LIBCDVD 1.13
Reference Manual

Filing System Research & Code
by A. Lee (aka Hiryu)

RPC Interface Code
by A. Lee (aka Hiryu)
and Nicholas Van-Veen (aka Sjeep)

Reference Manual
by A. Lee (aka Hiryu)

Table of Contents

Revision History

Version	Comment
1.00	First public release
1.10	Major reading speedups, and directory caching, and more
1.11	Fixed bug incorrectly handling the first entry of the root directory as a parent dir
1.12	Fixed potential cache problem (thanks Nick)
1.13	Added FlushCache function to allow reading from multiple CDs

Introduction

The disc filing system provided by the PS2 kernel only provides strict ISO Level 1 filing system support. This imposes a number of heavy restrictions: -

- 8.3 character filenames (no long filename support)
- Upper case filenames only (no mixed case filenames)
- Maximum of 31 files/sub-directories in each directory
- 255 char max path length
- Maximum of path depth of 8 directories

Libcdvd lifts these restrictions by providing full support for the following filing systems: -

- ISO Level 1 (relaxed)
  8.3 character filenames (upper-case only)
  1024 char max path length
  Max path depth restriction lifted

- ISO Level 2 (relaxed)
  Up to 31 character filenames (upper-case only)
  1024 character max path length
  Max path depth restriction lifted

- Joliet
  Up to 63 character filenames
  Mixed case filenames
  1024 character max path length
  No path depth restriction (except by path length)

- UDF
  Up to 128 character filenames (upper-case only)
  1024 character max path length
  No path depth restriction (except by path length)

This filing system can be used as a 100% replacement for the existing kernel filing system, and provided additional functionality via an RPC interface.

Since Joliet and UDF discs also contain ISO Level 1 volume descriptors and directory tables it is possible to create a Joliet or UDF disc which is bootable, as long as the SYSTEM.CNF file, the main executable code, and this IRX all have valid 8.3 filenames. (The SYSTEM.CNF specifies the ELF which is loaded by the kernel, and the ELF then loads the libcdvd IRX before loading any files using long filenames)

All information used in the development of this filing system module was gathered without documentation of any kind, and was cleanly 'reverse engineered' using only CD/DVD images as examples.

Filing System Interface

This filing system module provides a standard PS2 filing system interface which can be used with any existing file I/O routines (that work with the kernels CD/DVD filing system).

Rather than replacing the existing **_cdrom:_** device provided by the kernel filing system, this module provides a **_cdfs:_** device so that, if required, both filing systems can be used in parallel (although this module provides all the functionality provided by the kernel filing system, and more).

This means that to load a open a file from a CD using this filing system you would do: -
open("cdfs:/My Directory/My Filename.ext", O_RDONLY);
or the equivalent using your file I/O functions.

(note that before using the cdfs: filing system it is necessary to load the IRX, and call the CDVD_Init() function provided)

RPC Function Reference

The following section details the functions provided by Libcdvd that provide extra functionality beyond the basic operations of the standard PS2 file I/O interface.

These functions access the module via the RPC interface but full source and header files are included so that these can be easily used within your code.

Structures

Only one public structure is provided by Libcdvd.
This structure is used by the CDVD_FindFile and CDVD_GetDir functions, to return information describing a directory entry (or a number of directory entries)

```
struct TocEntry
{
        u32     fileLBA;
        u32     fileSize;
        u8      fileProperties;
        u8      padding1[3];
        u8      filename[128+1];
        u8      padding2[3];
} __attribute__((packed));
```

'**fileLBA**' is the Logical Block Address (or sector) location of the file on disk

'**fileSize**' is the size of the file in bytes

'**fileProperties**' is the ISO file properties (described below)

'**filename**' is the name of the file (Unicode filenames are pre-converted to ascii)

The '**fileProperties**' field may contain the following bit definitions:

Bit	Description
0	Hidden File – If set then the directory element is hidden
1	Directory – If set then the directory element is a sub-directory (rather than a file)
2	Not Used
3	Not Used
4	Not Used
5	Not Used
6	Not Used
7	Not Used

CDVD_Init

int CDVD_Init()

Description:

This function initialises the filing system module ready for use. This must be called before
using either the cdfs: filing system device, or using the extended RPC functionality.

