Software Development Seminar

CD-ROM (Advanced)

Improving CD-ROM access speed

CD-ROM access speed topics

* Decreasing the load from the CD-ROM access itself

* Decreasing the load on foreground processes

Other topics (not covered in this seminar)

- Data compression

- Seek time

CD-ROM Access

1) Seek time

2) Read time

Read time is fixed

↓

The important thing is to decrease the frequency and time for seeks

Multiple reads vs. Continuous reads

Continuous reads
=> 150 or 300 KB/sec

Multiple reads
=> generates seeks and decreases performance

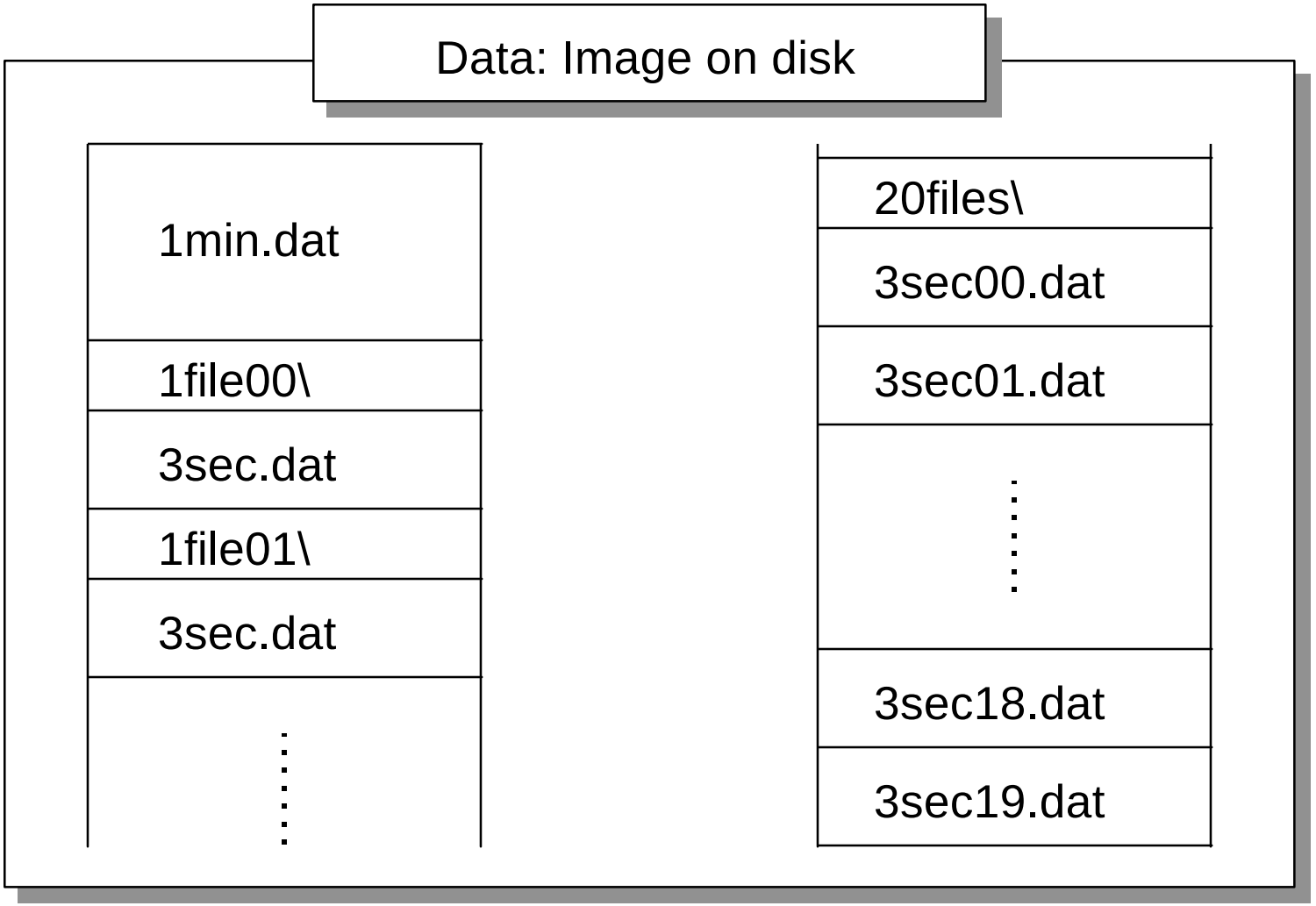
SAMPLE PROGRAM 1

Data

00: 1min.dat ... 1 minute of data

01: 1file00\3sec.dat ... 3 minute of data
02: 1file01\3sec.dat (20 units)
~
19: 1file18\3sec.dat
20: 1file19\3sec.dat
21: 20files\3sec00.dat ... 3 seconds of data
22: 20files\3sec01.dat (20 units)
~
39: 20files\3sec18.dat
40: 20files\3sec19.dat

SAMPLE PROGRAM 1



SAMPLE PROGRAM 1

Program

- Read size
  - a. 25 sectors (=50KB) 20 files 01-20
  - b. 500 sectors (=1MB) 00

- Interface
  - High-level (CDRead)

- Command issue
  - Control with status machine every 1v

Multiple reads vs. Continuous read (Results)

Multiple reads ... 500 (V)

Continuous read ... 220 (V)

It takes more than twice the time.
There are other factors such as the status machine, but the difference is obvious.

Keep files together!

CdSearchFile

- Utility function which finds the location of files

- Uses CdRead() internally

- Uses a cache mechanism
  => Caching is performed in directory units

Using CdSearchFile vs. Directly specifying sectors

Directly specifying sectors
=> seeks immediately

Using CdSearchFile
=> Excess reads are performed to find location

SAMPLE PROGRAM 2

PROGRAM

- Read size

a. 25 sectors (=50KB) * 20 files 01 - 20
b. 500 sectors (=1MB) 00
- Interface
