MS-Dosifying Your CD-ROM

Most of the following guidelines apply to the present version of the Microsoft CD-ROM Extensions. Future versions of the Extensions are expected to support many of the items listed below that are presently best avoided. The behavior of the Extensions with fields and records that are currently ignored might change at any time.

Correctness

Make sure that your disc is in valid High Sierra format. *Nothing* is guaranteed if your disc does not have a valid format. Surprisingly enough, we have received several discs that have one or more invalid formatted data areas. Some of the formatting errors included directories being sorted incorrectly, incorrect path table sizes, incorrect directory file sizes, directories missing from the path table, and invalid directory names. In almost every case, the Extensions will behave incorrectly, and at worst, the system might crash.

In addition to running validation software to verify the High Sierra image, you should also verify that the Extensions work with your CD-ROM disc and application software before distributing it. Unfortunately, it may not matter if your disc is correct and the Extensions are wrong if they don't work together. Please report any and all problems you think are in the Extensions to Microsoft so that they can be fixed.

Path Table and Directory Sizes

Using invalid sizes for the path table and directory are a common error when creating CD-ROMS for MS-DOS. The following guidelines apply to path tables and directories:

- MS-DOS directory file sizes are always a multiple of the logical sector size—2 kilobytes.
- Path table sizes are always the exact number of bytes contained in the path table (which is typically not a multiple of 2 kilobytes).
- You must not have any blank directory sectors.
- The directory length must reflect the correct length of the directory file.
- Directory sectors always begin on a logical sector boundary.

8.3 Filenames

MS-DOS cannot handle longer than 8.3 filenames (8 characters for the filename and 3 characters for the filename extension). If the CD-ROM filename is longer than 8.3, then the filename will be truncated. If this happens, two files that are not

unique within 8.3 characters map to the same filename. For example, both FILENAME1.TXT and FILENAME2.TXT will appear as FILENAME.TXT. Kanji filenames are also limited to 8.3 or 4.1 Kanji characters. Only shift-Kanji filenames are recognized at present. To get Kanji, you must specify a supplementary volume descriptor indicating you have Kanji filenames. Contact Microsoft to find out how this is done.

Record Formats

The Extensions do not support any record formats so the Extensions will ignore the RECORD bit in the file flags byte in the directory entry.

Interleaving

In the present version, the Extensions do not support interleaving so if the interleave size and interleave factor are non-zero, the file will ignore these fields and return erroneous data.

Multi-Extent Files

Multi-extent files are not supported in the present version. Each extent of a multi-extent file will appear as a separate file with the same name.

Multi-volume Disc Sets

Multi-volume disc sets are not supported in the present version. Directories that are located on another volume could potentially cause the Extensions to crash if searched and erroneous data will be returned for files that are located on another volume.

Coded Character Sets

Only one coded character set or supplementary volume descriptor is recognized in the latest version. This is for shift-Kanji.

Version Numbers and Associated Files

Version numbers and associated files are not supported by the Extensions. The Extensions will strip the version string off the end of the filename so that two identical filenames with different versions will appear to have the same name. There is no way to specifically ask for any but the first instance of that filename. Two files with the same name and different version numbers have the same accessing problem as two files with longer than 8.3 filenames that have been truncated to the same filename. In the case of associated files, they are not accessible by any MS-DOS function.

Protection

Protection bits are not used on MS-DOS. If the protection bit is set in the file flags byte in the directory entry for a file, it is ignored and normal access is allowed.

No XAR Support

At present, the Extensions ignore the contents of any XAR record.

Motorola Format Tables

The additional copies of the path table and any values in “Motorola” format (most significant bytes using the lowest address values) are ignored at present. MSCDEX only pays attention to “Intel” formatted values. They should be included though for portability sake.

Multiple Copies of the Path Table

The Extensions presently only read and use the first copy of the path table. Later versions may check to see that copies of the path table agree.

