter 6 on page 41 are based on two common formats.

1.5.1 Format of Commands

The description of each command in Chapter 5 is divided into a number of subsections. The information that each of these subsections contains is as follows.

NAME
The NAME section gives the name or names of the utility described by the entry and briefly states its purpose.

SYNOPSIS
The SYNOPSIS section summarises the syntax of the calling sequence for the utility, including options, option-arguments and operands.

DESCRIPTION
The DESCRIPTION section describes the actions of the utility.

OPTIONS
The OPTIONS section describes the utility options and option-arguments, and how they modify the actions of the utility.

OPERANDS
The OPERANDS section describes the utility operands, and how they affect the actions of the utility.

STDIN
The STDIN section describes the standard input of the utility.

INPUT FILES
The INPUT FILES section describes the files, other than the standard input, used as input by the utility. It includes files named as operands and option-arguments as well as other files that are referred to, such as startup / initialisation files, databases and so on.

ENVIRONMENT VARIABLES
The ENVIRONMENT VARIABLES section lists what variables affect the utility's execution.

STDOUT
The STDOUT section describes the standard output of the utility.

STDERR
The STDERR section describes the standard error output of the utility.

OUTPUT FILES
The OUTPUT FILES section describes the files created or modified by the utility.

EXIT STATUS
The EXIT STATUS section describes the values the utility will return to the calling program, or shell, and the conditions that cause these values to be returned. Usually, utilities return zero for successful completion and values greater than zero for various error conditions.

CONSEQUENCES OF ERRORS
The CONSEQUENCES OF ERRORS section describes the effects on the environment, file systems, process state and so on, when error conditions occur. It does not describe error messages produced or exit status values used.

APPLICATION USAGE
The APPLICATION USAGE section gives advice to the application programmer or user about the way that the utility should be used.

EXAMPLES
The EXAMPLES section gives example(s) of usage, where appropriate.

FUTURE DIRECTIONS
The FUTURE DIRECTIONS section is a guide to current thinking; there is not necessarily a commitment to implement all of these future directions in their entirety.

SEE ALSO
The SEE ALSO section lists related entries.

CHANGE HISTORY
The CHANGE HISTORY section shows the derivation of the description.

1.5.2 Format of Library Functions

The description of each function in Chapter 6 on page 41 is divided into a number of subsections. The information that each of these subsections contains is as follows.

NAME
The NAME section gives the name(s) of the entry and briefly states its purpose.

SYNOPSIS
The SYNOPSIS section summarises the use of the entry being described. If it is necessary to include a header to use this interface, the names of such headers will be shown, for example **#include <stdio.h>**.

DESCRIPTION
The DESCRIPTION section discusses the subject.

RETURN VALUE
The RETURN VALUE section indicates the return value, if any.

ERRORS
The ERRORS section gives the symbolic names of the values returned in the global variable *errno* if an error occurs.

EXAMPLES
The EXAMPLES section gives examples of usage, where appropriate.

APPLICATION USAGE
The APPLICATION USAGE section gives warnings and advice to application writers about the entry.

FUTURE DIRECTIONS
The FUTURE DIRECTIONS section provides comments which should be used as a guide to current thinking; there is not necessarily a commitment to adopt these future directions.

SEE ALSO
The SEE ALSO section gives references to related information.

CHANGE HISTORY
The CHANGE HISTORY section shows the derivation of the entry and any changes that have been made to it.

The formal description consists only of NAME, SYNOPSIS, DESCRIPTION, RETURN VALUE and ERRORS sections.

1.5.3 Typographical Conventions

The following typographical conventions are used throughout this document:

- **Boldface** strings are literals and are to be typed just as they appear.
- *Italic* strings usually represent substitutable argument prototypes and the names of entries found elsewhere.
- Names in upper case surrounded by braces, for example, {CONST}, represent constants which may be declared in appropriate headers by means of the C **#define** construct. For portability, only symbolic names should be used, never the value that a particular implementation may happen to use. The values of most of these constants are defined in **<limits.h>** or **<unistd.h>**.
- The notation *name*( ) indicates a function. Names without parentheses may be either variable names, or names of commands or utilities.
- The notation **<file.h>** indicates a header, see Chapter 7 on page 59.
- The notation [EABCD] is the name of the error value; the value of the EABCD error is defined in **<errno.h>**.
- Ellipses, ‘‘. . .’’, are used to show that additional arguments are optional.

1.5.4 Terminology

The following terms are used in this specification:

implementation-defined
The feature is not consistent across all implementations, and each implementation will provide documentation of its behaviour.

may
With respect to implementations, the feature is optional. Applications should not rely on the existence of the feature.

undefined
A feature is undefined if this specification imposes no portability requirements on applications for erroneous program construct or erroneous data. Implementations may specify the result of using the feature, but such specifications are not guaranteed to be consistent across all implementations.

unspecified
A feature is unspecified if this specification imposes no portability requirements on applications for correct program constructs or correct data. Implementations may specify the results of using the feature, but such specifications are not guaranteed to be consistent across all implementations.

will
The feature is required to be implemented and applications can rely on its existence.

1.6 Future Direction

XCDR will be extended to better allow for the form of software distribution where the applications are directly executed from the CD-ROM. This will include the ability to represent files and directories on CD-ROM in the XSI file hierarchy without restrictions on access permissions (apart from the write permission) and without other restrictions than applicable for XSI on the character set used for the names. This will possibly include set-user-ID, set-group-ID and symbolic links.

Future versions of XCDR will support the semantics of multi-volume sets more transparently.

Chapter 2

Functionality

2.1 Introduction

This chapter gives an overview of the functionality of XCDR. In short, the XCDR acts as a receiving system with implementation level 1 conformance to ISO 9660, with restrictions as mentioned below.

XCDR offers an API (either via a dedicated library interface or via a transparent use of XSI system interfaces) to access information on CD-ROMs that are conformant to ISO 9660 (interchange levels 1, 2 or 3) with restrictions as mentioned below. Restrictions comprise restrictions on the support of multi-volume sets and on the length of File Identifiers. See Chapter 4 on page 25 for detailed information on the relation to ISO 9660.

2.2 Making the CD-ROM Accessible

It is the responsibility of the system administrator to mount the CD-ROM in the XSI file hierarchy after the CD-ROM has been put in the drive. Before removing the CD-ROM, the system administrator has to unmount it. The commands or library functions to use for both are implementation-defined.

2.3 Accessing Files and Directories on the CD-ROM

After the CD-ROM has been mounted, it appears as a normal read-only file system in the XSI file hierarchy. This means that the usual system interfaces and commands which are also used for other file systems can be used for accessing the information. Examples of applicable commands are *cat*, *ls* and *cd*. The ownership (User ID and Group ID) and the access permissions of the files and directories will appear as specified on the CD-ROM in the Extended Attribute Record (XAR). See Section 3.2 on page 19 for the exact mapping of CD-ROM file attributes.

By default, XCDR interprets the execute attributes of directories to indicate search permission, as in the XSI file hierarchy.

Those files and directories for which the accessed CD-ROM does not hold a final XAR, or those for which the final XAR is restricted (whether the XAR is restricted or not can be determined from the CD-ROM platter, see Chapter 7 on page 67), the default ownership and access permissions are as specified below:

User ID
: Implementation-defined.

Group ID
: Implementation-defined.

File access permission
: {S_IRUSR} | {S_IXUSR} | {S_IRGRP} | {S_IXGRP} | {S_IROTH} | {S_IXOTH}.

Directory access permission
: As ‘‘file access permission’’.

The default for a whole CD-ROM can be changed to another value; see below.

2.4 System Administrator Actions after Mounting

After mounting the CD-ROM, the system administrator will normally execute a command to change the appearance of the names and attributes on the CD-ROM to the way best fitting the system. This change of appearance is in four areas:

- changing default ownership and access permissions
- changing the default search permission on directories
- mapping User ID and Group ID
- changing the names (character set conversion and version number suppression). The functions are globally described below. See Chapter 5 on page 27.

In future editions of the XPG, when the *mount* command is specified, it is envisioned that some of the functions can be performed with options on the **mount** command as well. The options on a future *mount* command are likely to be close to the ones on *cdmntsuppl* now. Note that the command *cdmntsuppl* is to be seen as a supplement to the *mount* command and thus is intended only to be used by the system administrator directly after the mounting of the CD-ROM, before any access to the CD-ROM is done.

2.4.1 Changing Default Ownership and Access Permissions

The default ownership and access permissions for those files and directories that do not have a final XAR, or do have a restricted final XAR, can be changed from the default value stated in Section 2.3 on page 11, by using the command *cdmntsuppl*, or the equivalent library function. The change will apply to the whole CD-ROM which is mounted on the mount-point given by the non-optional argument (**/mnt** in the example below). For example,

```
cdmntsuppl -u 4 /mnt
```

will change the default User ID of all files on the total CD-ROM identified by **/mnt** to the value 4.

Note that the attributes of files that do have a (non-restricted) final XAR will not be affected.

2.4.2 Setting Search Permission for Directories

By default, the execute bit on directories indicates search permission, as for other file systems in the XSI file hierarchy. ISO 9660 does not define the semantics of the execute bit on directories. To be able to read discs which were not specifically targetted for XCDR, the command *cdmntsuppl* can be used with option **–s** to grant search permission on those directories that have read permission alone.

2.4.3 Mapping User and Group ID

The User ID and Group ID of files and directories as recorded in the Extended Attribute Record on the CD-ROM may conflict with the User IDs and Group IDs used in the system where the CD-ROM is mounted. If that is the case, the command *cdmntsuppl* (with options **–U** and/or **–G**) can be used to let every occurrence of a certain User ID or Group ID on the CD-ROM be represented as a specific other User ID or Group ID.

