# CD-ROM

# *CD-ROM Access, Data Reading, Streaming, Error Handling, & Optimization*

- What is the CD Good At?
- What is the CD Bad At?
- Types of CD-ROM Access
- Streaming -VS- Normal Data Reading
- Error Handling
- Optimizing CD Throughput

# *What is the CD Good At?*

- Reading a continuous data stream at a constant data rate
  - Single-speed = 75 sectors per second
  - Double-speed = 150 sectors per second

# *What is the CD Good At? • Reading a continuous data stream*

- Single-Speed = 75 sectors per second
  - 153600 bytes for data (2048-byte) sectors
    - ISO 9660 format data sectors
    - Streaming Video Data
  - 175200 bytes for form 2 (2336-byte) sectors
    - XA-ADPCM Audio Data

# *What is the CD Good At? • Reading a continuous data stream*

- Double-speed = 150 sectors per second
  - 307200 bytes for data (2048-byte) sectors
    - ISO 9660 format data sectors
    - Streaming Video Data
  - 350400 bytes for form 2 (2336-byte) sectors
    - XA-ADPCM Audio Data

# *What is the CD Bad At? (Everything else…)*

- Seeking
- Starting & Stopping

# *What is the CD Bad At? Seeking*

- PlayStation CDROM Seek Time
  - Average seek time = 250ms (¼ second)
  - Some seeks may take more time
  - Seek errors can take seconds for recovery

# *What is the CD Bad At? Starting & Stopping*

- Not Optimized For Stopping & Starting
  - CD drives are designed to read a continuous data stream

# *Types of CDROM Access*

- ISO 9660
- High-level functions
- Low-level functions

# *Types of CDROM Access • ISO 9660*

- The PlayStation disc is an extended version of the CD-ROM XA format.
  - Combines digital audio & data tracks on same disc
  - Uses ISO 9660 file system

# *Types of CDROM Access • ISO 9660*

- ***_96_init***
  - What It Does
    - Initializes ISO 9660 file system.
  - When To Use It
    - When need to use functions that require ISO 9660 file system
      - ***Load, Exec,*** & ***LoadExec***
      - UNIX-style raw i/o functions ***open, read, close***
    - When background CD functions not required
      - CdRead, CdSearchFile, CdReadFile, CdControl, etc.

# *Types of CDROM Access • ISO 9660*

- ***_96_remove***
  - What It Does
    - Removes ISO 9660 file system.
  - When To Use It
    - When ISO 9660 file system no longer needed
    - Before calling ***_96_init***
    - Called by ***ResetCallback***

# *CDROM Initialization*

- CdInit
  - What It Does
    - Initializes CDROM system
  - When To Use It
    - At program initialization

# *Types of CDROM Access • High-level Functions*

- ***CdSearchFile***
- ***CdRead***
- ***CdReadFile***
- ***CdReadSync*** & ***CdReadCallback***

# *Types of CDROM Access • High-level*

- ***CdSearchFile***
  - What It Does
    - Searches the ISO9660 directory to locate the specified file and return the time code position of the beginning of the file, as well as the length of the file.

# *Types of CDROM Access • High-level*

- ***CdSearchFile*** Performance Considerations
  - Blocks execution of other code until finished.
  - Uses ***CdRead*** function internally
  - Only caches 1 directory at a time
    - Each time directory changes from the previous***CdSearchFile*** call, it must seek to root directory, then to each subdirectory.
      - Pauses the disc after reading each directory
      - Must do a seek for each directory

# *Types of CDROM Access • High Level*

- ***CdSearchFile*** execution chain of events

| CdSearchFile( "\DATA\LEVEL3\SHIPS\BATTLESH.DAT", &filedata ); | |
|---|---|
| **Searching for…** | **Actions Required** |
| DATA | Seek to DATA directory<br>Read Data, Pause Disc |
| LEVEL3 | Seek to LEVEL3 directory<br>Read Data, Pause Disc |
| SHIPS | Seek to SHIPS directory<br>Read Data, Pause Disc |
| BATTLESH.DAT | Find BATTLESH.DAT Entry In Directory & Return Data |
| TOTAL: | 3 seeks, 3 reads, 3 pauses |

# *Types of CDROM Access • High-level*

- ***CdSearchFile*** Alternatives
  - Include the timecode position of your files directly into your program at compile time
    - Obtain timecode information from CDGEN.
  - Use a lookup file with the name, timecode position, and length of every file, then read this at runtime.
    - Data file can be generated right before creating disc.
    - Locate this file with ***CdSearchFile*** or ***CdReadFile***, then use the information to avoid the need for subsequent calls.