Additional Volume Descriptor Records

Boot records and unspecified volume descriptors are ignored. The first standard volume descriptor found is the one that is used. Additional copies are ignored at present.

File Flags

The existence bit is treated the same as the hidden bit on MS-DOS. Some other operating systems may not handle the existence bit so you might not want to use it if you are also developing for these systems.
The directory bit for High Sierra is treated the same as the directory bit in MS-DOS.
Files with the protection bit set are not found when searched for or opened.
None of the remaining bits (Associated/Record/Multi-extent/Reserved) are handled at present. Using files with these bits set will have undefined behavior.

Unique Volume Identifiers

It is highly recommended that the volume identifier be unique. The Extensions use the volume identifier to do volume tracking and to double-check if the disc has changed. Using unique volume identifiers for each CD-ROM product and version minimizes the chances of a user having two discs with the same volume identifier. This will increase the effectiveness of the checks the Extensions perform to determine if the user has changed the CD-ROM.

Your application should also use the volume label to tell if the CD-ROM disc has changed. Your application can obtain the volume label for a CD-ROM on MS-DOS from the volume identifier field in the primary volume descriptor. The call to get the volume label is very inexpensive to make once the CD-ROM has been initialized and will cause no disc I/O to be done unless the media has changed. This is the best way for your application to tell if the disc it wants is in the drive. Your application should not communicate with the driver or drive to determine if the media has changed. Doing this can prevent the Extensions from learning that the disc has changed and from performing the initialization it needs for a new disc.

Many Small Directories or A Few Large Directories

As a rule, it is better to have many small directories that contain fewer files than one very large directory. However, the exact answer depends on your application's behavior

because if you try very hard, you can thrash almost as badly with many small directories as you can with one large directory.

What makes the difference? For example, suppose you have 1000 directories with 40 files. On average, you'll read about one sector per file open and scan one-half of it. On the other hand, you could have 1 directory with 4000 files. On average, each file open in this large directory (about 100 sectors) will involve scanning about 50 sectors to open that one file. As long as it is very inexpensive to get to each directory through the path table, it is much better to have many small directories.
You can get further improvements by grouping files that are related and are opened together in each of these subdirectories. As you open each successive file, the directory sector is very likely in the disc cache. This helps minimize the reads from the CD-ROM disc. Putting each file in a separate subdirectory is extreme and might increase the access time. With this strategy you never gain the benefits of locating the next file in a directory sector that has already been cached and you needlessly enlarge the path table.
The size of the path table also limits how many subdirectories you might want. If there are too many you might thrash reading the path table sectors. Each path table sector holds pointers to 100 to 200 directory files. (The number of pointers depends on the length of the directory names.) If you have a path table that is 10 sectors long, you will want at least 10 sectors of memory buffers to hold the path table. If you use less than 10 sectors you risk re-reading sections of the path table on every file open which is very costly.

The most important point from this discussion is that you can vastly improve your file open speed by making sure you have enough memory buffers. If you are repeatedly trying to scan a 10 sector directory file (approximately 400 entries) and you only have 4 sectors in the sector cache, the cache is going to work against you because you will end up churning it excessively. If you allocate 14 sectors for this example (/M:14), then the whole directory file will find its way into the cache and you will stop hitting the disc. The difference in speed may be several orders of magnitude. A safe recommendation is to reserve as many sectors as there are in the path table plus the number of sectors for the largest directory plus two. The last two sectors are reserved for data sectors and internal dynamic data. Multiple drives complicate this formula because the buffers are not tied to specific drives and are shared, and because not all drives are active at the same time.
Also, do not rely on the file system to do your searching for you. If you are performance conscious, finding a chunk of data by looking for it with a file name through the file system is expensive. Most of the time, locating data through the file system is fine because the cost is a single one-time operation. If this is repeated often enough, it may pay to do some of the work yourself. A better method is to place everything into one big file and cache your own hierarchy, indexing, binary trees, or whatever searching scheme you choose rather than asking for the file system to locate it.