If a system imposes a maximum value on the number of mappings, this will be defined via the symbolic names {CD_MAXUMAP} and {CD_MAXGMAP} in **<sys/cdrom.h>**. At least 50 User ID mappings and at least 50 Group ID mappings will be supported by XCDR.

User IDs and Group IDs for files and directories without a final XAR or with a restricted final XAR will not be affected. The change of the default values for those files and directories is discussed above in Section 2.4.1.

Example:

```
cdmntsuppl –U mapfile
```

where

mapfile is a file specifying the mapping as follows:

```
1000:208
1001:224
1009:john
```

will map the User ID of all files and directories with User ID = 1000 on the CD-ROM to a User ID of 208 in the XSI file hierarchy, those having User ID = 1001 on the CD-ROM to 224 and those with a User ID = 1009 to the User ID of the user *john*.

2.4.4 Filenames

ISO 9660 limits the filenames on a CD-ROM to a certain character set. See Annex A, Clauses 7.5, 7.6 and Table 14 of ISO 9660. One of the limitations is the exclusion of lower-case characters, and that exactly one dot must appear in the filename. It is common practice in X/Open-compliant systems to use lower-case characters by default.

Using the command *cdmntsuppl* –**l** or the equivalent library function, all upper-case names on the CD-ROM can be forced to appear in lower case in the XSI file hierarchy, and filenames with a trailing dot on CD-ROM will appear without this dot.

Another property of ISO 9660 is that the filename has a mandatory semicolon and version number at the end. This is not according to common practice in X/Open-compliant systems. The command *cdmntsuppl* –**m**, or the equivalent library function, can be used to suppress the semicolon and version number. Note that when *cdmntsuppl* –**l** and/or –**m** are in effect, the behaviour of *readdir*( ) and lookup functions are still consistent. When obtaining a filename via *readdir*( ), an *open*( ) using this filename must open the same file.

When filename conversion is applied, the files will behave (for reading, searching and matching) as if they were recorded on the CD-ROM with the converted name.

In case the option is used to suppress the version number, and multiple files in the same directory which are only different in their version number are recorded on the CD-ROM, only the one with the highest version number will be made visible in directories when accessed via *readdir*( ). Files with other than the highest version number are not accessible via the XSI interfaces, except possibly in conjunction with the use of implementation-defined interfaces.

2.5 Obtaining CD-ROM-specific Information

The CD-ROM contains information which is specific to it and cannot be obtained by standard XSI system interfaces. This can be information that relates to the total disc or to a single file or directory. Examples of the first are Volume Name, Volume Expiration Date and Publisher Identifier. Examples of the latter are the Application Use Field and the System Use Field.

Using the command *cdvd*, or the equivalent library function, all additional information on the CD-ROM level can be accessed (located in the Primary Volume Descriptor). Using the commands *cdxar*, *cddrec* and *cdptrec* or the equivalent library functions, all additional information on a file or directory can be accessed (located in an XAR, in a Directory Record or in a Path Table Record).

2.6 Multi-volume CD-ROM

The transparent handling of the semantics of multi-volume sets need not be supported by XCDR implementations. Nevertheless, it will be possible to mount each member of a multi-volume set separately and use it as ‘‘single’’ disc.

When an XCDR implementation does not transparently support the semantics of multi-volume sets, it will return an error [ENODEV] when an application attempts to *open*( ) or *stat*( ) a file of which one or more File Sections (as defined by ISO 9660) are located at another member of the Volume Set.

The system interface *readdir*( ) will, on repetitive invocations, return the names of all files in the directory, also the names of those files of which one or more File Sections are located on another member of the Volume Set. Note that when a file consists of multiple File Sections, *readdir*( ) will only return the filename once.

Chapter 3

Accessing CD-ROM Information

This chapter describes how the information in the various fields of the CD-ROM can be accessed. Descriptors, or fields of them, on the CD-ROM not mentioned in this list may be made available to the user by an XCDR implementation.

3.1 Conversion of File Identifiers and Directory Identifiers

3.1.1 File Identifiers

The conversion of File Identifiers is as follows:

- As a default, File Identifiers are represented as *filename.filename_extension;version* in the XSI file hierarchy.
- When the command *cdmntsuppl* −**l**, or the equivalent library function, has been applied, File Identifiers are represented as *filename.filename_extension;version* in the file hierarchy, except that the characters in File Name and File Name Extension are converted to lower case, and when the length of File Name Extension is zero, the SEPARATOR 1 (.) is not present.
- When command *cdmntsuppl* −**m**, or the equivalent library function, has been applied, File Identifiers are represented as *filename.filename_extension* in the file hierarchy.
- When command *cdmntsuppl* −**l** −**m** has been applied, the effects of both the −**l** and −**m** options will be applicable.

When the total length of the name (using the rules above) that should be represented in the XSI file hierarchy exceeds 14 bytes, the representation (whether or not, and if so how) of that file in the XSI file hierarchy is implementation-defined. An XCDR implementation needs only to support names to a maximum of 14 bytes. See Chapter 4 on page 25 for the background of this limitation.

The following table gives examples of the conversion of File Identifiers:

	Representation in XSI File Hierarchy			
On CD-ROM	**Default**	***cdmntsuppl* −m**	***cdmntsuppl* −l**	***cdmntsuppl* −l −m**
A.B;1	A.B;1	A.B	a.b;1	a.b
A.;1	A.;1	A.	a;1	a
.B;1	.B;1	.B	.b;1	.b
ABCDEFGHIJKLMN.;1	1.	1.	1.	abcdefghijklmn
ABCDEFGHIJKLM.N;1	1.	1.	1.	1.
ABCDEFGHIJKL.M;1	1.	ABCDEFGHIJKL.M	1.	abcdefghijkl.m
ABCDEFGHIJ.K;1	ABCDEFGHIJ.K;1	ABCDEFGHIJ.K	abcdefghij.k;1	abcdefghij.k
A;1	2. (dot required)			
A.	2. (version required)			
A.;0	2. (invalid version number)			
A.;45678	2. (version number too large)			
A.B.C;1	2. (only one dot allowed)			

1. Implementation-defined, resulting name would comprise more than 14 bytes.
2. Undefined, invalid name according to ISO 9660.

3.1.2 Directory Identifiers

Directory Identifiers are represented in the following way:

- As a default, they are represented as recorded on the CD-ROM.
- When the command *cdmntsuppl* **–l**, or the equivalent library function, has been applied, Directory Identifiers are represented as recorded on the CD-ROM except that the characters in the Directory Identifier are converted from upper case to lower case.

When the Directory Identifier Length on the CD-ROM exceeds 14 bytes, the representation of that directory in the XSI file hierarchy is implementation-defined. An XCDR implementation needs only to support names to a maximum of 14 bytes. See Chapter 4 on page 25 for the background of that limitation. Note that according to ISO 9660, Directory Identifiers do not have a dot or semicolon separator, neither do they have extensions nor version numbers.

3.2 Mapping Directory Record and XAR Fields

3.2.1 Directory Record

See Table 8 of ISO 9660. All the fields mentioned below can be obtained (on File Section basis) via the library function *cd_drec*( ).

ISO 9660	XSI
Length of Directory Record	Transparently handled.
Extended Attribute Record Length	Transparently handled.
Location of Extent	Transparently handled.
Data Length	The field *st_size* in the structure *stat* in the *stat*( ) and *fstat*( ) system interfaces will show the sum of all Data Length fields of all Directory Records associated with the same file or directory.
Recording date and time	The fields *st_ctime*, *st_mtime* and *st_atime* in the structure *stat* in the *stat*( ) and *fstat*( ) system interfaces are filled using the Recording Date and Time field recorded in the final Directory Record for the file when there is not a final XAR, otherwise the field is ignored.[1.]
File Flags	See below.
File Unit Size, Interleave Gap Size	Handled transparently.
Volume Sequence Number	Handled transparently by XCDR when all File Sections of the file are on a mounted CD-ROM.
Length of File Identifier	Number of bytes that will appear in the the field *d_name* of the structure *dirent* in the system interface *readdir*( ). Note that this value is subject to conversion by the command *cdmntsuppl* −**l** and *cdmntsuppl* −**m** and the equivalent library functions. Note that the value can also be obtained with the library function *cd_drec*( ), in which case it will not be influenced by the command *cdmntsuppl*.
File Identifier	Contents of the field *d_name* of the structure *dirent* in the system interface *readdir*( ). Note that the contents are subject to conversion by the command *cdmntsuppl* −**l** and *cdmntsuppl* −**m** and equivalent library functions. Note that the File Identifier can also be obtained with library function *cd_drec*( ), in which case the value will not be influenced by the command *cdmntsuppl*.
System Use	Will be ignored.

1. Times which cannot be represented in **time_t** will be represented by 0 (for times before 1 January 1970 UTC) and by the highest possible value of **time_t** (for times too far in the future to be represented).

3.2.2 File Flags

See Table 10 of ISO 9660. All the fields mentioned below can be obtained (on File Section basis) via the library function *cd_drec*( ).