# *Types of CDROM Access • High-level*

- ***CdRead***
  - What it Does
    - Begins reading the specified number of sectors from the current position on the CD into the specified memory buffer.

# *Types of CDROM Access • High-level*

- ***CdRead*** Performance Considerations
  - Non-blocking
    - Read command is issued to hardware, function returns to allow your code to continue execution while the hardware operation is happening.
  - Uses CdlPause When Finished
    - Significant delays if reading many small pieces of data
  - Uses ***CdReadyCallback***
    - Don’t install your own routine until read operation finished
  - Uses ***CdSyncCallback***
    - Don’t install your own routine until read operation finished

# *Types of CDROM Access • High-level*

- ***CdRead*** Error Detection & Correction
  - Performs retries automatically
    - Individual sectors retried up to 8 times
    - Entire operation retried up to 4 times

# *Types of CDROM Access • High-level*

- ***CdReadFile***
  - What it Does
    - Locates a specific file by name and reads it into memory at the specified address
  - Performance Considerations
    - Uses ***CdSearchFile*** & ***CdRead***
  - When to use
    - Good for reading long files located in the root directory or perhaps 1 subdirectory level down from the root directory

# *Types of CDROM Access • High-level*

- ***CdReadSync*** & ***CdReadCallback***
  - int *status* = ***CdReadSync***( int *mode*, char **result* );
  - *previousfunc* = ***CdReadCallback***( *newfunc* );
    int (*newfunc* *)(int *status*, char **result* )

# *Types of CDROM Access • High-level*

- ***CdReadSync*** & ***CdReadCallback***
  - What They Do
    - ***CdReadSync*** waits for the operation initiated by the ***CdRead*** or ***CdReadFile*** to be completed. Returns error & status information.
    - ***CdReadCallback*** executes a specified routine when the ***CdRead*** or ***CdReadFile*** operation is completed. Error & status information passed to specified routine as arguments.

# *Types of CDROM Access • High-level*

- ***CdReadSync*** & ***CdReadCallback***
  - When to Use
    - After calling either ***CdRead*** or ***CdReadFile***, before calling another CD-related function

# *Types of CDROM Access • Low-Level*

- ***CdControl***
  - What it Does
    - Sends low-level commands to the CD controller
    - Obtains CD controller status
  - How it works
    - Input parameters for command and command-specific data
    - 8-byte Output array returns status & error codes and any returned data

# *Types of CDROM Access • Low-Level*

- ***CdControlB***
  - What it Does
    - Blocking version of ***CdControl.***
    - No need to use ***CdSync.***
      - ***CdSyncCallback*** still active
  - When To Use
    - When no other processing can be done while waiting for CD operation to complete.

# *Types of CDROM Access • Low-Level*

- ***CdControlF***
  - What it Does
    - No handshake version of ***CdControl***
      - Issues commands to CD controller without any handshaking.
      - Does not return result code
  - When To Use
    - Only when you're positive CD isn't already busy

# *Low-Level CD-ROM Primitive Commands*

- ***CdControl*** Primitives
  - Initialization
    - CdlReset — Reset CD Controller
    - CdlSetmode — Set transfer speed (single / double-speed), ADPCM & CD-DA playback, & other parameters.
  - Error Detection / Status Request
    - CdlNop — Do nothing but return status

# *Low-Level CD-ROM Primitive Commands*

- ***CdControl*** Primitives
  - Reading
    - CdlReadN — Start data reading, retry on errors
    - CdlReadS — Start data reading, no retry on errors

# *Low-Level CD-ROM Primitive Commands*

- ***CdControl*** Primitives
  - Seeking
    - CdlSetloc — Set position for next seek
    - CdlGetlocL — Determine current logical location
    - CdlGetlocP — Determine current physical location
    - CdlSeekL — Move laser to new position (logical)
    - CdlSeekP — Move laser to new position (physical)
    - CdlForward — Fast-forward
    - CdlBackward — Rewind

# *Low-Level CD-ROM Primitive Commands*

- ***CdControl*** Primitives
  - Stopping
    - CdlPause
    - CdlStandby
    - CdlStop

# *Low-Level CD-ROM Primitive Commands*

- ***CdControl*** Primitives
  - CD Digital Audio
    - CdlPlay — Play CD-DA
    - CdlMute — Mute CD-DA sound, continue playback
    - CdlDemute — Restore CD-DA sound
    - CdlSetFilter — Specify XA-ADPCM sound playback channel

# *Low-Level CdControl Primitive Commands*

- CdlReadN
  - What it Does
    - Low-level command to read from CD, retries if error detected
  - When To Use
    - When reading important data & program code
    - When maintaining a constant data rate is not primary concern