MSCDEX Extended Errors

MSCDEX issues the following extended errors that your application can trap if it chains into the INT 24h vector:

Value	Message

100	CDR100: Unknown error
101	CDR101: Not ready
102	CDR102: EMS memory no longer valid
103	CDR103: CD-ROM not High Sierra or ISO-9660 format
104	CDR104: Door open

Special Considerations When Using a Read-Only Device

When developing software that is used with both read/write and read-only devices, your application should be prepared to handle the MS-DOS function responses from a read-only device if it attempts to write to it. This section summarizes how MSCDEX responds to the common functions used to update files or directories.

INT 21h Function 3Ch - Create File
This function always sets the carry flag to indicate failure.

INT 21h Function 3Dh - Open File
This function fails with write access and when attempting an open with a non-existent file. To prevent compatibility problems, it does not fail when opening for read or for read/write access. Note that even when the file is opened for read/write access, any attempts to write to the file (INT 21h Function 40h) will fail.

INT 21h Function 4301h - Setting File Attributes
This function always sets the carry flag to indicate failure.

INT 21h Function 56h - Rename File
This function always sets the carry flag to indicate failure.

INT 21h Function 40h - Write File or Device
This function always fails. This is true even if the file was opened for read/write access.

INT 21h Function 39h - Create Directory
This function always sets the carry flag to indicate failure.

INT 21h Function 3Ah - Delete Directory
This function always sets the carry flag to indicate failure.

INT 21h Function 6Ch - Extended Open File
This function fails with write access and when attempting an open with a non-existent file. To prevent compatibility problems, it does not fail when opening for read or for read/write access. Note that even when the file is has been opened for read/write access, any attempts to write to the file (INT 21h Function 40h) will fail.

INT 21h Function 13h - Delete File (FCB and Extended FCB)
This function always sets AL = FFh to indicate failure.

INT 21h Function 16h - Create File (FCB and Extended FCB)
This function always sets AL = FFh to indicate failure.

INT 21h Function 17h - Rename File (FCB and Extended FCB)
This function always sets AL = FFh to indicate failure.

Special Considerations For Other MS-DOS INT 21h Functions

In most instances, file-based MS-DOS INT 21h functions behave identically to those used with any other media. Drive- or device-based MS-DOS INT 21h functions are a different matter. Since MSCDEX relies upon the MS-NET interface to DOS, many of these MS-DOS functions will behave as if the CD-ROM were a remote drive. With other MS-DOS functions, the MSCDEX redirected drives will not respond as a network drive would in order to preserve some characteristics of removable local media.

INT 21h Function 42h - Set File Pointer
As a note, repositioning the file pointer (seeking) does not translate directly to moving the actual CD-ROM device head location. The CD-ROM drive might not reposition the heads until data is read from the disc.

INT 21h Function 44h (IOCTL) Subfunction 08h - Check Block Device Removable
This function fails when checking for a drive supported by MSCDEX, but since it checks as being remote this is correct.

INT 21h Function 44h (IOCTL) Subfunction 09h - Check Block Device Remote
Checking for a CD-ROM drive returns as a remote drive and fails when checking for removability. This is similar to a network drive; however, note that the CD-ROM drive does not appear on the network redirection list.

INT 21h Function 47h - Get Current Directory
After a disc is ejected, this function will return the directory selected for the last current directory. Thus, this function is not a reliable indicator to determine if the media is present in the drive or if the directory is still valid. For a reliable indication, execute the Volume Label function (INT 21h Function 440Dh Minor Code 66h) prior to executing this function.

INT 21h Function 5Fh Subfunction 02h - Get Redirection List Entry
The logical drive supported by MSCDEX does not show up in this list under any

of the indexes. To this respect the CD-ROM drive behaves like a local drive; however, the drive checks as remote making it behave somewhere between a network drive and a local drive.

End.