39.3. cd — CD-ROM access on SGI systems¶

Platforms: IRIX

Deprecated since version 2.6: The `cd` module has been deprecated for removal in Python 3.0.

This module provides an interface to the Silicon Graphics CD library. It is available only on Silicon Graphics systems.

The way the library works is as follows. A program opens the CD-ROM device with `open()` and creates a parser to parse the data from the CD with `createparser()`. The object returned by `open()` can be used to read data from the CD, but also to get status information for the CD-ROM device, and to get information about the CD, such as the table of contents. Data from the CD is passed to the parser, which parses the frames, and calls any callback functions that have previously been added.

An audio CD is divided into *tracks* or *programs* (the terms are used interchangeably). Tracks can be subdivided into *indices*. An audio CD contains a *table of contents* which gives the starts of the tracks on the CD. Index 0 is usually the pause before the start of a track. The start of the track as given by the table of contents is normally the start of index 1.

Positions on a CD can be represented in two ways. Either a frame number or a tuple of three values, minutes, seconds and frames. Most functions use the latter representation. Positions can be both relative to the beginning of the CD, and to the beginning of the track.

Module `cd` defines the following functions and constants:

`cd.createparser`()¶
Create and return an opaque parser object. The methods of the parser object are described below.

`cd.msftoframe`(*minutes*, *seconds*, *frames*)¶
Converts a `(minutes, seconds, frames)` triple representing time in absolute time code into the corresponding CD frame number.

`cd.open`([*device*[, *mode*]])¶
Open the CD-ROM device. The return value is an opaque player object; methods of the player object are described below. The device is the name of the SCSI device file, e.g. `'/dev/scsi/sc0d4l0'`, or `None`. If omitted or `None`, the hardware inventory is consulted to locate a CD-ROM drive. The *mode*, if not omitted, should be the string `'r'`.

The module defines the following variables:

exception `cd.error`¶
Exception raised on various errors.

`cd.DATASIZE`¶
The size of one frame's worth of audio data. This is the size of the audio data as passed to the callback of type `audio`.

`cd.BLOCKSIZE`¶
The size of one uninterpreted frame of audio data.

The following variables are states as returned by `getstatus()`:

`cd.READY`¶
The drive is ready for operation loaded with an audio CD.

`cd.NODISC`¶
The drive does not have a CD loaded.

`cd.CDROM`¶
The drive is loaded with a CD-ROM. Subsequent play or read operations will return I/O errors.

`cd.ERROR`¶
An error occurred while trying to read the disc or its table of contents.

`cd.PLAYING`¶
The drive is in CD player mode playing an audio CD through its audio jacks.

`cd.PAUSED`¶
The drive is in CD layer mode with play paused.

`cd.STILL`¶

The equivalent of PAUSED on older (non 3301) model Toshiba CD-ROM drives. Such drives have never been shipped by SGI.

`cd.audio`¶

`cd.pnum`¶

`cd.index`¶

`cd.ptime`¶

`cd.atime`¶

`cd.catalog`¶

`cd.ident`¶

`cd.control`¶

Integer constants describing the various types of parser callbacks that can be set by the `addcallback()` method of CD parser objects (see below).

39.3.1. Player Objects¶

Player objects (returned by `open()`) have the following methods:

`CD player.allowremoval()`

Unlocks the eject button on the CD-ROM drive permitting the user to eject the caddy if desired.

`CD player.bestreadsize()`

Returns the best value to use for the *num_frames* parameter of the `readda()` method. Best is defined as the value that permits a continuous flow of data from the CD-ROM drive.

`CD player.close()`

Frees the resources associated with the player object. After calling `close()`, the methods of the object should no longer be used.

`CD player.eject()`

Ejects the caddy from the CD-ROM drive.

`CD player.getstatus()`

Returns information pertaining to the current state of the CD-ROM drive. The returned information is a tuple with the following values: *state*, *track*, *rtime*, *atime*, *ttime*, *first*, *last*, *scsi_audio*, *cur_block*. *rtime* is the time relative to the start of the current track; *atime* is the time relative to the beginning of the disc; *ttime* is the total time on the disc. For more information on the meaning of the values, see the man page *CDgetstatus(3dm)*. The value of *state* is one of the following: ERROR, NODISC, READY, PLAYING, PAUSED, STILL, or CDROM.