# *Low-Level CdControl Primitive Commands*

- CdlReadS
  - What it Does
    - Low-level command to read from CD, no retry on error detect
  - When To Use
    - When constant data rate is more important than data integrity (i.e. when streaming audio and/or video)

# *Low-Level CdControl Primitive Commands*

- CdlSetloc
  - What it Does
    - Low-level command to set target position used by a command to follow (CdlPlay, CdlReadN, CdlReadS, CdlSeekP, or CdlSeekL)
  - When To Use
    - Whenever you need to specify the location of the data you wish to read

# *Low-Level CdControl Primitive Commands*

- CdlSeekP & CdlSeekL
  - What They Do
    - Low-level commands to move laser to position specfied using CdlSetloc

# *Low-Level CdControl Primitive Commands*

- CdlSeekP & CdlSeekL
  - When To Use
    - Normally used when pre-seeking to disc location in anticipation of a forthcoming data read to reduce the seek delay when reading actually begins.
    - Use CdlSeekL only for reading data sectors, uses sector header to aid seek precision
    - Use CdlSeekP for both data sectors and audio sectors without sector headers

# *Low-Level CdControl Primitive Commands*

- CdlNop
  - What it Does
    - Low-level command to return CD controller status
  - When To Use
    - When you need to determine status of on-going operation

# *Low-Level CdControl Primitive Commands*

- CdlPause
  - What it Does
    - Low-level command to pause CD reading.
    - Temporarily halts reading and / or playback, leaves head in current position in standby mode.

# *Low-Level CdControl Primitive Commands*

- CdlPause
  - When To Use
    - Only when no more data reads or seeks are required
    - Avoid using in the middle of a block of reads.
  - Performance Considerations
    - Pause/Unpause takes about 200ms or more.
      - Pausing once per second can cut throughput by over 20% (data transfer rate goes from 300kps to 240kps or less)

# *Low-Level CdControl Primitive Commands*

- CdlStandby
  - What it Does
    - Same as CdlPause except waits with spindle motor still rotating.
  - When To Use
    - Avoid using in the middle of a block of reads.

# *Low-Level CdControl Primitive Commands*

- CdlStop
  - What it Does
    - Same as CdlStandby, except also halts spindle motor.
  - When To Use
    - Avoid using in the middle of a block of reads.

# *Types of CDROM Access • Low-Level*

- ***CdSync*** & ***CdSyncCallback***
  - int *status* = ***CdSync***( int *mode*, char **result* );
  - *previousfunc* = ***CdSyncCallback***( *newfunc* );
    int (*newfunc* *)(int *status*, char **result* )

# *Types of CDROM Access • Low-Level*

- ***CdSync*** & ***CdSyncCallback***
  - What They Do
    - ***CdSync*** waits for the operation initiated by *CdControl* to be completed. Returns error & status information.
    - ***CdSyncCallback*** executes a specified routine when the ***CdControl*** operation is completed. Error & status information passed to specified routine as arguments.
  - When To Use
    - After calling a ***CdInit*** or ***CdControl***, and before calling another CD-related function

# *Types of CDROM Access • Low-Level*

- ***CdSync*** & ***CdSyncCallback***
  - The *status* value returned by ***CdSync*** or passed to the routine installed via ***CdSyncCallback*** indicates what happened with the command.
    - CdlComplete
      - Execution of command is completed
    - CdlNoIntr
      - Command is still being executed
    - CdlDiskError
      - An error of some kind occured.

# *Types of CDROM Access • Low-Level*

- ***CdReady*** & ***CdReadyCallback***
  - *status* = ***CdReady***( *mode*, *result* );
  - *previousfunc* = ***CdReadyCallback***( *newfunc* );
    int (*newfunc* *)(int *status*, char **result* );

# *Types of CDROM Access • Low-Level*

- ***CdReady*** & ***CdReadyCallback***
  - What They Do
    - ***CdReady*** either polls or waits for a sector becoming available in the CD controller's sector buffer
    - ***CdReadyCallback*** installs a user-specified routine to be executed when a sector becomes available.
  - When To Use
    - Whenever using low-level read data commands, to determine when data must be transferred from sector buffer.

# *Types of CDROM Access • Low-Level*

- ***CdReady*** & ***CdReadyCallback***
  - The *status* value returned by ***CdReady*** or passed to the routine installed via ***CdReadyCallback*** indicates what happened when the sector was read.
    - CdlDataReady
      - Data preparation is complete, data in CD controller sector buffer
    - CdlNoIntr
      - Command is still being executed
    - CdlDiskError
      - An error of some kind occured.