ISO 9660	XSI
Existence	The field is either ignored or the file or directory to which it applies is not made visible nor accessible. Which of these two options is in effect is implementation-defined.
Directory	The field *st_mode* in the structure *stat* in the *stat*( ) and *fstat*( ) system interface has {S_IFDIR} set when the Directory bit is set to ONE, {S_IFREG} is set when the Directory bit is set to ZERO.
Associated File	Associated files will not be made visible nor accessible, except possibly via the use of implementation-defined mechanisms.
Record	Will be ignored.
Protection	Value in the non-final Directory Record is ignored. If set to ONE on the final Directory Record, XAR owner, group and permission fields are honoured, otherwise defaults are used (see Section 2.3 on page 11).
Multi-Extent	Multiple extent files are handled transparently and appear as regular files with data from all extents. When one or more extents are located on another volume, an XCDR implementation will either make the total file transparently available, or return an error [ENODEV] on *open*( ) or *stat*( ). See also Section 2.6 on page 15 for this restriction on multi-volume set handling.

3.2.3 XAR Fields

See Tables 12 and 13 of ISO 9660. All the fields mentioned below can be obtained via the library function *cd_xar*( ).

ISO 9660	XSI
Owner Identification	When the XAR is a restricted XAR, the field is ignored. Otherwise the field *st_uid* in the structure *stat* in the *stat*( ) or *fstat*( ) system interfaces. 1. 2.
Group Identification	When the XAR is a restricted XAR, the field ignored. Otherwise the field *st_gid* in the structure *stat* in the *stat*( ) or *fstat*( ) system interfaces. 1. 2.
Permissions: 1.	
When the XAR is a restricted XAR, the field is ignored. Otherwise:	
Read / Execute owner of system class	Will be ignored.
	Indication set in the field *st_mode* in the structure *stat* in the *stat*( ) or *fstat*( ) system interface:
Read owner	{S_IRUSR} 3.
Execute owner	{S_IXUSR}
Read group	{S_IRGRP} 3.
Execute group	{S_IXGRP}
Read any user	{S_IROTH} 3.
Execute any user	{S_IXOTH}
File Creation Date and Time	*st_ctime* in the structure *stat* in the *stat*( ) and *fstat*( ) system interfaces. 4. 5.
File Modification Date and Time	*st_mtime* in the structure *stat* in the *stat*( ) and *fstat*( ) system interfaces. *st_atime* will be equal to *st_mtime*. 4. 5.
File Expiration Date and Time	Will be ignored.
File Effective Date and Time	Will be ignored.
Record Format / Attr. / Length	Will be ignored.
System Identifier	Will be ignored.
System Use	Will be ignored.
Application Use	Will be ignored.
Escape Sequence	Will be ignored.

1. This XAR field is ignored when the XAR is not a final XAR, or when the XAR is a restricted final XAR.

2. A value of zero (no owner) is mapped to an implementation-defined value in the XSI file hierarchy, because, according to ISO 9660, the Protection File Flag in the Directory Record will be set to ZERO in this case. This default value is subject to the command *cdmntsuppl* –**u** and *cdmntsuppl* –**g** and the equivalent library functions.

   When Owner Identification or Group Identification hold a value which cannot be represented by the data types **uid_t** and **gid_t**, respectively, the value will be mapped to the default value. Note that an XCDR implementation will be capable of mapping any legal value for Owner Identification and Group Identification to a value which is representable by **uid_t** and **gid_t**, respectively, (done by *cdmntsuppl* –**U** and *cdmntsuppl* –**G**, respectively, or the equivalent library functions). When the value has not been mapped by *cdmntsuppl* –**U** or *cdmntsuppl* –**G**, the value will be subject to mapping by *cdmntsuppl* –**u** and *cdmntsuppl* –**g** (the order of command execution has no influence).

3. When the command *cdmntsuppl* –**s**, or the equivalent library function, has been applied, execute permission will also be set when the XAR is the XAR of a directory (then Read owner will also set {S_IXUSR}, Read group also {S_IXGRP} and Read any user also {S_IXOTH}).

4. This XAR field is ignored when the XAR is not a final XAR.
5. Times which cannot be represented in **time_t** will be represented by 0 (for times before 1 January 1970 UTC) and by the highest possible value of **time_t** (for times too far in the future to be represented).

3.3 Access to Miscellaneous Fields and to File Contents

ISO 9660	XSI
File contents	Handled transparently by XSI system interfaces. All XSI system interfaces are supported, provided that they accept the read-only limitation of CD-ROM.
Primary Volume Descriptor	Library function *cd_vd*( ).
Path Table Record	Library function *cd_ptrec*( ).

Chapter 4

Conformance Levels to Standards

XCDR will support CD-ROM discs which are conformant to the following:

- The physical format of the platter must be according ISO / IEC 10149. This means that the standard for recording, as mentioned in ISO 9660, is for XCDR ISO / IEC 10149.
- The volume and file structure of the CD-ROM must be according to ISO 9660. XCDR supports interchange levels 1, 2 and 3 as defined by Clause 10 of ISO 9660, with the two restrictions mentioned below.

XCDR conforms to the requirements for a receiving system, implementation level 1, as defined by Clause 13 of ISO 9660, with the two restrictions mentioned below.

Implementations of XCDR may deviate from ISO 9660 in the following two ways:

- An XCDR implementation may restrict the length of the Files Identifiers and Directory Identifiers as they will appear in the XSI file hierarchy to a limit of at least 14 bytes.

  The reason for this limitation is the fact that 14 bytes is the lowest maximum length of filenames that an X / Open-compliant system must support. See the definitions of the symbolic names {NAME_MAX} and {_POSIX_NAME_MAX} in XSH, Issue 3, Volume 2 (**<limits.h>**).

  The length that Directory Identifiers have when represented in the XSI file hierarchy is equal to the Directory Identifier length of ISO 9660.

  By default, the length that File Identifiers have when represented in the XSI file hierarchy is equal to the File Identifier length as defined in Section 7.5.2 of ISO 9660. When the *cdmntsuppl* command, or the equivalent library function, has been applied, the length as represented in the XSI file hierarchy will be shorter:

  — When *cdmntsuppl* –**l** has been applied and the length of the File Name Extension is zero, then 1 is subtracted.

  — When *cdmntsuppl* –**m** has been used then the number of digits in the File Version Number plus 1 is subtracted.

  — When both of the above conditions hold, both values are subtracted.

  Producers of CD-ROMs to be read by XCDR conforming implementations should consider and document whether *cdmntsuppl* –**l**, *cdmntsuppl* –**m**, none, or both of them have to be applied when actually using the CD-ROM. If interested in maximum portability among X / Open-compliant systems, they should make sure that files and directories have identifiers that will not result in more than 14 bytes when represented in the XSI file hierarchy after the use, or not, of *cdmntsuppl* with the options required for that CD-ROM. See also Appendix A on page 69.

- XCDR is not required to provide access to files that consist of multiple File Sections which are located on different volumes. Apart from this, an XCDR implementation is not required to provide transparant access to those files which are contained in a directory which is accessible to the application but which files are actually located on another volume. The application will have access to the file when the appropriate volume has been mounted. See also Section 2.6 on page 15.

The requirements for a receiving system, implementation level 1, as defined in Clause 13 of ISO 9660, are met by providing access to the required information in the way specified in Chapter 3 on page 17.

Chapter 5

Commands

5.1 Introduction

This section defines the commands that can be used by a user to access information on the CD-ROM, or by a system administrator to set and get administrative features. All the CD-ROM-specific commands are based on the system interfaces defined in Chapter 6 on page 41.

The CD-ROM-specific commands supported by XCDR are as follows:

cdvd	Read Volume Descriptor from CD-ROM.
cdxar	Read Extended Attribute Record from CD-ROM.
cddrec	Read Directory Record from CD-ROM directory.
cdptrec	Read Path Table Record from CD-ROM Path Table.
cdmntsuppl	Set and get administrative CD-ROM features.

5.2 Definition of CD-ROM-specific User Commands

This section presents the manual pages which describe the CD-ROM-related user commands in detail.

NAME

cdvd — read Volume Descriptor from CD-ROM

SYNOPSIS

```
cdvd  [−b]  file
```

DESCRIPTION

This command is used to read the Primary Volume Descriptor from a CD-ROM and to list the contents on the standard output.

OPTIONS

The following option is available:

−b With this option, the entirety of the Primary Volume Descriptor is copied from CD-ROM to standard output in binary format.

OPERANDS

The operand *file* is either a file or directory within the CD-ROM file hierarchy of the mounted CD-ROM file system, or the name of the block special file for a CD-ROM file system.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

LC_TIME determines the format and contents of date and time strings. If *LC_TIME* is not set in the environment, or if it is set to the empty string, the value of *LANG* will be used as a default. If *LANG* is not set, or it is set to the empty string, the corresponding value from the implementation-specific default locale will be used. If *LC_TIME* or *LANG* contain an invalid setting, the utility will behave as if none of the variables had been defined.

STDOUT

The output is formatted in the form of a table which contains the name of each field of the Primary Volume Descriptor and the corresponding contents of the entry as recorded on the CD-ROM. The Application Use field is not listed because it may contain non-printable characters.

If the option **−b** is applied, the contents of the Primary Volume Descriptor are copied to the standard output in binary format as it is on the CD-ROM. This includes the Application Use field.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 *file* not found, or *file* is not within a CD-ROM file hierarchy, or *file* is not a block special file for a CD-ROM file system, or access permission denied.

2 *file* is a block special file and a read error occurred, or the CD-ROM is not recorded according to the ISO 9660 standard.

CONSEQUENCES OF ERRORS

None.

APPLICATION USAGE

The user must have read permission for *file* to execute the command successfully.

EXAMPLES

None.

FUTURE DIRECTIONS

This command may be extended for further types of Volume Descriptors (for example, Supplementary Volume Descriptor, Volume Partition Descriptor). For this purpose, additional options will be introduced.

SEE ALSO

None.

CHANGE HISTORY

None.

