
Subject: Re: IDL implementation of SHA1 checksum
Posted by [Haje Korth](#) on Sat, 14 Sep 2013 11:20:49 GMT
[View Forum Message](#) <> [Reply to Message](#)

Thanks for sharing!

On Thursday, September 12, 2013 3:23:35 PM UTC-4, John Correia wrote:

```
> All:
>
>
>
> I wanted to be able to calculate SHA1 hashes from within IDL (without
>
> resorting to SPAWN). Since I've never seen another IDL implementation I
>
> thought I'd share what I cooked up with the group. It works with either
>
> a file or a string as input.
>
>
>
> IDL> print, jc_sha1('The quick brown fox jumps over the lazy dog')
>
> 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
>
>
>
> IDL> print, jc_sha1('')
>
> da39a3ee5e6b4b0d3255bfef95601890afd80709
>
>
>
> IDL> print, jc_sha1('/path/to/an/empty.file')
>
> da39a3ee5e6b4b0d3255bfef95601890afd80709
>
>
>
> I have not tested this on Windows or Mac (or a big endian machine).
>
>
>
> Best regards,
>
>
>
```



```

>  isFile = file_test(input) or KEYWORD_SET(FILE)
>
>  if KEYWORD_SET(String) then isFile=0
>
>  if isFile then begin
>
>    isZeroLength = file_test(input,/ZERO_LENGTH)
>
>    canRead = file_test(input,/READ)
>
>    if ~(canRead) then begin
>
>      print, "Can't read this file"
>
>      return, -1
>
>    endif
>
>    bytearray = isZeroLength ? byte("") : read_binary(input)
>
>  endif else begin
>
>    bytearray = byte(input)
>
>  endelse
>
>
>
>  mlen = bytearray[0] eq 0b ? 0ULL : 8ULL*N_ELEMENTS(bytearray)
>
>  bytearray = bytearray[0] eq 0 ? 128b : [TEMPORARY(bytearray),128b]
>
>  while (8*N_ELEMENTS(bytearray) mod 512) ne 448 do $
>
>    bytearray = [TEMPORARY(bytearray),0b]
>
>    bytearray = [TEMPORARY(bytearray),reverse(byte(mlen,0,8))]
>
>  message = ulong(bytearray)
>
>
>
>  h0 = '67452301'xul
>
>  h1 = 'EFCDAB89'xul
>
>  h2 = '98BADCFE'xul
>
>

```

```

> h3 = '10325476'xul
>
> h4 = 'C3D2E1F0'xul
>
>
>
> w0 = ULONARR(80)
>
>
>
> for chind=0, n_elements(message)-1, 64 do begin
>
>
>
>     M = message[chind:chind+63]
>
>     w = w0
>
>     for i=0,15 do $
>
>         w[i] = TOTAL(M[i*4:i*4+3]*[16777216ul,65536ul,256ul,1ul],$
>
> /PRESERVE_TYPE)
>
>     temp = w
>
>     for i=16,79 do begin
>
>         temp = w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]
>
>         w[i] = (temp*2ul) OR (temp/2147483648ul)
>
>     endfor
>
>
>
>
>     a = h0
>
>     b = h1
>
>     c = h2
>
>     d = h3
>
>     e = h4
>
>
>
>

```

```

>   for i=0, 19 do begin
>
>       temp = ((a*32ul) OR (a/134217728ul)) + jc_SHAfunction1(b,c,d) $
>
>   + e + 1518500249ull + w[i]
>
>       e = d
>
>       d = c
>
>       c = (b*1073741824ul) OR (b/4ul)
>
>       b = a
>
>       a = ulong(temp)
>
>   endfor
>
>   for i=20, 39 do begin
>
>       temp = ((a*32ul) OR (a/134217728ul)) + jc_SHAfunction2(b,c,d) $
>
>   + e + 1859775393ull + w[i]
>
>       e = d
>
>       d = c
>
>       c = (b*1073741824ul) OR (b/4ul)
>
>       b = a
>
>       a = ulong(temp)
>
>   endfor
>
>   for i=40, 59 do begin
>
>       temp = ((a*32ul) OR (a/134217728ul)) + jc_SHAfunction3(b,c,d) $
>
>   + e + 2400959708ull + w[i]
>
>       e = d
>
>       d = c
>
>       c = (b*1073741824ul) OR (b/4)
>

```

```

>     b = a
>
>     a = ulong(temp)
>
>   endfor
>
>   for i=60, 79 do begin
>
>     temp = ((a*32ul) OR (a/134217728ul)) + jc_SHAfunction2(b,c,d) $
>
> + e + 3395469782ull + w[i]
>
>     e = d
>
>     d = c
>
>     c = (b*1073741824ul) OR (b/4ul)
>
>     b = a
>
>     a = ulong(temp)
>
>   endfor
>
>
>
>   h0 += a
>
>   h1 += b
>
>   h2 += c
>
>   h3 += d
>
>   h4 += e
>
>
>
>   endfor
>
>
>
>   h0 = string(h0,format='(z08)')
>
>   h1 = string(h1,format='(z08)')
>
>   h2 = string(h2,format='(z08)')
>
>

```

```
> h3 = string(h3,format='(z08)')
>
> h4 = string(h4,format='(z08)')
>
>
>
> return, h0+h1+h2+h3+h4
>
> end
```