# *Types of CDROM Access • Low-Level*

- ***CdGetSector***
  - What it does
    - Transfers sector data from the CD controller buffer to a specified location in RAM.
  - When To Use
    - When ***CdReady*** indicates a sector is available or during your ***CdReadyCallback*** function.
  - Special Considerations
    - Blocks execution until finished, uses high-priority DMA transfer mode

# *Types of CDROM Access • Low-Level*

- ***CdDataSync*** & ***CdDataCallback***
  - *dataready* = ***CdDataSync***( *mode* );
  - *previousfunc* = ***CdDataCallback***( *newfunc* );
    int (*newfunc* *)(void);

# *Types of CDROM Access • Low-Level*

- ***CdDataSync*** & ***CdDataCallback***
  - What They Do
    - ***CdDataSync*** either polls or waits for the DMA transfer of sector data to RAM initiated by ***CdGetSector*** to be completed.
    - ***CdDataCallback*** installs a user-specified routine to be executed when the DMA transfer is completed.

# *Types of CDROM Access • Low-Level*

- ***CdDataSync*** & ***CdDataCallback***
  - When To Use
    - The ***CdDataSync*** function is essentially is not really necessary since ***CdGetSector*** is blocking function anyway.
      - May go away in library v3.6.
    - Use ***CdDataCallback*** when you must access sector data as quickly as possible after it's transferred to RAM.

# *High-Level -VS- Low-Level*

- High Level is quicker and easier to code
  - ***CdReadFile*** locates & reads file in one call.
  - ***CdRead*** handles all sector-level error correction & retries.
  - ***CdSearchFile*** easiest way to locate file position.
  - Good choice when reading large data files, or when absolute peak performance not required.
    - Does it really matter if it takes 4 seconds to load a new level instead of 3.75?

# *High-Level -VS- Low-Level*

- Low level gives better performance.
  - Avoid pausing with ***CdRead***, ***CdReadFile***, & ***CdSearchFile***
  - Avoid excessive seeking with ***CdSearchFile***
- Allows more efficient, application-specific error detection & correction
  - Retry individual sectors or ignore errors according to importance of the data.

# *Streaming -VS- Normal Data Reading*

- Streaming is designed for video and/or audio playback directly from CD.
  - Typically uses a ring buffer large enough to contain no more than a second or two of data.
    - Data may be used directly from ring buffer and discarded when finished or copied out for additional use.
- Multiple types of data may be interleaved together (video, audio, 3D animation data, program code, etc.)

# *Streaming -VS- Normal Data Reading*

- Most of the streaming library calls are for managing the data buffer, not CD access.
  - ***StSetStream*** sets streaming parameters
  - ***StSetRing*** sets data buffer

# *Streaming -VS- Normal Data Reading*

- Streaming uses ***CdRead2*** for CD access
  - ***CdRead2*** uses ***CdDataCallback*** and ***CdReadyCallback*** when streaming mode bit set.
    - ***CdRead2*** installs ***StCdInterrupt*** routine to handle data transfers
      - Automatically inserts data into ring buffer specified using Streaming Library calls
      - XA-ADPCM audio is automatically sent to audio subsystem without going through main memory

# *CDROM Error Handling*

- The return codes from functions ***CdRead***, ***CdReadFile***, and ***CdControl*** is not the CD error status
  - These functions return “command was issued OK” status, not result of the operation.

# CDROM Error Handling

- With ***CdRead*** and ***CdReadFile***, use error status array from ***CdReadSync*** or ***CdReadCallback***.
  - ***CdRead*** retries entire operation up to 4 times before failing and bailing.
  - ***CdRead*** retries individual sectors up to 8 times.

# CDROM Error Handling

- With ***CdRead*** and ***CdReadFile***, use error status array from ***CdReadSync*** or ***CdReadSync*** callback.
  - If you get an error code in the ***CdReadSync*** result array
    - Retry the ***CdRead*** operation a maximum of 4 or 5 times.
    - If still fails, throw up a “disc is dirty or damaged” screen.
    - Offer user chance to save game to memory card.
      - This procedure may be added to global technical standards list

# *CDROM Error Handling*

- To obtain error information for commands issued using ***CdControl:***
  - Use ***CdReady*** to get error status array for read operations
  - Use ***CdSync*** to get error status for operations such as ***CdControl***(CdlSeekP)
  - Do ***CdControl***(CdlNop) to get current status of CD.

