hashes egg

Miscellaneous Hash Functions
Extension for Chicken Scheme
Version 1.9

Kon Lovett

Table of Contents

1 About this egg

1.1 Version history

`1.9` Rename of make-inexact -> make-real, bug fix tounsigned-int32

`1.8` Exports

`1.7` Fix for undeclared uint16_t [Thanks to Benedikt Rosenau]

`1.6` Rename of unsigned-long to unsigned-int32, needs Chicken 2.3+

`1.5` Minor implementation changes, needs box 1.1+

`1.4` Added dependency on box

`1.3` Added Fowler/Noll/Vo hash

`1.2` Update doc & renamed string->unsigned-long

`1.1` Added procs

`1.0` Initial release

1.2 Requirements

This egg requires the following extensions:

`message-digest`, `crc`, `box`, `misc-extn`, `miscmacros`

1.3 Usage

Load this egg like so:

```
(require-extension hashes)
```

2 Documentation

2.1 Hash Procedures

A suite of hash procedures. All have the same signature. All return hash values in a 32-bit range, so non-fixnum results possible! The `HASH` name refers to one of the hash algorithm symbols below.

Though this egg is categorized as 'cryptographic' not a one of these is suitable for cryptographic work!

The hash values produced are not necessarily portable across big/little-endian machines.

`HASH-prim` [procedure]

```
(HASH-prim STRING LENGTH SEED)
```

Returns the HASH-prim of `STRING` of `LENGTH`, using `SEED`.

`HASH` [procedure]

```
(HASH STRING [LENGTH [SEED]])
```

Returns the HASH-prim of `STRING`. When `LENGTH` is missing `(string-length STRING)` is assumed. When `SEED` is missing 0 is assumed.

`RJMXHash`

Bob Jenkins' MIX hash function.

`TWMXHash`

Thomas Wang's MIX hash function.

`FNVHash`

Fowler/Noll/Vo 1 hash function. Substitutes the algorithm's initial value when the seed is non-zero.

`FNVAHash`

Fowler/Noll/Vo 1a hash function. Substitutes the algorithm's initial value when the seed is non-zero.

`PHSFHash`

Paul Hsieh's SuperFast hash function. Substitutes the algorithm's initial value when the seed is non-zero.

`RSHash`

Robert Sedgwick's "Algorithms in C" hash function.

`JSHash`

A bitwise hash function written by Justin Sobel. Ignores the seed when 0.

`PJWHash`

Hash algorithm is based on work by Peter J. Weinberger of AT&T Bell Labs.

`ELFHash`

Similar to the PJW Hash function, but tweaked for 32-bit processors. It's the hash function widely used on most UNIX systems.

`BKDRHash`

This hash function comes from Brian Kernighan and Dennis Ritchie's book "The C Programming Language". It is a simple hash function using a strange set of possible seeds which all constitute a pattern of 31....31...31 etc, it seems to be very similar to the DJB hash function

`SDBMHash`

This is the algorithm of choice which is used in the open source SDBM project. The hash function seems to have a good over-all distribution for many different data sets. It seems to work well in situations where there is a high variance in the MSBs of the elements in a data set.

`DJBHash`

An algorithm produced by Professor Daniel J. Bernstein and shown first to the world on the usenet newsgroup comp.lang.c. It is one of the most efficient hash functions ever published. Substitutes the algorithm's initial value when the seed is non-zero.

`NDJBHash`

Now favored by Bernstein. Substitutes the algorithm's initial value when the seed is non-zero.

`DEKHash`

An algorithm proposed by Donald E. Knuth in "The Art Of Computer Programming, Volume 3", under the topic of sorting and search chapter 6.4. Substitutes the algorithm's initial value when the seed is non-zero.

`APHash`

Arash Partow's hash function. A hybrid rotative and additive hash function algorithm based around four primes 3, 5, 7, and 11.

`CRCHash`

The `crc32` procedure wrapped as above. Ignores the seed when 0.

2.2 Hash Auxillary Procedures

`current-hash-seed` [parameter]

```
(current-hash-seed [NEW-SEED])
```

Returns or sets the current default hash seed. The initial value is 0.

`make-seeded-hash` [procedure]

```
(make-seeded-hash HASH-PROC [SEED])
```

Returns a curried `HASH-PROC` of 1 or 2 arguments with the supplied `SEED`. When the seed is missing the `(current-hash-seed)` is assumed.

`make-range-hash` [procedure]

```
(make-range-hash HASH-PROC UPPER [LOWER])
```

Returns a `HASH-PROC` with the hash value restricted to the supplied exact interval, [`LOWER UPPER`]. When `LOWER` is missing 0 is if (data == NULL) return initval. The signature is that of the `HASH-PROC`.

`make-real-hash` [procedure]

```
(make-real-hash HASH-PROC)
```

Returns a `HASH-PROC` with the hash value restricted to the interval, [0.0 1.0]. The signature is that of the `HASH-PROC`.

`make-range-restriction` [procedure]

```
(make-range-restriction UPPER [LOWER])
```

Returns a procedure of 1 argument, a number. The arguments will be swapped if necessary so the range is [`LOWER` `UPPER`]. When `LOWER` missing 0 is assumed.

2.3 Digest Procedures

The acceptable input objects for the digest procedures are strings, input-ports, byte-vectors, or anything that can be converted to a byte-vector. See message-digest for more information.

The HASH name below refers to one of the hash algorithm symbols above.

`HASH:digest` [procedure]

```
(HASH:digest OBJECT)
```

Returns the HASH of `OBJECT` as a hex string.

`HASH:binary-digest` [procedure]

```
(HASH:binary-digest OBJECT)
```

Returns the HASH of `OBJECT` as a string.

`HASH:primitive` [procedure]

```
(HASH:primitive)
```

Returns the HASH primitive object.

2.4 Miscellaneous Procedures

`string-binary->unsigned-int32` [procedure]

```
(string-binary->unsigned-int32 STRING)
```

Returns the first 32-bits of `STRING` as an unsigned-int32, not guaranteed to be a fixnum.

`string-binary-unsigned-int32-set!` [procedure]

```
(string-binary-unsigned-int32-set! STRING UNSIGNED_INT32)
```

Sets the first 32-bits of `STRING` to `UNSIGNED_INT32`.

`UNSIGNED-INT32-SIZE` [constant]

Sizeof unsigned-int32 in bytes.

3 License

Copyright (c) 2006, Kon Lovett. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the Software), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED ASIS, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Does not supercede any restrictions found in the source code.

Index