    High-level (CdRead)
- Command issue
    Control with status machine every 1v

-File location search
    a. yes
    b. no

CdSearchFile vs. Directly specifying sectors (Results)

Using CdSearchFile ...700 (V)

Directly specifying sectors ... 500 (V)

A read for location info is generated for each file.

Even though it is a single sector read, there is overhead in seeking and pausing.

Specify file locations directly!

CdSearchFile vs. Directly specifying sectors (Notes)

CdSearchFile performs caching in directory units
=> File searches within the same directory are quick

Program

- Read size
  c. 25 sectors (=50KB) * 20 files 21-40

CdSearchFile vs. Directly specifying sectors (Notes)

Using CdSearchFile ... 520 (V)
(OnCache)

There is initially only one read generated for location info.
Thus, the result is almost the same as
directly specifying the sector.

If searching, do it on cache!

Performing CdSearchFiles each time vs. all at once

Search locations all at once at the beginning
=> Initially slow, but then is fast

Search for location performed each time
=> Slow if no caching is performed

SAMPLE PROGRAM 3

Program

- Read size
  - a. 25 sectors (=50KB) * 20 files 01-20
  - b. 500 sectors (=1MB)
- Interface
  - High-level (CdRead)

- File location searches
  - a. yes (each time)
  - b. no (all at once)

Performing CdSearchFiles each time vs. all at once

CDSearchFile each time ... 700 (V)

CDSearchFile all at once ... 300 + 500 (V)

Searching all at once is fairly costly.
Performing a single initial search and then
reading is best (although this depends on the position).

Perform CdSearchFile once at the start!

CDSearchFile (Reference)

- Begin by getting the location information for the directory from the path table

⬇ (as much as will fit in one sector, with a max of 128)

