

Re: page sizes

Source: <http://coding.derkeiler.com/Archive/General/comp.programming/2008-04/msg00249.html>

- *From:* moi <root@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sun, 13 Apr 2008 01:55:10 +0200
-

On Sat, 12 Apr 2008 23:01:23 +0200, copx wrote:

"Philip" <phlip2005@xxxxxxxx> schrieb im Newsbeitrag
[news:4800038f\\$0\\$25921\\$4c368faf@xxxxxxxxxxxxxxxxxxxxxx](mailto:news:4800038f$0$25921$4c368faf@xxxxxxxxxxxxxxxxxxxxxx)

copx wrote:

Do memory managers usually use page sizes which are a power of two or not?

Yes. This optimizes the variables and hardware registers used to manage the pages. None of their bits are wasted. User-level code doesn't need these optimizations, so they pick array sizes with different conveniences.

Usually 4096 bytes.

X86 has an optional bigger pagesize (1MB or 4MB, can't remember), which is hardly ever used. NB: DEC-alpha has 8K pages which makes it a great platform for testing portability.

I am trying to squeeze a few more nanoseconds out of a programming language interpreter. The documentation of `malloc()` on one of the platforms I am targetting explicitly gives the advice to allocate powers of 2 to improve performance. However, that is only true if the allocated blocks are \leq pagesize. Larger blocks allocate fastest if they are multiples of the page size.

Re: page sizes

For performance, think in terms of cache lines (16 bytes on x86) or pages (4K on x86) Spanning a cache line or page boundary results in (approx) twice the number of cache misses/ page faults. Malloc *may* avoid this overhead by always putting chunks on cacheline boundaries (and wasting some space).

Also; IMHO you should not rely on malloc's implementation. Some mallocs may return consecutive ranges, others may intersperse the chunks with their own overhead. Pre-allocating big (page aligned) chunks is always better, iff you can handle the overhead.

HTH,
AvK

.