NAME

cdxar — read Extended Attribute Record from CD-ROM

SYNOPSIS

```
cdxar  [–s number]  [–b]  file
```

DESCRIPTION

This command is used to access the Extended Attribute Record associated with a File Section of a file or directory and to list its contents on the standard output. If the command is used without the option –**b**, only the fixed part of the XAR is copied from the CD-ROM; this does not include the Application Use field and the Escape Sequences field.

OPTIONS

The following options are available:

–**s** *number*
This option specifies the File Section for which the XAR shall be read. The numbering starts with one. If this option is omitted the last File Section of that file is assumed.

–**b** With this option the entirety of the XAR is copied from the CD-ROM to the standard output in binary format.

OPERANDS

The operand *file* is the name of any file or directory within the CD-ROM file hierarchy.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

LC_TIME determines the format and contents of date and time strings. If *LC_TIME* is not set in the environment, or if it is set to the empty string, the value of *LANG* will be used as a default. If *LANG* is not set, or it is set to the empty string, the corresponding value from the implementation-specific default locale will be used. If *LC_TIME* or *LANG* contain an invalid setting, the utility will behave as if none of the variables had been defined.

STDOUT

The output is formatted in the form of a table which contains the name of each field of the XAR and the corresponding contents of the entry as recorded on the CD-ROM. The Application Use field and the Escape Sequences field are not listed because they may contain non-printable characters.

If the option –**b** is applied, the contents of the full XAR is written to the standard output in binary format as it is on the CD-ROM.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 *file* not found, or *file* is not file or directory within a CD-ROM file hierarchy, or access permission denied.

2 File Section indicated by –**s** does not exist.

3 File Section indicated by –**s** has no XAR.

4 The File Section of *file* indicated by –**s** is not on the currently mounted CD-ROM.

CONSEQUENCES OF ERRORS

None.

APPLICATION USAGE

The user must have read permission for *file* to execute the command successfully.

EXAMPLES

None.

FUTURE DIRECTIONS

None.

SEE ALSO

None.

CHANGE HISTORY

None.

NAME

cddrec — read Directory Record from CD-ROM directory

SYNOPSIS

```
cddrec  [−s number]  [−b]  file
```

DESCRIPTION

This command is used to access the Directory Record associated with a CD-ROM file or directory and to list its contents on standard output.

If the command is used without the option −**b**, only the fixed part of the Directory Record is copied from CD-ROM; this does not include the System Use field.

OPTIONS

−**s** *number*
This option specifies the File Section for which the Directory Record should be read. The numbering starts with one. If the option is omitted the last File Section of that file is assumed.

−b With this option the entirety of the Directory Record is copied from CD-ROM and written to standard output in binary format.

OPERANDS

The operand *file* is the name of any file or directory within the CD-ROM file hierarchy.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

LC_TIME determines the format and contents of date and time strings. If *LC_TIME* is not set in the environment, or if it is set to the empty string, the value of *LANG* will be used as a default. If *LANG* is not set, or it is set to the empty string, the corresponding value from the implementation-specific default locale will be used. If *LC_TIME* or *LANG* contain an invalid setting, the utility will behave as if none of the variables had been defined.

STDOUT

The output is formatted in the form of a table which contains the name of each field of the Directory Record and the corresponding contents of the entry as recorded on the CD-ROM. The System Use field is not listed because it may contain non-printable characters. If the option −**b** is applied, the content of the Directory Record is written to standard output in binary format as it is on the CD-ROM. The System Use field is included.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 *file* not found, or not within the CD-ROM file hierarchy, or access permission denied.

2 File Section indicated by −**s** does not exist.

CONSEQUENCES OF ERRORS

None.

APPLICATION USAGE

The user must have read permission for *file* to execute the command successfully.

EXAMPLES

None.

FUTURE DIRECTIONS

None.

SEE ALSO

None.

CHANGE HISTORY

None.

NAME
cdptrec — read Path Table Record from CD-ROM Path Table

SYNOPSIS
```
cdptrec  [−b]  directory
```

DESCRIPTION
This command is used to access a Path Table Record associated with a CD-ROM directory and lists its contents on the standard output.

OPTIONS
−b With this option the Path Table Record is copied from the CD-ROM to standard output in binary format.

OPERANDS
The operand *directory* is the name of any directory within the CD-ROM file hierarchy.

STDIN
Not used.

INPUT FILES
None.

ENVIRONMENT VARIABLES
No environment variables affect the execution of *cdptrec*.

STDOUT
The output is formatted in the form of a table which contains the name of each field of the Path Table Record and the corresponding contents of the entry as recorded on the CD-ROM.

STDERR
Used only for diagnostic messages.

OUTPUT FILES
None.

EXIT STATUS
The following exit values are returned:

0 Successful completion.

1 *directory* not found, or is not within a CD-ROM file hierarchy, or access permission denied.

CONSEQUENCES OF ERRORS
None.

APPLICATION USAGE
The user must have read permission for *directory* to execute the command successfully.

EXAMPLES
None.

FUTURE DIRECTIONS
None.

SEE ALSO
None.

CHANGE HISTORY

None.

5.3 Definition of CD-ROM-specific Administrator Command

This section presents the manual page that describes the CD-ROM-related administrator command in detail.

NAME

cdmntsuppl — set and get administrative CD-ROM features

SYNOPSIS

```
cdmntsuppl   [−u owner]  [−g group]  [−F mode]  [−D mode] [−U umfile]
             [−G gmfile]  [−c]  [−l]  [−m]  [−x]  [−s]  mount-point
```

DESCRIPTION

This command sets up administrative CD-ROM features, such as default ownership and access permissions, mapping of user and group identifications, conversion of filenames and setting of execute permissions for directories. Each combination of options may be used, including no option at all. If *cdmntsuppl* is executed without any option, it lists the current settings.

OPTIONS

The options −**u**, −**g**, −**F** and −**D** may be used to set the default owner, group and/or default access permissions to be associated with those files and directories in a CD-ROM file system that do have a restricted final XAR, or no final XAR. This is useful in situations where files are supplied with a restricted final XAR, or no final XAR, but access specific to one user or group (other than the default at mount time) is required, or where the default access permissions (being read and execute permission for user, group and others) for files and directories are inappropriate.

−**u** *owner*

The operand *owner* may be either a decimal User ID or a login name found in the User Database.

−**g** *group*

The operand *group* may be either a decimal Group ID or a group name found in the Group Database.

−**F** *mode*

This option is used to set the default permissions for files. The permissions are changed according to *mode*, which may be absolute or symbolic. An absolute *mode* is a four-octal-digit number constructed from the logical-or (sum) of the following modes:

0400	read by owner
0100	execute by owner
0040	read by group
0010	execute by group
0004	read by others
0001	execute by others

A symbolic *mode* has the form:

[who] op [permission]

The *who* part is a combination of the letters **u** (user), **g** (group) and **o** (other). The letter **a** stands for **ugo**, the default if *who* is omitted.

The argument *op* can be **+**, to add *permission* to the file's mode, –, to take away *permission*, or **=**, to assign *permission* absolutely (all other bits will be reset).

The argument *permission* is any combination of the letters **r** (read) and **x** (execute); **u**, **g** or **o** indicate that permission is to be taken from the current mode. Omitting *permission* is only useful with **=** to take away all permissions.

−**D** *mode*

This option is used to set the default permissions for directories in the same fashion as for the option −**F**. Execute permission will be interpreted as search permission.

The options –**U** and –**G** may be used in the case where a CD-ROM file system has been supplied with undesirable User / Group IDs associated with files and directories. These undesirable IDs may be transformed to more appropriate numbers. Only files and directories with an unrestricted final XAR are subject to this mapping (see [–**u** *owner*] and [–**g** *group*] above for other files and directories). An owner or group identification of zero is not permitted by ISO 9660 to appear in an unrestricted XAR, and thus the behaviour is undefined if a mapping is given for the value zero.

–**U** *umfile*, –**G** *gmfile*
These options need a file (*umfile* / *gmfile*) as operand which must contain pairs. Each pair must be provided using the following syntax: value as on CD-ROM, colon, numeric ID or User / Group name as found in User / Group Database. Pairs must be separated by a newline character. The maximum number of mappings is defined in the header file **<sys/cdrom.h>**.

The –**c**, –**l** and –**m** options establish / de-establish a specific name conversion of File / Directory Identifiers on a CD-ROM. Name conversion is a desirable feature to represent File / Directory Identifiers in a way that is in accordance to common practice in X / Open-compliant systems. See Section 2.4.4 on page 13 and Section 3.1.1 on page 17 for further information. The options –**l** and –**m** may be used in conjunction. The option –**c** may not be used in conjunction with the options –**l** or –**m**.

–**c** This option causes names to be handled as recorded on CD-ROM, that is, no conversion takes place. This matches the default after initialisation.

–**l** Upper-case characters in Identifiers are converted to lower case. If the File Identifier contains no File Name Extension, the SEPARATOR 1 (.) is not represented. The effect of the –**m** option is unchanged.

–**m** The Version Number and the SEPARATOR 2 (;) of a File Identifier is not represented. The effect of the –**l** option is unchanged.

The –**x** and –**s** options determine the setting of the execute (search) permission bits for those directories in the CD-ROM file hierarchy that have a non-restricted final XAR. See Section 2.4.2 on page 12 for further information.

–**x** The execute permission bits for directories within the CD-ROM file hierarchy are set as provided in the Permissions field in the XAR of that directory.

–**s** Each execute permission bit for a directory in the XSI file hierarchy is set to the inclusive OR of the corresponding read and execute bits in the XAR of that directory on the CD-ROM.

OPERANDS

The operand *mount-point* is the name of the mount-point of the CD-ROM file system.