- Find the directory for the file to be found

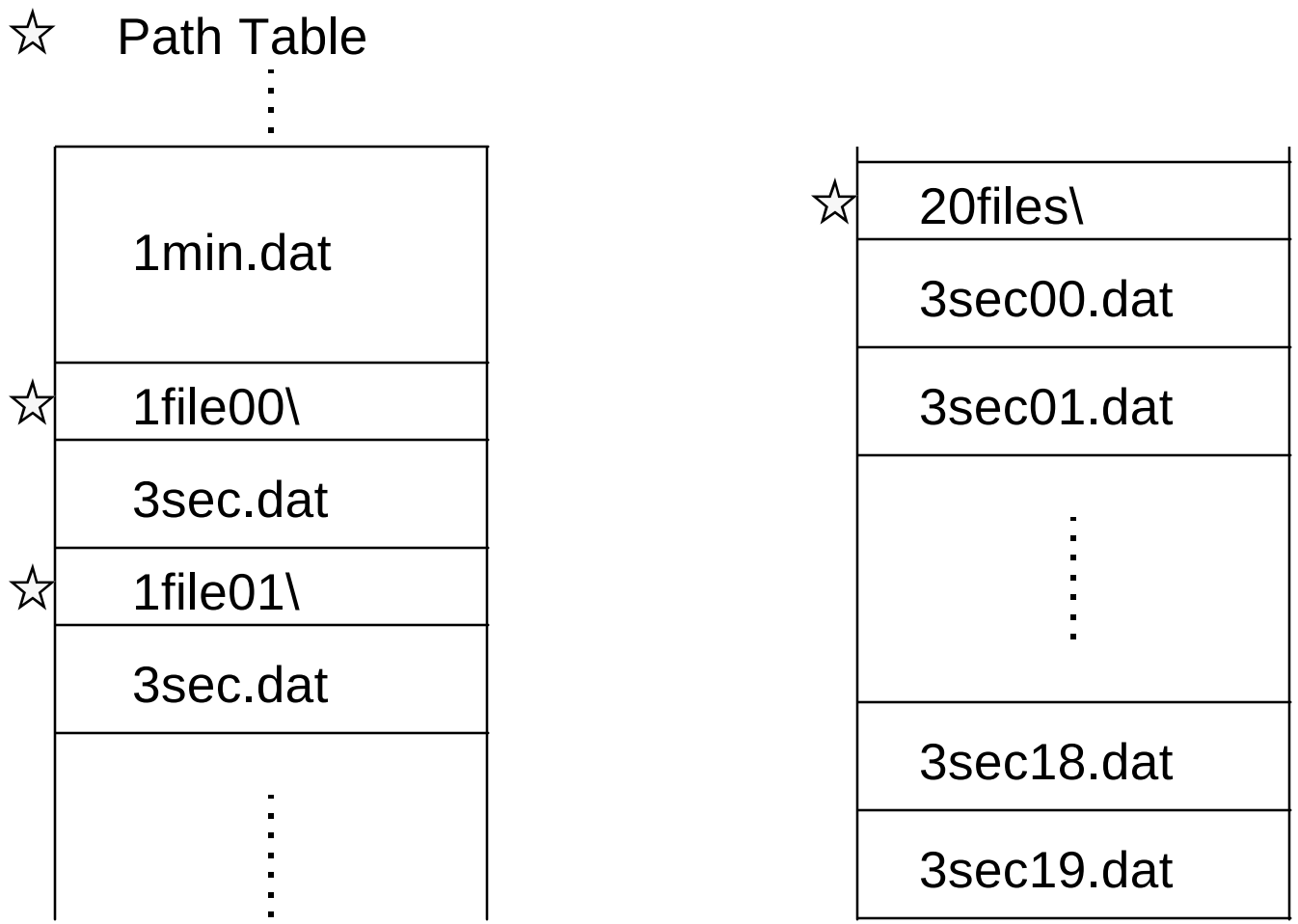
⬇

- Get the file location information for that directory

⬇

- Find the file and get its location

CdSearchFile (Reference)



Seek

1) Seeks accompanied by displacement of the pickup

2) Seeks that need only a change in the lens angle

Operation **1)** requires that operation **2)** be performed

↓

Seeks that only require a change in lens angle are considerably faster

Seek (Reference)

Range for seeks requiring only a lens angle change:
up to +/- 128 tracks from the current track

==> Inner perimeter: approximately 15 seconds,
Outer perimeter: approximately 38 seconds

[Explanation]
CDs are accessed at a constant linear velocity, so rotation speed varies according to position.

