Version 2.21

CD-ROMifying Your Software

CD-ROM is the first of what will probably be several alien file structures that will start appearing in the MS-DOS world primarily with the introduction of installable file systems under newer versions of MS-DOS. The following will attempt to outline some guidelines for writing software that will help in porting your software to these new file systems and for CD-ROM specifically.

Choice of Filename Characters

On the first Microsoft Test CD-ROM disc, the Codeview demo failed because certain filename characters that were legal on MS-DOS were not allowed according to the High Sierra file format. When the software looked for file 'S1.@@@', it wasn't found because the character '@' is illegal for High Sierra filenames and during High Sierra premastering, the file was renamed 'S1'.
Valid High Sierra filename characters are the letters 'A' through 'Z', the digits '0' through '9', and the underscore character '_'. All other characters are invalid. Note that the letters 'a' through 'z' are not included so that High Sierra file names are not case sensitive. Under MS-DOS, filenames are mapped to upper case before they are looked up so this is typically not a problem. When choosing file name characters, keep in mind the restrictions of the file structure format and the operating systems your media may be targeted towards.

Depth of Path

The High Sierra format allows for pathnames to be up to 8 levels deep. It's possible to create a path on MS-DOS that is deeper than that but you won't be able to transfer it to a CD-ROM. The following example shows valid and invalid pathnames:

```
\one\two\three\four\five\six\seven\eight\file.txt                /* Valid        */

\one\two\three\four\five\six\seven\eight\nine\file.txt /* Invalid */
```

Length of Path

The High Sierra format allows for the entire pathname to be a maximum of 255 characters. Since MS-DOS imposes a limit far lower than this, it should not present a problem. The MS-DOS call to connect to a sub-directory is limited to a directory string of 64 characters. The length of path restriction is more a concern for Xenix/Unix than MS-DOS.
Amusingly enough, for MS-DOS versions 2.X and 3.X, the MS-DOS call to create a sub-

directory allows a directory string greater than 64 characters which allows you to create sub-directories that you cannot connect to.

Unfortunately, a CD-ROM may potentially contain a pathname that is much larger than 64 characters long. This is not a concern here but is discussed in a related section—"MS-DOSifying your CD-ROM." As a rule, try to keep the length of your longest path less than 64 characters and you should be pretty safe.

Read-only Attributes

Even though most people understand that CD-ROM discs are read-only, there's still a lot of software written that assumes the current disk is always writable. For example, the Microsoft Multiplan Demo assumes that it can create and write temporary files to the presently connected drive.

To avoid this problem, try to provide another means of letting the user specify where temporary files can be created. Many applications check the environment for the variables TMP or TEMP which contain the pathname to use when creating temporary files. Most people understand this convention now. (If the TEMP directory is located on a ram-drive, the user gains an added benefit of improving the speed of accessing the information in the temporary file.) If the environment variable is not set, then the application can fall back on the assumption that the media is writable or ask where temporary files should be kept.

As a rule, for both temporary and permanent files, if a file creation error occurs, allow the user to re-specify the pathname used so that he can work around the error. The last thing that should happen is to lose work because the user was not allowed to store his work data in a valid place.

Data and Disk Format

Don't depend on the format of data on the disk. For example, CD-ROM's do not have a FAT so an application should not rely on finding one. Do not talk to any media at a physical level (reading/writing sectors) unless you expect to be media dependent (such as CHKDSK or FORMAT). MS-DOS INT 21h calls should provide everything you need to access the file contents and attributes.

Small Directories

For performance, try to keep directory sizes smaller than about 40 files. Much beyond this the directory files use more than one 2048 byte sector. Typically this is not a problem unless the number of sector buffers specified when MSCDEX is started is small and the

directory uses multiple sectors. In this case, there is a possibility of software thrashing badly when it searches a directory if it must reload sectors read previously to find an entry.

For certain pathological programs, such as certain implementations of the Xenix utility FIND, the penalty is about 1 second per directory sector that you have to scan to get to the next entry. If the directory is large, say 8 sectors, the time for FIND to scan that one directory could potentially take a half hour for something that would take less than a second if all the entries fit in the cache.

The solution for this problem is to make sure that MSCDEX never throws out of the cache what it will need next. This is accomplished by growing the cache (very easy—simply change the parameter to MSCDEX) and to make sure that the largest object that goes through the cache will not clear it out. There is a balance between having too many directories and too many files in a few directories, but the balance is heavily weighted towards many small to medium sized directories. Keep this in mind when laying out your files.

Since the penalty for using a file in the lowest subdirectory instead of the root directory is virtually nil and as more directories don't cost much, it's a good idea to break up large directories into several smaller ones. This will help avoid problems of flushing the disc sector cache. Try to keep related files close together—both in location on the CD-ROM and in the same directories. Keeping related files close together will reduce seek time when accessing them concurrently. Keeping them in the same directory will help prevent swapping out directory sectors.

Updating CD-ROM Databases and Software

Many people are interested in providing updates to files contained on a CD-ROM disc. They would like to create a directory on their hard disk will all updated files in them and have the CD-ROM Extensions look there first before searching the CD-ROM. Unfortunately, by the time the Extensions gets the request, it is very difficult for it to look for updates on the hard disk. So whatever alternative searching that is necessary will have to be done in the application software.

For this reason, it's a good idea to set your path so that it lists the directories on the hard disk first. Another good strategy is to copy executables to a directory on your hard disk. This lets you update them, and it lets them start faster. Also, have the application software search alternative hard disk directories for updates before it searches the CD-ROM. Storing both software updates and updated or commonly used database files on a hard disk will both speed performance and allow incremental updating.

Search Strategies

Try to avoid relying on the operating system in your search strategy. If your database is broken up into a hierarchy and your order is imposed through the file structure by breaking up the database into many files in a tree, then accessing data in the database is typically going to require a lot of directory reading and searching.

On a hard disk, the time involved to access this type database is not large. However, on a CD-ROM the search times can add up. Opening a file can be an expensive operation simply because the system must read the directory before it can open the file. With a fast drive, seeking to a location on a CD-ROM can take about 10 milliseconds. At worst, a seek can exceed a second on some older CD-ROM drives. Some newer drives have a worst case seek time of about half a second. If you can avoid this, your application will respond faster. MSCDEX caches as many directory sectors as it can so that searching the most active directories is very quick. However, any operations that search multiple directories once through continually clears out the cache and renders its caching ineffective.

The strategy used by *Microsoft Bookshelf* was to lump the entire database into a single file and structure the indexing so that searching used a minimum of seeks. Bookshelf uses packed b-trees with each node holding as many entries as will fit into a single sector and also cache in memory as much of the root of the tree as it can.

Combining databases avoids the overhead of repeatedly opening and closing database files. Caching as much of the indexes in memory as possible allows searching of keywords to be completed typically with a single seek.

In general, identify your software bottlenecks and through judicious use of faster storage media (either memory or hard disk) you can both have large storage and respectable performance.

Portability

One advantage of the High Sierra format is data interchangeability with other operating systems. For portability, chose a set of the High Sierra features that are supported across different operating systems to be sure you can read the disc in each of them. The lowest common denominator then (this list is not complete - see also what must be done to target MS-DOS) would need a logical block size of 512 bytes, both type L and M path tables and for all fields, single volume sets, at least one Primary Volume Descriptor and terminator. Be aware that if one of your goals is data portability, you must determine what restrictions the other operating systems might impose on the High Sierra format.

End.