STDIN

Not used.

INPUT FILES

The input files are text files.

ENVIRONMENT VARIABLES

No environment variables affect the execution of *cdmntsuppl*. Note that *LC_CTYPE* will not be used in filename conversion.

STDOUT

If no options are used, the current settings are listed on the standard output. In the case of setting features, the new setting is listed if the command is completed successfully.

STDERR

Used only for diagnostic messages.

OUTPUT FILES

None.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 *mount-point* not found, or is not mount-point of a CD-ROM file system, or access permission denied.

2 No user with appropriate privilege.

3 Too many mappings.

4 Parameter error or bad format in a mapping file (*umfile* /*gmfile*).

CONSEQUENCES OF ERRORS

None.

APPLICATION USAGE

Only a user with appropriate privileges may change administrative CD-ROM features successfully. To read the current settings, the user must have read permission on the mount-point of that CD-ROM file system. In case of setting CD-ROM features, this command is intended to be used only directly after mounting the CD-ROM and before any access to the CD-ROM is done. If the command is applied for setting features when files or directories have already been opened, the effect of this command on these files and directories is undefined.

EXAMPLES

See Section 2.4 on page 12 and Section 3.1.1 on page 17 for some examples.

FUTURE DIRECTIONS

None.

SEE ALSO

None.

CHANGE HISTORY

None.

Chapter 6

Library Functions

6.1 Introduction

This chapter provides a description of the CD-ROM library **libcdrom**.

This library is designed for application writers to access CD-ROM-specific information and to set and get XCDR-specific features. These functions are used in the commands described in Chapter 5 on page 27.

The following library functions are available:

cd_pvd( ), *cd_cpvd*( )	Read Primary Volume Descriptor from CD-ROM.
cd_xar( ), *cd_cxar*( )	Read Extended Attribute Record for CD-ROM file/directory from CD-ROM.
cd_drec( ), *cd_cdrec*( )	Read Directory Record from CD-ROM directory.
cd_ptrec( ), *cd_cptrec*( )	Read Path Table Record from CD-ROM Path Table.
cd_type( )	Get identification of CD-ROM type.
cd_defs( )	Set and get default values for User/Group ID and file/directory permission.
cd_idmap( )	Set and get mappings of User/Group IDs.
cd_nmconv( )	Set and get CD-ROM filename conversion.

6.2 Definition of CD-ROM-specific Library Functions for Users

This section presents the manual pages for the CD-ROM library functions for users.

NAME

cd_pvd, cd_cpvd — read Primary Volume Descriptor from CD-ROM

SYNOPSIS

```
#include <sys/cdrom.h>

int  cd_pvd (path, pvd)
char  *path;
struct iso9660_pvd  *pvd;

int  cd_cpvd (path, pvd)
char  *path;
char  *pvd;
```

DESCRIPTION

The function *cd_pvd*( ) fills the *pvd* structure with the contents of the Primary Volume Descriptor from the CD-ROM. The declaration for **struct** *iso9660_pvd* is contained in **<sys/cdrom.h>**. The *path* argument points to a pathname naming a file or directory within a CD-ROM file hierarchy, or naming a block special file for a CD-ROM file system. To execute this function successfully the user must have read or execute permission on the file/directory pointed to by **path** or an appropriate privilege.

The function *cd_cpvd*( ) copies the complete Primary Volume Descriptor as recorded on the CD-ROM to the address *pvd*. The user must allocate {CD_PVDLEN} bytes for the Primary Volume Descriptor. {CD_PVDLEN} is contained in **<sys/cdrom.h>**.

RETURN VALUE

Upon successful completion, the functions return a value of zero.

In case of an error, –1 is returned and *errno* is set to indicate the error.

ERRORS

The functions will fail if:

[EACCES] Search permission is denied for a component of the *path* prefix.

Read permission for the file or directory pointed to by *path*, or read permission for the block special file pointed to by *path*, is denied.

[ENAMETOOLONG]
The length of the *path* string exceeds {PATH_MAX}, or a pathname component is longer than {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.

[ENOENT] A component of *path* does not exist, or the *path* argument points to an empty string.

[EINVAL] The named file is a block special file and the CD-ROM is not recorded according to ISO 9660.

The argument **path** points to a file/directory that is not within the CD-ROM file hierarchy.

[ENOTDIR] A component of the **path** prefix is not a directory.

[ENXIO] The named file is a block special file and the device associated with the special file does not exist.

The CD-ROM is not in the drive, or a read error occurred.

[EFAULT]	The address of *pvd* or *path* is invalid.
[EINTR]	A signal was caught during one of the functions.
[EMFILE]	{OPEN_MAX} file descriptors are currently open in the calling process.
[ENFILE]	The system file table is full.

SEE ALSO

<sys/cdrom.h>

NAME

cd_xar, cd_cxar — read Extended Attribute Record for CD-ROM file / directory from CD-ROM

SYNOPSIS

```
#include <sys/cdrom.h>

int  cd_xar (path, fsec, xar, applen, esclen)
char  *path;
int  fsec;
struct iso9660_xar  *xar;
int  applen, esclen;

int  cd_cxar (path, fsec, xar, xarlen)
char  *path;
int  fsec;
char  *xar;
int  xarlen;
```

DESCRIPTION

The *cd_xar*( ) function fills the *xar* structure with the contents of the XAR associated with a file or directory which is referred to by the argument *path*. The argument *fsec* specifies the File Section of that file. The numbering starts with one. If *fsec* is set to –1, the XAR of the last File Section of that file is assumed. The argument *path* points to a file or directory within the CD-ROM file hierarchy. The two arguments (*applen*, *esclen*) determine how many bytes will be copied to the addresses specified in the *iso9660_xar* structure by *app_use* and *esc_seq*. The total number of logical blocks of an XAR can be obtained by the *cd_drec*( ) function. The Logical Block Size in bytes can be obtained by the *cd_pvd*( ) function. The length of the fixed part of the XAR is given by {CD_XARFIXL}. The declaration for **struct** *iso9660_xar* and the definition of {CD_XARFIXL} are contained in **<sys/cdrom.h>**.

The *cd_cxar*( ) function copies the XAR as recorded on the CD-ROM to the address *xar*. With *xarlen* set appropriately, part of the XAR or the full XAR will be read.

RETURN VALUE

The *cd_xar*( ) function returns the number of bytes copied for the variable part of the XAR. The *cd_cxar*( ) function returns the number of bytes copied. In case of an error, –1 is returned and *errno* is set to indicate the error.

ERRORS

The functions will fail if:

[EACCES] Search permission is denied for a component of the *path* prefix, or read permission on the file or directory pointed to by *path* is denied.

[ENAMETOOLONG]
The length of the *path* string exceeds {PATH_MAX}, or a pathname component is longer than {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.

[ENOENT] A component of *path* does not exist, or the *path* argument points to an empty string.

The File Section indicated by *fsec* has no XAR.

[ENOTDIR] A component of the *path* prefix is not a directory.

[EFAULT] The address of *xar* or *path* is invalid.

[EINVAL]	The value of *fsec* or *xarlen* is invalid.
	The argument *path* points to a file / directory not within a CD-ROM file hierarchy.
[ENODEV]	The Volume containing the File Section indicated by *fsec* is not mounted.
[ENXIO]	The CD-ROM is not in the drive or a read error occurred.
[EINTR]	A signal was caught during one of the functions.
[EMFILE]	{OPEN_MAX} file descriptors are currently open in the calling process.
[ENFILE]	The system file table is full.

SEE ALSO

cd_drec( ), **<sys/cdrom.h>**

NAME

cd_drec, cd_cdrec — read Directory Record from CD-ROM directory

SYNOPSIS

```
#include <sys/cdrom.h>

int  cd_drec (path, fsec, drec)
char  *path;
int  fsec;
struct iso9660_drec  *drec;

int  cd_cdrec (path, fsec, drec)
char  *path;
int  fsec;
char  *drec;
```

DESCRIPTION

The function *cd_drec*( ) fills the *drec* structure with the contents of the Directory Record associated with a file or directory referred to by *path*. The argument *fsec* specifies the File Section of that file. The numbering starts with one. The number –1 denotes the last File Section of the named file, or the only File Section of the named directory. The argument *path* points to a file or directory within the CD-ROM file hierarchy. The declaration for **struct** *iso9660_drec* is contained in **<sys/cdrom.h>**.

The function *cd_cdrec*( ) copies the complete Directory Record as recorded on the CD-ROM to the address *drec*. The user must allocate {CD_MAXDRECL} bytes for the Directory Record. {CD_MAXDRECL} is contained in **<sys/cdrom.h>**.

RETURN VALUE

Upon successful completion, the functions return a value of zero. In case of an error, –1 is returned and *errno* is set to indicate the error.

ERRORS

The functions will fail if:

[EACCES] Search permission is denied for a component of the *path* prefix, or read permission is denied for the directory in which the file or directory pointed to by *path* is located.

[ENAMETOOLONG]
The length of the *path* string exceeds {PATH_MAX}, or a pathname component is longer than {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.

[ENOENT] A component of *path* does not exist or the *path* argument points to an empty string.

[ENOTDIR] A component of the *path* prefix is not a directory.

[EFAULT] The address of *drec* or *path* is invalid.

[EINVAL] The value of *fsec* is invalid.

The argument *path* points to a file/directory not within a CD-ROM file hierarchy.

[ENXIO] The CD-ROM is not in the drive or a read error occurred.

[EINTR] A signal was caught during the one of the functions.

[EMFILE] {OPEN_MAX} file descriptors are currently open in the calling process.

[ENFILE] The system file table is full.