==> Inner perimeter: 500 rpm,
Outer perimeter: 200 rpm

a. 500 rpm -> 8.3 rotations/sec
b. 75 sectors per second

The sector count for 128 tracks can be determined from a and b

==> 128 * 75 / 8.3 = 1156 (sectors)

Reading without sorting vs. Reading with sorting

Sorting by position
=> Seek distance is shorter

No sorting
=> Seek distance is longer

SAMPLE PROGRAM 4

Program

-Read size

a. 25 sectors (=50KB) * 20 files 01-40

b. 25 sectors (=50KB) * 20 files 01-40

-Interface

High-level (CdRead)

- Command issue

Control with status machine every 1v

- File position sorting

a. yes

b. no

Reading without sorting vs. Reading with sorting (Results)

Reading without sorting	... 700 (V)
Reading with sorting	... 560 (V)

With no sorting there were more cases with excess loading because seeks involved pickup displacement.

Sort when reading multiple files!

CdRead

- High-level interface

- Performs retries internally

- Overhead involved in issuing command

CdReady callback

The CD-ROM subsystem has 1 sector's worth of local memory. Once data is placed in the sector buffer, an interrupt is generated.

This triggers the generation of a callback.

=> CdReady callback

CdGetSector() transfers the data from the sector buffer to main memory (DMA).

CdData callback

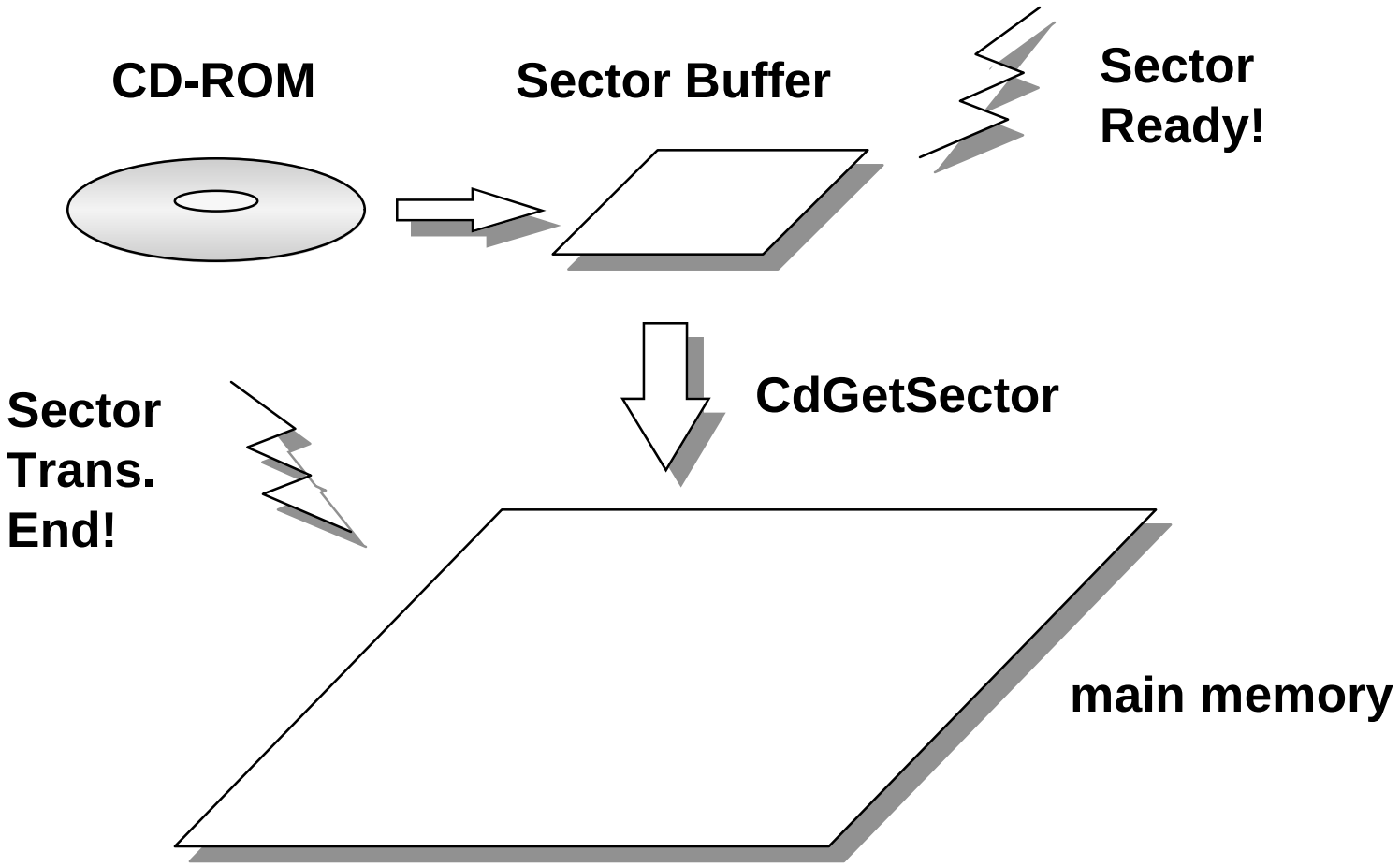
CdGetSector can be used to DMA transfer the data from the sector buffer to main memory. The completion of this operation can be used to trigger a callback.