`CD player.gettrackinfo(track)`

Returns information about the specified track. The returned information is a tuple consisting of two elements, the start time of the track and the duration of the track.

`CD player.msftoblock(min, sec, frame)`

Converts a minutes, seconds, frames triple representing a time in absolute time code into the corresponding logical block number for the given CD-ROM drive. You should use `msftoframe()` rather than `msftoblock()` for comparing times. The logical block number differs from the frame number by an offset required by certain CD-ROM drives.

`CD player.play(start, play)`

Starts playback of an audio CD in the CD-ROM drive at the specified track. The audio output appears on the CD-ROM drive's headphone and audio jacks (if fitted). Play stops at the end of the disc. *start* is the number of the track at which to start playing the CD; if *play* is 0, the CD will be set to an initial paused state. The method `togglepause()` can then be used to commence play.

`CD player.playabs(minutes, seconds, frames, play)`

Like `play()`, except that the start is given in minutes, seconds, and frames instead of a track number.

`CD player.playtrack(start, play)`

Like `play()`, except that playing stops at the end of the track.

`CD player.playtrackabs(track, minutes, seconds, frames, play)`

Like `play()`, except that playing begins at the specified absolute time and ends at the end of the specified track.

`CD player.preventremoval()`

Locks the eject button on the CD-ROM drive thus preventing the user from arbitrarily ejecting the caddy.

`CD player.readda(num_frames)`

Reads the specified number of frames from an audio CD mounted in the CD-ROM drive. The return value is a string representing the audio frames. This string can be passed unaltered to the `parseframe()` method of the parser object.

`CD player.seek(minutes, seconds, frames)`

Sets the pointer that indicates the starting point of the next read of digital audio data from a CD-ROM. The pointer is set to an absolute time code location specified in *minutes*, *seconds*, and *frames*. The return value is the logical block number to which the pointer has been set.

`CD player.seekblock(block)`

Sets the pointer that indicates the starting point of the next read of digital audio data from a CD-ROM. The pointer is set to the specified logical block number. The return value is the logical block number to which the pointer has been set.

`CD player.seektrack(track)`

Sets the pointer that indicates the starting point of the next read of digital audio data from a CD-ROM. The pointer is set to the specified track. The return value is the logical block number to which the pointer has been set.

`CD player.stop()`

Stops the current playing operation.

`CD player.togglepause()`

Pauses the CD if it is playing, and makes it play if it is paused.

39.3.2. Parser Objects¶

Parser objects (returned by `createparser()`) have the following methods:

`CD parser.addcallback(type, func, arg)`

Adds a callback for the parser. The parser has callbacks for eight different types of data in the digital audio data stream. Constants for these types are defined at the `cd` module level (see above). The callback is called as follows: `func(arg, type, data)`, where *arg* is the user supplied argument, *type* is the particular type of callback, and *data* is the data returned for this *type* of callback. The type of the data depends on the *type* of callback as follows:

Type	Value
`audio`	String which can be passed unmodified to `al.writesamps()`.
`pnum`	Integer giving the program (track) number.
`index`	Integer giving the index number.
`ptime`	Tuple consisting of the program time in minutes, seconds, and frames.
`atime`	Tuple consisting of the absolute time in minutes, seconds, and frames.
`catalog`	String of 13 characters, giving the catalog number of the CD.
`ident`	String of 12 characters, giving the ISRC identification number of the recording. The string consists of two characters country code, three characters owner code, two characters giving the year, and five characters giving a serial number.
`control`	Integer giving the control bits from the CD subcode data

`CD parser.deleteparser()`

Deletes the parser and frees the memory it was using. The object should not be used after this call. This call is done automatically when the last reference to the object is removed.

`CD parser.parseframe(frame)`

Parses one or more frames of digital audio data from a CD such as returned by `readda()`. It determines which subcodes are present in the data. If these subcodes have changed since the last frame, then `parseframe()` executes a callback of the appropriate type passing to it the subcode data found in the frame. Unlike the C function, more than one frame of digital audio data can be passed to this method.

`CD parser.removecallback(type)`

Removes the callback for the given *type*.

`CD parser.resetparser()`

Resets the fields of the parser used for tracking subcodes to an initial state. `resetparser()` should be called after the disc has been changed.