SEE ALSO

<sys/cdrom.h>

NAME

cd_ptrec, cd_cptrec — read Path Table Record from CD-ROM Path Table

SYNOPSIS

```
#include <sys/cdrom.h>

int  cd_ptrec (path, ptrec)
char  *path;
struct iso9660_ptrec  *ptrec;

int  cd_cptrec (path, ptrec)
char  *path;
char  *ptrec;
```

DESCRIPTION

The function *cd_ptrec*( ) fills the *ptrec* structure with the contents of the Path Table Record associated with a directory which is referred to by the argument *path*. The argument *path* points to a directory within the CD-ROM file hierarchy. The declaration for **struct** *iso9660_ptrec* is contained in **<sys/cdrom.h>**.

The function *cd_cptrec*( ) copies the complete Path Table Record as recorded on the CD-ROM to the address *ptrec*. The user must allocate {CD_MAXPTRECL} bytes for the Path Table Record. {CD_MAXPTRECL} is contained in **<sys/cdrom.h>**.

RETURN VALUE

Upon successful completion, the functions return a value of zero.

In case of an error, –1 is returned and *errno* is set appropriately.

ERRORS

The functions will fail if:

[EACCES]	Search permission is denied for a component of the *path* prefix, or read permission is denied for the named directory.
[ENAMETOOLONG]	The length of the *path* string exceeds {PATH_MAX}, or a pathname component is longer than {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.
[ENOENT]	A component of *path* does not exist or the *path* argument points to an empty string.
[ENOTDIR]	A component of *path* is not a directory.
[ENXIO]	The CD-ROM is not in the drive, or a read error occurred.
[EFAULT]	The address of *ptrec* or *path* is invalid.
[EINVAL]	The argument *path* points to a directory not within a CD-ROM file hierarchy.
[EINTR]	A signal was caught during one of the functions.
[EMFILE]	{OPEN_MAX} file descriptors are currently open in the calling process.
[ENFILE]	The system file table is full.

SEE ALSO

<sys/cdrom.h>

NAME

cd_type — get identification of CD-ROM

SYNOPSIS

```
#include <sys/cdrom.h>

int  cd_type (path)
char  *path;
```

DESCRIPTION

This function determines the type of a CD-ROM. The argument *path* points to a pathname naming a file or directory within the CD-ROM file hierarchy, or to a pathname naming the block special file for the CD-ROM file system. The return value of *cd_type*( ) indicates the type of the CD-ROM. The definition for the return value is contained in **<sys/cdrom.h>**. This function is intended for future expansion, when XCDR may support more than one CD-ROM type, like CD-ROM XA.

RETURN VALUE

Upon successful completion, *cd_type*( ) returns the following value:

{CD_ISO9660} CD-ROM is recorded according to ISO 9660.

In case of other CD-ROM types, the return value is implementation-defined.

In case of an error, –1 is returned and *errno* is set to indicate the error.

ERRORS

The *cd_type*( ) function will fail if:

[EACCES]	Search permission is denied for a component of the *path* prefix, or read and execute permission are denied on the named file or directory, or read permission is denied on the block special file pointed to by *path*.
[ENAMETOOLONG]	The length of the *path* string exceeds {PATH_MAX}, or a pathname component is longer than {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.
[ENOENT]	A component of *path* does not exist or the *path* argument points to an empty string.
[ENOTDIR]	A component of the *path* prefix is not a directory.
[EFAULT]	The address of *path* is invalid.
[EINVAL]	The argument *path* points to a file/directory not within a CD-ROM file hierarchy.
[ENXIO]	The named file is a block special file and the device associated with the special file does not exist. The CD-ROM is not in the drive or a read error occurred.
[EINTR]	A signal was caught during the *cd_type*( ) function.
[EMFILE]	{OPEN_MAX} file descriptors are currently open in the calling process.
[ENFILE]	The system file table is full.

SEE ALSO

<sys/cdrom.h>

6.3 CD-ROM-specific Administrative Library Functions

This section presents the manual pages for the CD-ROM library functions for system administrators.

NAME

cd_defs — set and get default values for User / Group ID and file / directory permissions

SYNOPSIS

```
#include <sys/cdrom.h>

int cd_defs (path, cmd, defs)
char  *path;
int  cmd;
struct cd_defs *defs;
```

DESCRIPTION

This function sets or gets (based upon *cmd*) the defaults for User IDs, Group IDs, file / directory permissions and directory search permissions for the mounted CD-ROM. The argument *path* points to a mount-point of a CD-ROM file system. The argument *cmd* is either {CD_SETDEFS} or {CD_GETDEFS}. The declaration for **struct** *cd_defs* and the definitions for {CD_SETDEFS} and {CD_GETDEFS} are contained in **<sys/cdrom.h>**. See also Section 2.4.1 on page 12 and Section 2.4.2 on page 12 for further information.

RETURN VALUE

Upon successful completion, *cd_defs*( ) returns a value of zero.

In case of an error, –1 is returned and *errno* is set appropriately.

ERRORS

The *cd_defs*( ) function will fail if:

[EACCES]	Search permission is denied for a component of the *path* prefix, or read permission is denied on the mount-point.
[ENAMETOOLONG]	The length of the path string exceeds {PATH_MAX}, or a pathname component is longer than {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.
[ENOENT]	A component of *path* does not exist, or the *path* argument points to an empty string.
[ENOTDIR]	A component of the *path* is not a directory.
[EINVAL]	The value of *cmd* or values of members of the *cd_defs* structure are invalid.
	The argument *path* does not point to a mount-point of a CD-ROM file system.
[EFAULT]	The address for the structure *cd_defs* or *path* is invalid.
[EPERM]	User does not have appropriate privileges in case of setting values.
[EINTR]	A signal was caught during the *cd_defs*( ) function.
[EMFILE]	{OPEN_MAX} file descriptors are currently open in the calling process.
[ENFILE]	The system file table is full.

APPLICATION USAGE

The setting of default values is restricted to a user with appropriate privileges. In case of setting default values, this function is intended to be used only directly after the CD-ROM has been mounted, before any access to the CD-ROM is done. If the function is applied for setting default values when files or directories have already been opened, the effect of this function on these files and directories is undefined.

SEE ALSO

<sys/cdrom.h>

NAME

cd_idmap — set and get mappings of User / Group IDs

SYNOPSIS

```
#include <sys/cdrom.h>

int  cd_idmap (path, cmd, idmap, nmaps)
char  *path;
int  cmd;
struct cd_idmap  *idmap;
int  *nmaps;
```

DESCRIPTION

This function sets or gets (based upon *cmd*) the mapping of User or Group IDs for the mounted CD-ROM. The argument *path* points to a mount-point of a CD-ROM file system.

If *cmd* is {CD_SETUMAP} or {CD_SETGMAP}, this uses the *idmap* array of mappings to map User or Group IDs. The argument *nmaps* indicates the number of mappings in the array. Any mapping or value set with a previous invocation of *cd_idmap*( ) is overridden. When *nmaps* is zero, none of previously set mappings will stay in effect.

If *cmd* is {CD_GETUMAP} or {CD_GETGMAP}, this fills the array of mappings of User or Group IDs with the current mappings. On call, *nmaps* must contain the maximum number of mappings that may be returned. On return, *nmaps* will contain the number of mappings that are returned.

The declaration for **struct** *cd_idmap* and the definitions for {CD_SETUMAP} / {CD_SETGMAP} and {CD_GETUMAP} / {CD_GETGMAP} are contained in **<sys/cdrom.h>**. See Section 2.4.3 on page 12 for further information.

RETURN VALUE

Upon successful completion, *cd_idmap*( ) returns a value of zero.

In case of an error, –1 is returned and *errno* is set to indicate the error.

ERRORS

The *cd_idmap*( ) function will fail if:

[EACCES] Search permission is denied for a component of the *path* prefix, or read permission is denied on the mount-point.

[ENAMETOOLONG]
The length of the *path* string exceeds {PATH_MAX}, or a pathname component is longer than {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.

[ENOENT] A component of *path* does not exist, or the *path* argument points to an empty string.

[ENOTDIR] A component of the *path* is not a directory.

[EFAULT] The address for the structure *cd_idmap* or *path* is invalid.

[EINVAL] The value of *cmd* or *nmaps* (negative or larger than {CD_MAXUMAP} and {CD_MAXGMAP}, respectively) is invalid.

A member of the *cd_idmap* structure is invalid: *from_id* is larger than 65535 or a value in *to_uid* or *to_gid* is not supported by the system. Note that this error will not be given when *from_id* does not exist on the CD-ROM, or when *to_uid* is not defined in the User Database or *to_gid* is not defined in the Group Database.

The argument *path* does not point to a mount-point of a CD-ROM file system.

[EPERM]	User does not have appropriate privileges in case of setting values.
[EINTR]	A signal was caught during the *cd_idmap*( ) function.
[EMFILE]	{OPEN_MAX} file descriptors are currently open in the calling process.
[ENFILE]	The system file table is full.

APPLICATION USAGE

The setting of values is restricted to a user with appropriate privileges. Only files and directories with an unrestricted final XAR are subject to this mapping. An owner or group identification of zero is not permitted by ISO 9660 to appear in an unrestricted XAR, and thus the behaviour is undefined if a mapping is given for the value zero. Use *cd_defs*( ) to change the default values. The maximum number of mappings is defined in **<sys/cdrom.h>**. In case of set mappings this function is intended to be used only directly after the CD-ROM has been mounted, before any access to the CD-ROM is done. If the function is applied for setting mappings when files or directories have already been opened, the effect of the function on these files and directories is undefined.

