Subject: Re: Optimized memcmp for x86
Posted by mirek on Sat, 23 Feb 2008 15:15:21 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Fri, 22 February 2008 13:08luzr wrote on Fri, 22 February 2008 11:07

```
 if(len & 2)
  if(*(word *)x != *(word *)y)
   return int(_byteswap_ushort(*(word *)x) - _byteswap_ushort(*(word *)y));
 if(len & 1)
  return int(*((byte *)x + 2)) - int(*((byte *)y + 2));
```

I don't get this end.

switch (len & 3)
0: it looks ok to me.
1: the return int(*((byte *)x)) - int(*((byte *)y)); should be returned?
2: looks ok
3: looks ok

I would maybe try masking out unused bytes, but that would lead to read out of buffer boundary. Is it safe?
I mean something like this

```
   ...
   const static dword masks[4] = { 0x00000000, 0x000000FF, 0x0000FFFF, 0x00FFFFFF };
//Intel-like endian only!
   return int(_byteswap_ulong(*x & masks[len&3]) - _byteswap_ulong(*y & masks[len&3]));
```

I'm not sure I got the byteswap purpose correctly, but I think I got, so my code is probably ok (but I didn't test it).

Of course it reads beyond buffer end, so you need to know it will not raise exception or crash the application on target platform.

Yes, you are right, len&1 was wrong... (now is not it nice to have the code checked by posting here?

And the idea of mask is interesting, but poses interesting problem as well:

We are not guaranteed by "len" parameter that we can read the whole dword. At the same time, "external" (like memory model and allocator) seem to guarantee that (because we have started on aligned...). I will have to think hard about this

Mirek