=> CdData callback

This does not need to be used in normal data reads. However, it is necessary for systems performing real-time processing in sector units.

(This topic is not covered in this seminar.)

Data flow


CD-ROM
Sector Buffer
Sector
Ready!
CdGetSector
Sector
Trans.
End!
main memory

Using CdlReadN instead of CdRead

CdlReadN (low-level)
==> suitable for background processes

CdRead (high-level)
==> not suitable for background processes

SAMPLE PROGRAM 5-1

Program

- Read size

  25 sectors (=50KB) * 20 files 01-20

-Interface

  Low-level (CdlReadN)

  Using CdReady callback

- Command issue

  Control with status machine every 1v

Using CdlReadN instead of CdRead (Results)

Using CdlReadN	... 560 (V)

The overall time is somewhat slow due to the load from the status machine. However, the load on the foreground process is low since there is no overhead when a read operation is started.

Use low-level processes in the background!

CdSync callback

When a command is issued to the CD subsystem, the result of the operation is posted through an interrupt.

A callback can be generated based on this trigger

==> CdSync callback

Using this technique, it is possible to chain commands.

Improving the status machine

Using CdSync

==> Synchronized with the subsystem

Calling at fixed intervals

==> Not synchronized with the subsystem

SAMPLE PROGRAM 5-2

Program

- Read size

  25 sectors (=50KB) * 20 files 01-20

-Interface

  Low-level (CdlReadN)

  Uses CdReady,CdSync callback

- Command issue

  Control with CdSync callback status machine

Improving the status machine (Results)

Using CdSync callback ... 500 (V)

CD Because of synchronization with the CD subsystem, the overall time is less when compared to the case before any improvement.

Synchronize with the subsystem!

Handling open shell / Titles with multiple CDs

Opening/closing of the shell

In the debugging station and in the real machine, a DISC check (for copy protection) is performed when the shell is opened or closed.

During this time, commands issued to the subsystem are ignored and status is undefined.

DISC check

Time required for DISC check

↓

Actual unit: Maximum of approximately 3 seconds

Debugging station: Maximum of approximately 12 seconds

During DISC check there should be no commands issued to the subsystem.
However, waiting should not be based on the clock.

Use **CdlGetTN**

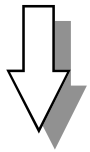
Differences with the development environment

With the DTL-H2000, DISC checks are not performed.
Therefore the CD stops rotating after the shell has been opened or closed
=> The processing of the first command after that is slow

Since rotation is stopped, the CD must first spin up.
The unit must wait until rotation is stable.

Checking for opening/closing of the shell

The open/close state of the shell is
determined by the status returned from the subsystem.



```
if (result[0] & CdlStatShellOpen) {
        ...         // Shell was Opened.
}
```

SAMPLE PROGRAM 6

Program

- Play CD-DA

- Command issue
  Control with status machine every 1v
- Check for opening/closing of shell
  and perform a retry

Titles with multiple CDs

The status from the subsystem is checked and the open/close state of the shell and spindle rotation are monitored.
Specific file/sector contents are used to determine that the disc has been replaced.

Titles with multiple CDs

1 Message: "Please insert DISC2"

↓

2 Wait for shell to open

↓

3 Wait for shell to close

↓

4 Wait for DISC check to finish

↓

5 Check for presence of DISC (CdlStatStandby in status flag)

OK NO => 1

↓

6 Check DISC2

OK NO => 1

↓

7 Done

* 4, 5 need to be performed simultaneously