CP/M Era MBASIC String Garbage Collection

Documentation Date: October 29, 2025

Question

How did CP/M era MBASIC string garbage collection work?

Answer

CP/M MBASIC (Microsoft BASIC-80) used a string garbage collection algorithm that was shared across many Microsoft BASIC implementations of that era, including Commodore 64 BASIC, Applesoft BASIC, and others. This algorithm, while simple in concept, had significant performance characteristics that could cause problems with programs that used many string variables.

How the Algorithm Worked

The garbage collector scanned through all areas which could contain string descriptors (the string stack, simple variable area, and array area) looking for the string with the highest address. This string would be moved to the new top of the string heap, including adjustment of the string address in the string descriptor.

It then scanned the string descriptors for the next one pointing just below the current string heap top and moved that string up too so it was right below the previous relocated string, adjusting the string heap top downwards accordingly. This procedure was repeated as long as any string descriptor below the new string heap top could be found.

Thanks to this garbage collection process, all strings would end up laying adjacent on the string heap without gaps in between, eliminating wasted space.

The Performance Problem

The critical issue with this algorithm was its quadratic time complexity. For each string pointing to the string heap, the garbage collector had to scan all string descriptors. If there were N such string variables, the number of string descriptors had to be scanned N times.

This meant the time required increased with the square of the number of strings ($O(n^2)$). These scans occurred even if it turned out that a string didn't need to be moved.

Impact on users: When there were only a handful of strings, the running time of the garbage collector was negligible. However, if there were hundreds of strings being dealt with, garbage collection could take 20 minutes or far longer. During this time, the computer would appear quite dead, not even responding to the RUN/STOP key on systems like the Commodore 64.

Additional Context

Building or manipulating strings could also cause the string heap to rapidly fill up with garbage because BASIC allocated new copies of strings instead of reassigning pointers where possible.

Both of these factors created a trap for unwary programmers who would test code on a small handful of strings without realizing that a much larger set of strings would cause the program to halt for lengthy periods of time due to garbage collection.

Later Improvements

BASIC 4.0 and later versions (including BASIC 3.5 and BASIC 7.0) solved the problem of garbage collection delays by including "back pointers" in the string space. This made it possible to do the full set of garbage collections with just a single sweep, eliminating the $O(n^2)$ performance problem. However, the Commodore 64 never went beyond BASIC 2.0, so C64 programmers had to take measures to prevent garbage collection from becoming too intrusive when dealing with a large number of strings.

Similarly, Applesoft BASIC's garbage collection had the same $O(n^2)$ performance issues, with pauses ranging from a few seconds to a few minutes. A replacement garbage collector for Applesoft BASIC by Randy Wigginton identified a group of strings in every pass over the heap, reducing collection time dramatically. BASIC.SYSTEM, released with ProDOS in 1983, provided a windowing garbage collector for BASIC that was many times faster.

Sources

• C64-Wiki: Garbage Collection

• Wikipedia: Garbage collection (computer science)

• Discussion forums on retrocomputing and CP/M systems

• Dr. Dobb's Journal archives