hash1() — Jenkins’s one-at-a-time hash function

Description

hash1(x) returns Jenkins’s one-at-a-time hash calculated over the bytes of x; 0 ≤ hash1(x) ≤ 4,294,967,295.

hash1(x, n) returns Jenkins’s one-at-a-time hash scaled to 1 ≤ hash1(x, n) ≤ n, assuming n < . (missing). hash1(x, .) is equivalent to hash1(x).

hash1(x, n, byteorder) returns hash1(x, n) performed on the bytes of x ordered as they would be on a HILO computer (byteorder = 1), or as they would be on a LOHI computer (byteorder = 2), or as they are on this computer (byteorder ≥ .). See [M-5] byteorder() for a definition of byte order.

In all cases, the values returned by hash1() are integers.

Syntax

\[
\text{real scalar} \quad \text{hash1}(x \ [, \ \text{real scalar} \ n \ [, \ \text{real scalar} \ \text{byteorder} \ ]])
\]

where

- \(x\): any type except \text{struct} and any dimension
- \(n\): 1 ≤ \(n\) ≤ 2,147,483,647 or .; default is . (missing)
- \(\text{byteorder}\): 1 (HILO), 2 (LOHI), . (natural byte order); default . (missing)

Remarks and examples

Calculation is significantly faster using the natural byte order of the computer. Argument \text{byteorder} is included for those rare cases when it is important to calculate the same hash value across different computers, which in the case of hash1() is mainly for testing. hash1(), being a one-at-a-time method, is not sufficient for constructing digital signatures. It is sufficient for constructing hash tables; see [M-5] asarray(), in which case, byte order is irrelevant. Also note that because strings occur in the same order on all computers, the value of \text{byteorder} is irrelevant when \(x\) is a string.

For instance,

\[
\begin{align*}
: \text{hash1("this"), hash1("this",.,1), hash1("this",.,2)} \\
1 & 2385389520 & 2385389520 & 2385389520 \\
: \text{hash1(15), hash1(15,.,1), hash1(15,.,2)} \\
1 & 463405819 & 3338064604 & 463405819
\end{align*}
\]
The computer on which this example was run is evidently \textit{byteorder} = 2, meaning LOHI, or least-significant byte first.

In a Mata context, it is the two-argument form of \texttt{hash1()} that is most useful. In that form, the full result is mapped onto \([1, n]\):

\[
\text{hash1}(x, n) = \left\lfloor \frac{\text{hash1}(x)}{4294967295} \ast n \right\rfloor + 1
\]

For instance,

: \texttt{hash1("this", 10)}
6

: \texttt{hash1(15, 10)}
2

The result of \texttt{hash1}(x, 10) could be used directly to index a 10 × 1 array.

**Conformability**

\texttt{hash1}(x, n, \textit{byteorder}):

\begin{align*}
x & : r \times c \\
n & : 1 \times 1 \quad \text{(optional)} \\
\text{byteorder} & : 1 \times 1 \quad \text{(optional)} \\
\text{result} & : 1 \times 1
\end{align*}

**Diagnostics**

None.

Note that \texttt{hash1}(x[, . . .]) never returns a missing result, even if \(x\) is or contains a missing value. In the missing case, the hash value is calculated of the missing value. Also note that \(x\) can be a vector or a matrix, in which case the result is calculated over the elements aligned rowwise as if they were a single element. Thus \texttt{hash1(\("a", "b")\)} == \texttt{hash1(\("ab")\)}.

**References**


——. unknown. A hash function for hash table lookup. \url{http://www.burtleburtle.net/bob/hash/doobs.html}.

**Also see**

[M-5] \texttt{asarray()} — Associative arrays

[M-4] \textbf{Programming} — Programming functions