# *CDROM Error Handling*

- Using low-level CD access allows you to detect errors on individual sectors.
  - Less overhead for retry.
    - Don’t have to retry entire read.
  - With some data, errors may be ignored if cannot afford time to retry.
    - Minor glitches due to errors better than game stopping.
    - Video frames can be thrown out occasionally.
    - Audio may pop or have noise.

# *CDROM Error Handling*

- The CD is NOT a hard disk.
  - Errors don’t always, or even usually, mean that the media is damaged.
  - Soft Errors -VS- Hard Errors.
    - Some errors are because CD controller is still busy with previous function.
  - Always Use ***CDSync***.
    - Most CD commands are non-blocking.
    - Important to make sure previous operations are complete.

# *CDROM Error Handling*

- Avoid End of Data Seek errors.
  - Seeking to position within last 3 minutes of data can cause laser to get lost.
  - Solutions:
    - Place 3 minutes of data at the end of your disc.
      - Dummy data file.
      - Movie file or CD-DA track.
      - If no seeks are done into middle of data, there's no problem seeking to the start of the data.

# *CDROM Error Handling*

- Avoid Dangling Callbacks
  - Some CD-related callback functions ARE NOT reset by the ***ResetCallback*** function.
    - ***CdReadyCallback***
      - Reset using ***CdReadyCallback***(0)
    - ***CdSyncCallback***
      - Reset using ***CdSyncCallback***(0)
    - ***CdReadCallback***
      - Reset using ***CdReadCallback***(0)

# *Optimizing CD Throughput*

- Use Data Compression.
- Avoid Pausing or Stopping the CD.
- Avoid use of ***CdSearchFile***.

# *Optimizing CD Throughput*

- Avoid use of ***CdRead*** & ***CdReadFile*** for reading many small files at once.
- Use low-level routines & callback functions to do CD access.
- Arrange files on your CD to avoid long seeks.

# *Optimizing CD Throughput*

- Place multiple copies of important files at various locations throughout your disc to reduce seek time.
- Use pre-seek trick.

# *Optimizing CD Throughput*

- Use data compression.
  - Use the MDEC to compress still graphics images.
  - Use lossless Huffman encoding or RLE compression to compress program code and data.
  - Use ADPCM for audio data.

# *Optimizing CD Throughput*

- Avoid pausing or stopping the CD.
  - Combine data & read in big chunks, not little ones.
  - Avoid use of ***CdRead, CdReadFile,*** and ***CdSearchFile*** functions, all of which pause the CD.

# *Optimizing CD Throughput*

- Avoid use of ***CdSearchFile***.
  - Slow.
  - Uses 1 seek & 1 CdlPause per directory level.

# *Optimizing CD Throughput*

- Avoid use of ***CdRead*** & ***CdReadFile*** for small files.
  - Uses CdlPause at end of file.
  - Combine multiple smaller files to improve read performance.

# *Optimizing CD Throughput*

- Use low-level routines & callback functions to do CD access.
  - Avoids application blocking.
  - Allows application-specific error checking & correction.
    - Don’t retry when errors occur while streaming data.
      - Constant data rate is more important when streaming.

# *Optimizing CD Throughput*

- Arrange Files On Your CD to avoid long seeks.
  - Seeks at outside of disc are faster.
    - Place streaming movie files or audio at inside of disc.
      - You don’t seek into the middle of these types of files, so reserve the rest of the disc for data that requires seeking.

# *Optimizing CD Throughput*

- Short Seeks Are Faster
  - Seeks within a range of +/- 128 revolutions can be done by changing laser angle instead of moving head.
    - Fast seek range is about +/- 2880 sectors at outer perimeter
      - 75 sectors per second
      - Rotation speed 200 RPM at outer perimeter (3.333333 rotations/sec.)
      - 128 rotations * (75 / 3.33333) = 2880

# *Optimizing CD Throughput*

- Short Seeks Are Faster
  - Seeks within a range of +/- 128 revolutions can be done by changing laser angle instead of moving head.
    - Fast seek range is about +/- 1152 sectors at inner perimeter
      - 75 sectors per second
      - Rotation speed 500 RPM at inner perimeter (8.333333 rotations/sec.)
      - 128 rotations * (75 / 8.33333) = 1152

# *Optimizing CD Throughput*

- Place multiple copies of important files at various locations throughout your disc to reduce seek time.
  - Very effective if the extra disc space is available.
  - Good for code overlays… not much space required.

# *Optimizing CD Throughput*

- Use Pre-seek Trick.
  - At end of previous read, seek to position that precedes next read and/or seek.
  - Next read will require minimal seek.
  - Avoid need to use CdlPause.
    - Estimate time before next read or seek and seek that much earlier on disc.

# *The End*