SEE ALSO

<sys/cdrom.h>

NAME

cd_nmconv — set and get CD-ROM filename conversion

SYNOPSIS

```
#include <sys/cdrom.h>

int  cd_nmconv (path, cmd, flag)
char  *path;
int  cmd;
int  *flag;
```

DESCRIPTION

This function sets or gets (based upon *cmd*) the name conversion flag for filenames on the mounted CD-ROM. The argument *path* points to a mount-point of a CD-ROM file system. The argument *cmd* is either {CD_SETNMCONV} or {CD_GETNMCONV}. For further information see also Section 2.4.4 on page 13 and Section 3.1 on page 17. The parameter flag is one of the following:

{CD_NOCONV}:	No conversion (default after mounting of the CD-ROM).
{CD_LOWER}:	Characters in Identifiers on CD-ROM are converted to lower case when represented in the XSI file hierarchy. If a File Identifier contains no File Name Extension, the SEPARATOR 1 (.) is not represented.
{CD_NOVERSION}:	The Version Number and the SEPARATOR 2 (;) of a File Identifier is not represented in the XSI file hierarchy.

{CD_LOWER} and {CD_NOVERSION} may be bitwise-inclusive or-ed.

RETURN VALUE

Upon successful completion, *cd_nmconv*( ) returns a value of zero.

In case of an error, –1 is returned and *errno* is set to indicate the error.

ERRORS

The *cd_nmconv*( ) function will fail if:

[EACCES]	Search permission is denied for a component of the *path* prefix, or read permission is denied on the mount-point.
[ENAMETOOLONG]	The length of the *path* string exceeds {PATH_MAX}, or a pathname component is longer than {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.
[ENOENT]	A component of *path* does not exist or the *path* argument points to an empty string.
[ENOTDIR]	A component of the *path* is not a directory.
[EFAULT]	The address of *flag* or *path* is invalid.
[EINVAL]	The value of *cmd* or *flag* is invalid.
	The argument *path* does not point to a mount-point of a CD-ROM file system.
[EPERM]	User does not have appropriate privileges in case of setting values.
[EINTR]	A signal was caught during the *cd_nmconv*( ) function.
[EMFILE]	{OPEN_MAX} file descriptors are currently open in the calling process.
[ENFILE]	The system file table is full.

APPLICATION USAGE

The setting of values is restricted to a user with appropriate privilege. In case of setting filename conversion, this function is intended to be used only directly after the CD-ROM has been mounted, before any access to the CD-ROM is done. If the function is applied for setting filename conversion when files or directories have already been opened, the effect of this function on these files and directories is undefined.

SEE ALSO

<sys/cdrom.h>

Chapter 7

Header

This chapter describes the contents of the header used by XCDR.

Headers contain the definitions of the symbolic constants, common structures, preprocessor macros and defined types. Each function in he=1 .ds ;p Chapter 6 on page 41specifies in order to use that function. This header must be present on an application's development system. It does not have to be present on the target execution system.

NAME

cdrom.h — XCDR definitions and declarations

SYNOPSIS

```
#include  <sys/cdrom.h>
```

DESCRIPTION

The **<cdrom.h>** header contains the XCDR constants definitions and structure declarations for the XCDR library functions.

The structures *iso9660_pvd*, *iso9660_xar*, *iso9660_drec* and *iso9660_ptrec* contain fields in a form such that their contents are directly usable by an application. When the same value is recorded on the same CD-ROM multiple times (for example, in different byte orders), XCDR will convert one of those values to the proper byte ordering for the system where the application runs.

The contents of the time fields on CD-ROM are converted to the nearest value of time in seconds since Epoch (hundreds of seconds are ignored). Times which cannot be represented in **time_t** will be represented by 0 (for times before 1 January 1970 UTC) and by the highest possible value of **time_t** (for times too far in future to be represented).

The structure *iso9660_pvd* used by *cd_pvd*( ) contains at least the following members in any order:

unsigned char	voldestype	Volume Descriptor Type
unsigned char	std_id[5]	Standard Identifier
unsigned char	voldesvers	Volume Descriptor Version
unsigned char	sys_id[32]	System Identifier
unsigned char	vol_id[32]	Volume Identifier
unsigned long	volspcsize	Volume Space Size (in logical blocks)
unsigned short	volsetsize	Volume Set Size
unsigned short	volseqno	Volume Sequence Number
unsigned short	lblksize	Logical Block Size (in bytes)
unsigned long	ptsize	Path Table Size (in bytes)
unsigned long	locpt_l	Location of Occurrence of Type L Path Table
unsigned long	locptopt_l	Location of Optional Occurrence of Type L Path Table
unsigned long	locpt_m	Location of Occurrence of Type M Path Table
unsigned long	locptopt_m	Location of Optional Occurrence of Type M Path Table
unsigned char	rootdir[34]	Directory Record for Root Directory
unsigned char	volset_id[128]	Volume Set Identifier
unsigned char	pub_id[128]	Publisher Identifier
unsigned char	dtpre_id[128]	Data Preparer Identifier
unsigned char	app_id[128]	Application Identifier
unsigned char	cpfile_id[37]	Copyright File Identifier
unsigned char	abfile_id[37]	Abstract File Identifier
unsigned char	bgfile_id[37]	Bibliographic File Identifier
time_t	cre_time	Volume Creation Date and Time
time_t	mod_time	Volume Modification Date and Time
time_t	exp_time	Volume Expiration Date and Time
time_t	eff_time	Volume Effective Date and Time
unsigned char	filestrver	File Structure Version
unsigned char	res1	(Reserved for future standardisation) byte position 883 in Primary Volume Descriptor on CD-ROM
unsigned char	appuse[512]	Application Use
unsigned char	res2[653]	(Reserved for future standardisation) byte position 1396 up to 2048 in Primary Volume Descriptor on CD-ROM

The following symbolic name defines the length of the Primary Volume Descriptor. It is intended to be used by the *cd_cpvd*( ) library function.

{CD_PVDLEN}	Length of Primary Volume Descriptor on CD-ROM in bytes (2048).

The structure *iso9660_xar* is used by *cd_xar*( ). It does not contain the complete Application Use and the Escape Sequences fields because these fields are of variable length (without reasonable limit). If an application wants to read the XAR with these fields, it must allocate enough memory beyond this structure. The structure *iso9660_xar* contains at least the following members in any order:

unsigned short	own_id	Owner Identification
unsigned short	grp_id	Group Identification
unsigned short	permissions	Permissions (see below)
time_t	cre_time	File Creation Date and Time
time_t	mod_time	File Modification Date and Time
time_t	exp_time	File Expiration Date and Time
time_t	eff_time	File Effective Date and Time
unsigned char	rec_form	Record Format
unsigned char	rec_attr	Record Attributes
unsigned short	rec_len	Record Length
unsigned char	sys_id[32]	System Identifier
unsigned char	sys_use[64]	System Use
unsigned char	xar_vers	XAR Version
unsigned char	esc_len	Length of Escape Sequences
unsigned char	resv[64]	(Reserved for future standardisation)
unsigned short	appuse_len	Length of Application Use
unsigned char	*app_use	Pointer to Application Use field data
unsigned char	*esc_seq	Pointer to Escape Sequences field data

The following symbolic names for the value *permissions* are also defined:

{CD_RSYS}	read permission, system class
{CD_XSYS}	execute permission, system class
{CD_RUSR}	read permission, owner
{CD_XUSR}	execute permission, owner
{CD_RGRP}	read permission, group
{CD_XGRP}	execute permission, group
{CD_ROTH}	read permission, others
{CD_XOTH}	execute permission, others

The following symbolic name defines the length of the fixed part of the XAR. This does not include the Application Use field and the Escape Sequences field. It is intended to be used by the *cd_cxar*( ) library function.

{CD_XARFIXL}	Length of fixed part of XAR in bytes (250).

The structure *iso9660_drec* is used by *cd_drec*( ) and contains at least the following members in any order:

unsigned char	drec_len	Length of Directory Record (in bytes as taken from CD-ROM)
unsigned char	xar_len	XAR Length (in logical blocks)
unsigned long	locext	Location of Extent
unsigned long	data_len	Data Length
time_t	rec_time	Recording Date and Time
unsigned char	file_flags	File Flags
unsigned char	file_usize	File Unit Size
unsigned char	ileav_gsize	Interleave Gap Size
unsigned short	volseqno	Volume Sequence Number
unsigned char	fileid_len	Length of File Identifier
unsigned char	file_id[37]	File Identifier [1.]
unsigned char	sysuse_len	Length of System Use [2.]
unsigned char	sys_use[218]	System Use

1. Note that the File Identifier, as recorded on the CD-ROM, can exceed 37 bytes due to leading zeros being present. Leading zeros will be stripped if the File Identifier is larger than 37 bytes and may be stripped otherwise.
2. Note that the *sysuse_len* field is calculated by *drec_len* minus *fileid_len* minus 34 (fixed part) minus 1 (only if *fileid_len* is even).

The following symbolic names for the value of *file_flags* are also defined:

{CD_EXIST}	Existence bit.
{CD_DIR}	Directory bit.
{CD_ASSOFILE}	Associated File bit.
{CD_RECORD}	Record bit.
{CD_PROTEC}	Protection bit.
{CD_MULTIEXT}	Multi-Extent bit.

The following symbolic name defines the maximal length of a Directory Record. It is intended to be used by the *cd_cdrec*( ) library function.

{CD_MAXDRECL}	Maximum Length of Directory Record in bytes (255).

