Subject: Re: Compiling file with many functions: huge performance difference between IDL and IDLDE
Posted by mwvogel on Fri, 19 Mar 2004 11:30:54 GMT
View Forum Message <> Reply to Message

> You can find a hash_table implementation on the RSI user contribution
> site. Quick performance test for 12000 sets/gets:
> #hashes set/get per entry (ms)
> 13 6
> 101 0.8
> 1001 0.15
> 12000 0.08/0.05
>
> Talking about it:
> How would you calculate a hash value from a string? In C I would
> base it on the ASCII value of the chars, but in IDL? Above mentioned
> implementation converts the string via byte() and then loops over
> the resulting array. Is there a faster way (loops always take so long)?

For real short strings (up to approx 11 chars) one could replace

```
        ascii = ulong(byte(key))
        total = 0UL
        for i = 0, n_elements(ascii) - 1 do begin
           total = total * 37UL + ascii[i]
              ; 37UL is a magic number suggested by the literature
        endfor
        return, total
with
        ascii = ULONG(BYTE(key))
        total = ULONG(TOTAL(ascii *
ULONG(37D^(N_ELEMENTS(ascii)-FINDGEN(N_ELEMENTS(ascii)) -1)), /DOUBLE))
        return, total
```

However, TOTAL() produces a double, and is therefore prone to roundoff errors, possibly reducing the hashing efficiency. Also, I am not so sure that the for loop is that much slower for small arrays. At least the original code is easier to read :-)