How can I preallocate disk space for a file without it being reported as readable?

devblogs.microsoft.com/oldnewthing/20160714-00

July 14, 2016

Raymond Chen

A customer wanted to create a file with the following properties:

- The file has a known maximum size. (The file is a log file, and when the log file gets full, the program closes the log file and creates a new one.)
- Disk space for the log file should be preallocated up to the maximum size.
- Aside from the fact that disk space for the maximum size has been preallocated, the file should behave like a normal file: Code that reads the log file should be able to read up to the last written byte, but if they try to read past the last written byte, they should get "end of file reached".

The last requirement exists because there are third party tools that read the log files, and those tools are just going to use traditional file I/O to access the log file.

The customer suggested an analogy: "If we were operating on `std::vector`, then what I'm looking for is `vector.reserve()` to expand the vector's capacity, and `vector.push_back()` to append entries. Code that iterates over the vector or reads the `vector.size()` see only the vector elements that have been pushed onto the vector."

The file system team responded with this solution:

Use the `SetFileInformationByHandle` function, passing function code `FileAllocation-Info` and a `FILE_ALLOCATION_INFO` structure. "Note that this will decrease fragmentation, but because each write is still updating the file size there will still be synchronization and metadata overhead caused at each append."

The effect of setting the file allocation info lasts only as long as you keep the file handle open. When you close the file handle, all the preallocated space that you didn't use will be freed.

Here goes a Little Program. Remember, Little Programs do little to no error checking.

```
#include <windows.h>

int __cdecl main(int argc, char** argv)
{
  auto h = CreateFile(L"test.txt", GENERIC_ALL,
    FILE_SHARE_READ, nullptr, CREATE_ALWAYS,
    FILE_ATTRIBUTE_NORMAL, nullptr);
  FILE_ALLOCATION_INFO info;
  info.AllocationSize.QuadPart =
    1024LL * 1024LL * 1024LL * 100; // 100GB
  SetFileInformationByHandle(h, FileAllocationInfo,
    &info, sizeof(info));
  for (int i = 0; i < 10; i++) {
    DWORD written;
    WriteFile(h, "hello\r\n", 7, &written, nullptr);
    Sleep(5000);
  }
  CloseHandle(h);
  return 0;
}
```

This program creates a file and preallocates 100GB of disk space for it. It then writes to the file very slowly. While the program is running, you can do a `type test.txt` to read the contents of the file, and it will print only the contents that were written. Watch the free disk space on the drive, and you'll see that it drops by 100GB while the program is running, and then most of the disk space comes back when the program exits.

The preallocated disk space is also released when you call `SetEndOfFile`.

There's a special gotcha about setting the file allocation info: If you set the file allocation info to a nonzero value, then the file contents will be forced into nonresident data, even if it would have fit inside the MFT.

Raymond Chen

Follow