Real-world hash functions

Information presented here is taken from the article “Selecting a Hashing Algorithm”, B.J. McKenzie et al., *Software Practice & Experience*, vol. 20, no. 2, February 1990.

Hashing algorithms for strings

All of these algorithms compute a hash value $H$ for a string of length $n$ whose characters are $c_1, c_2, \ldots, c_n$. The hash value is determined from successive partial results $h_0$, $h_1$, $h_2$, ..., $h_n$, with each $h_k$ computed from $h_{k-1}$ as given in the formulas below. The hash table size is the value used in the mod operation at the end of each algorithm.

1. Amsterdam Compiler Kit (ACK)

   There is a "mask" for characters, built as follows:
   
   $m_1 = 171; m_k = \text{rightmost 8 bits of } 77m_{k-1} + 153$
   
   The hash value $H$ is then the last 8 bits of $h_n$, where $h_0 = 0$ and $h_k = h_{k-1} \oplus \text{XOR}(c_k, m_k)$.

2. Eidgenossische Technische Hochschule Modula-2 Cross Compiler (ETH)

   $h_0 = 1; h_k = c_k \ast ((h_{k-1} \mod 257) + 1); H = h_n \mod 1699$

3. GNU C preprocessor (GNU-cpp)

   $h_0 = 0; h_k = 4h_{k-1} + c_k; H = \text{last 31 bits of } h_n, \mod 1403$

4. GNU compiler front end (GNU-cc1)

   $h_0 = n; h_k = 613h_{k-1} + c_k; H = \text{last 30 bits of } h_n, \mod 1008$

5. Portable C Compiler front end (PCC)

   $h_0 = 0; h_k = 2h_{k-1} + c_k; H = \text{last 15 bits of } h_n, \mod 1013$

6. Unix 4.3 BSD C preprocessor (CPP)

   $h_0 = 0; h_k = 2h_{k-1} + c_k; H = h_n \mod 2000$

7. AT&T C++ compiler (C++)

   $h_0 = 0; h_k = 2h_{k-1} + c_k; H = h_n \mod 257$

8. Icon translator (Icon)

   $h_0 = 0; h_k = h_{k-1} + c_k; H = h_n \mod 128$
Performance

Algorithms were tested on 36,376 identifiers from a large bunch of C programs, and 24,473 words from a UNIX dictionary.

ACK is a loser (U-shaped distribution). Icon, C++, GNU-cc1, and GNU-cpp seem to distribute the words well. Theoretical results suggest that an algorithm of the form \( h_k = A \cdot h_{k-1} + c_k; \quad H = h_n \mod N \) will be good, with \( A \) a power of 2 for speed and \( N \) chosen appropriately. The authors note:

"[A] and N need to be selected with care. Although it may seem unlikely that anyone would choose one of the really bad combinations, the facts ... indicate that far-from-optimal choices are made and persisted with. The experiments have shown that very small variations in N can produce large variations in the efficiency of the hash-table lookup, and that the popular view, that choice of a prime number will automatically ensure a good result, is not well founded."