The structure *iso9660_ptrec* is used by the function *cd_ptrec*( ) and contains the following members in any order:

unsigned char	dirid_len	Length of Directory Identifier (in bytes)
unsigned char	xar_len	XAR Length (in logical blocks)
unsigned long	loc_ext	Location of Extent
unsigned short	pdirno	Parent Directory Number
unsigned char	dir_id[31]	Directory Identifier

The following symbolic name defines the maximal length of a Path Table Record. It is intended to be used by the *cd_cptrec*( ) library function.

{CD_MAXPTRECL}	Maximum Length of Path Table Record in bytes (40).

The function *cd_type*( ) uses the following symbolic name:

{CD_ISO9660}	CD-ROM is recorded according to ISO 9660.

The function *cd_defs*( ) uses the following symbolic names for the argument *cmd*:

{CD_SETDEFS}	Set default values.
{CD_GETDEFS}	Get default values.

The structure *cd_defs* contains the following members:

uid_t	def_uid	Default User ID
gid_t	def_gid	Default Group ID
mode_t	def_fperm	Default File Permissions
mode_t	def_dperm	Default Directory Permissions
int	dirsperm	Directory search permission

The following symbolic names are possible for the variable *dirsperm*:

{CD_DIRXAR}	The execute permission bits for directories within the CD-ROM file hierarchy are set as provided in the Permissions field in the XAR of that directory.
{CD_DIRRX}	The execute permission bits for directories within the CD-ROM file hierarchy are set if the read or execute bits are set in the XAR of that directory on CD-ROM.

The function *cd_idmap*( ) uses the following values for the argument *cmd*:

{CD_SETUMAP}	Set User ID mapping.
{CD_SETGMAP}	Set Group ID mapping.
{CD_GETUMAP}	Get User ID mapping.
{CD_GETGMAP}	Get Group ID mapping.

The structure *cd_idmap* used by the function *cd_idmap*( ) contains the following members:

unsigned short	from_id	Owner Identification resectively Group Identification on CD-ROM
uid_t	to_uid	Owner ID in XSI file hierarchy
gid_t	to_gid	Group ID in XSI file hierarchy

The field *to_uid* is only used with {CD_SETUMAP} and {CD_GETUMAP}. The field *to_gid* is only used with {CD_SETGMAP} and {CD_GETGMAP}.

If an implementation imposes a limit on the number of User/Group ID mappings, they will be defined by the following symbolic names. The definition of these symbolic names may be omitted from **<sys/cdrom.h>** if the actual value of the limit is indeterminate but greater than the stated minimum. Applications should therefore only use these symbols in code conditionally compiled on the existence of the symbol.

Name	Minimum Acceptable Value
{CD_MAXUMAP}	50
{CD_MAXGMAP}	50

The function *cd_nmconv*( ) uses the following symbolic names for the argument *cmd*:

{CD_SETNMCONV}	Set filename conversion.
{CD_GETNMCONV}	Get filename conversion.

The following symbolic names are possible for the argument flag:

{CD_NOCONV}	No conversion.
{CD_LOWER}	Conversion to lower case; no "." if File Identifier contains no File Name Extension.
{CD_NOVERSION}	No Version Number and no ";".

Glossary

API
Application Programming Interface.

XAR
Extended Attribute Record.

Final XAR
The XAR associated with the last or only File Section of a file, or with the only File Section of a directory. This is the XAR that determines, among others, the access permissions of the whole file or directory, see Section 6.4.6 of ISO 9660. Note that if a file has no final XAR, the access permissions will be the default permissions, regardless of what was specified in any non-final XARs.

Restricted Final XAR
A final XAR, in combination with the Protection Bit set to ZERO; that is, resulting in the consequences described in Table 10 of ISO 9660,

Final Directory Record
The Directory Record identifying the last or only File Section of a file, or the only File Section of a directory.

XCDR
X/Open CD-ROM Support Component (as specified by this document).

Mount-point
The directory in the file hierarchy which is to become the root of the removable file system when it is mounted.

Multi-volume set
A Volume Set (as defined in ISO 9660) which consists of more than one Volume.

In addition, all the definitions in ISO 9660 apply to this document.

Appendix A

Recommendations for CD-ROM Publishers

The recommendations below are intended to provide a summary of the information required by publishers of CD-ROMs who want to make their CD-ROMs accessible to compliant XCDR implementations. This appendix is not a formal part of the XCDR specification.

A.1 Inclusion of an XAR

If files on the CD-ROM must have different access permissions, for example, read/execute permission and ownership of files, an Extended Attribute Record (XAR) must be included with the last File Section of each file or the (only) File Section of a directory. Note that, dependent on the implementation of XCDR and the way that file attributes are accessed, inclusion of an XAR may decrease performance. Because the XAR is located on the CD-ROM consecutive to the file contents rather than in the directory, accessing the XARs of all files in a directory (for example, by the command *ls* **–l**) will imply a seek operation of the CD-ROM drive for each XAR. CD-ROM seek operations are basically slow because, for example, the rotation speed of the platter must be adjusted when seeking. Note that accessing the XAR of a directory (with similar performance implications) will take place when the directory is part of the path of a file that is accessed by *open*( ) or *stat*( ) system interfaces.

An XCDR implementation can, and typically will, improve the performance behaviour by caching the XAR information.

A.2 Execute Permission of Directories

By default, XCDR interprets execute permission of a directory to mean search permission. In order to *open*( ) or *stat*( ) a file, all components of the *path* prefix must have search permission. When a directory has a non-restricted XAR (see the definition in Chapter 7 on page 67), consideration must be given whether to set the execute permissions of that directory.

A.3 Values for User ID and Group IDs

Unless the CD-ROM is targetted at a known collection of systems, the values of User ID and Group ID recorded on a CD-ROM will not identify the desired target user(s) at the system where the CD-ROM is mounted. Also, the range of values for User ID and Group ID that a system can handle might be smaller than what can be recorded on the CD-ROM. When producing a CD-ROM to be used at various systems, it is recommended to number the User IDs and Group IDs consecutively from 1, and to provide the system administrator with sufficient information (for example, by the installation mechanism described in Appendix B on page 71) to let the system administrator map each recorded User ID and Group ID to the best values for the target system.

A.4 Multi-volume Sets and Multiple File Sections

An XCDR implementation is not required to support files consisting of multiple File Sections on different volumes. An XCDR implementation is not required to support transparent access to files located on one CD-ROM referenced from a directory on another CD-ROM.

A.5 Install Mechanisms

It is recommended to include a shell script and readable information, which can be used by system administrators to map names on the CD-ROM, adjust access rights and map User IDs and Group IDs to the values most suitable in the receiving system. See Appendix B on page 71.

A.6 Naming of Files and Directories

An XCDR implementation can impose a limit on the length of the representation of the File Identifier (after conversion) in the XSI file hierarchy. See Section 3.1 on page 17. For full portability of CD-ROM discs between X/Open-compliant systems with XCDR, producers should ensure that the represented File Identifiers and Directory Identifiers do not exceed this limit.

File Identifiers and Directory Identifiers can be converted, see Section 2.4.4 on page 13 and Section 3.1 on page 17.

- The truncation of version number gives additional bytes that can be used for other parts of the name and can effectively be used in those cases where no multiple versions of files will exist, or where only the file with the highest version number needs to be accessed. It also is closer to common practice in X/Open-compliant systems.
- The conversion of names to lower case and stripping of a trailing dot separator can effectively be used when the CD-ROM is intended to be visible to users as they would see other (read-only) file systems. This conversion makes the CD-ROM filenames closer to common practice in X/Open-compliant systems.

Installation of Application Software and Usage for Software Distribution

This appendix is not a formal part of the XCDR specification.

It is recommended to parties producing information on the CD-ROM, to include installation files that hold shell scripts, interpretable by the *sh* command (defined in the XCU, Issue 3, Volume 1) as well as descriptive texts with instructions. These files can be used to install the software of the package(s) on the disc, and/or to advise the system administrator on the commands to execute after mounting.

The recommended way of doing this is to include an Abstract File, the name of which is contained in the Abstract File Identifier field of the Primary Volume Descriptor on the CD-ROM. The Abstract File contains the pathname of the Installation Directory on the CD-ROM relative to the root of the CD-ROM. This pathname starts with a forward slash (/). If the Abstract File contains other information apart from the pathname, the pathname must be followed by a blank. Any contents of the Abstract File following a blank are unspecified.

The Installation Directory will hold a subdirectory for each package on the CD-ROM. This subdirectory will hold files with descriptive texts and shell scripts.

Three examples of the practical use of this mechanism are given below.

- CD-ROMs can be used for software distribution. Though applications can be executed from CD-ROM, this is sometimes not desirable for reasons of performance. Also the restrictions to filenames and the read-only property of the CD-ROM might sometimes make copying (part of) the contents to magnetic disk desirable. The Installation Directory will, in this case, contain a subdirectory for each package on the CD-ROM, and each subdirectory will contain a shell script which can copy the appropriate files to the correct places on magnetic disk under the desired names.

- When a CD-ROM is used for information retrieval, and the application which performs the information retrieval is distributed on the CD-ROM itself, it can also be desirable to copy it to magnetic disk. This can be for performance reasons, but also where the CD-ROM holds multiple versions of the application in binary form for different hardware architectures. By copying the applicable version to magnetic disk under a fixed name, end-user documentation can be the same for all hardware architectures. The Installation Directory will in this case hold a subdirectory with a shell script that detects or asks for the hardware architecture of the machine on which the CD-ROM is mounted and will copy the appropriate instance of the application to magnetic disk.

- The installation mechanism can effectively be used to present the system administrator with a list of the User IDs and Group IDs recorded on the CD-ROM, with a description of the types of information on the CD-ROM and the intended access privileges. The system administrator can then perform the best mapping for the actual system.

Index