CDVD_DiskReady

```
int CDVD_DiskReady(
            int mode            // Define whether to return current state, or to wait for disk
      ready
      )
```

Description:

This function returns the readiness state of the drive and filing system.

'mode' can be set to one of the following values:

> **CdBlock**
> The function will block until the drive is ready.
>
> **CdNonBlock**
> The function will return immediately with the current state.

This function can return the following values:

> **CdComplete**
> The drive is ready to be used for file I/O.
>
> **CdNotReady**
> The drive is not ready to be used for file I/O.

CDVD_FindFile

```
int CDVD_FindFile(
            char* fname,                // Case insensitive filename to find
            struct TocEntry* tocEntry   // Pointer to a TocEntry structure to fill
      )
```

Description:

This function is used to find a file of known filename, but unknown filename case.

i.e.: you know what the file (or directory) name should be, but don't know whether the filename is upper-case or lower-case or a mixture of both.

It can also be used to return the additional information provided by the TocEntry structure (i.e.: file size, and file properties etc).

'fname' should contain the (case-insensitive) filename of the file to be found (including full path)

'tocEntry' should contain a pointer to a TocEntry structure to be filled.

This function will return TRUE if a matching file is found, or FALSE if no matching file is found.

CDVD_Stop

void CDVD_Stop()

Description:

This function will stop the motor of the disc drive. (Mostly to make the damn thing quieter).

This will interrupt any current background file I/O operations.

It will also mean that any subsequent file I/O operations will have to wait for the drive to spin up to speed again before they can proceed.

CDVD_TrayReq

```
int CDVD_TrayReq(
            int mode
      )
```

Description:

This function is used to open or close the drive tray, or to request the current state of the tray.

'**mode**' can be set to one of the following values:

CdTrayOpen
Open the drive tray (and return the state after the operation)

CdTrayClose
Close the drive tray (and return the state after the operation)

CdTrayCheck
Request the current state of the drive tray (without opening or closing it)

This function can return the following values:

CdTrayOpen
The drive tray is currently open

CdTrayClose
The drive tray is currently closed

CDVD_GetDir

```
int CDVD_GetDir(
              const char* pathname,        // Pathname to retrieve file entries from
              const char* extensions,      // List of file extensions to retrieve file entries of
              struct TocEntry tocEntry[],  // Array of TocEntry structures to fill
              unsigned int req_entries     // Max number of entries to return
              char* new_pathname           // updated pathname
       )
```

Description:

This function returns the directory entries (files and sub-directories) within the specified path. An optional (white-space separated) list of file-extensions can be specified. If a list is provided then only files with matching extensions will be returned.

'**pathname**' should contain the pathname of the directory to be retrieved. This may be a const string, but if it is a changeable string (as preferred) then the same string should also be specified as the 'new_pathname' parameter so that it may be updated to the absolute path when navigating through directories using ".." parent pointers

'**extensions**' may contain a white-space separate list of file extensions (including the dot seperator) of files to include in the returned list. If no list is provided (the parameter is NULL) then all directory entries (up to 'req_entries') will be returned.

'**tocEntry**' should contain a pointer to an array of TocEntry structures to be filled (this array must contain at least the number of elements specified by 'req_entries' for safe operation).

'**req_entries**' contains the maximum number of directory entries to return in the 'tocEntry' array (typically this number should be the same as the number of elements in the 'tocEntry' array).

'**new_pathname**' should point to a string to be updated with the absolute path which has been retrieved. For example, when this function is used for navigating through directories and the user selects the ".." entry to go up a level, then this string is updated with the new absolute path. This pointer may be NULL if the specified pathname is const, and you are only trying to retrieve a fixed directory path.

The function will return the number of directory entries retrieved and stored in the 'tocEntry' array.

CDVD_FlushCache

void CDVD_FlushCache()

Description:

This function flushes the cached CD contents. This is needed if you wish to allow reading from multiple CDs or DVDs.

The function MUST be called after changing the CD or DVD and before calling any other functions.

Remember that this will still only allow you to read from a CD that is 'authorised'. It will not allow the reading of CDRs or DVDRs on an unmodified PS2 console.