

# OT/drift: when is a RAMdisk an appropriate solution

---

*Source:* [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2006-08/msg05063.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2006-08/msg05063.html)

---

- *From:* Chris Torek <nospam@xxxxxxxx>
  - *Date:* 29 Aug 2006 07:39:06 GMT
- 

Flash Gordon wrote:

[RAM-disk creation and manipulation is] no more difficult and no less system specific than it was back then.

Well, no less system-specific, to be sure. Sometimes "more (or less) difficult" depending on the system. (Aside: in vxWorks, just include the "ram disk" component in your project. The default name is "/ram0" but you can change it. If you want more than one, it is somewhat more difficult.)

The difference is that if you are doing it to speed things up you are quite possibly wasting your time and effort.

Indeed.

In article <H5GIg.4036\$y61.1915@fed1read05>  
jmccgill <jmccgill@xxxxxxxxxxxxxxxxxxxx> wrote:

It still makes sense on diskless clients that boot over NFS and so on. It's also indicated in some applications where security is sensitive, or for devices that need to operate in environments where moving parts are impractical.

Sometimes, a physical (as opposed to virtual, in-kernel-only) RAM disk is appropriate. That is, you have a piece of hardware you plug in, which looks and acts like a disk as far as the system is concerned, but actually stores data in RAM.

While many systems (including vxWorks, as mentioned above) do include in-main-memory pseudo-disk devices, in a purely theoretical sense, there is *\*never\** [%] any reason to use one: whatever you are doing that is going through some sort of "file system" layer, only to wind up going directly to regular old ordinary memory, is

## OT/drift: when is a RAMdisk an appropriate solution

going through a "unnecessary" manipulation. File systems do a lot of work in order to deal with structured, and somewhat klunky, disk drive hardware, where memory exists in the form of "disk blocks" or "sectors" that can only be rewritten in entire units, cannot be moved or resized, and is generally very slow to access (about 5 or 6 decimal orders of magnitude slower than RAM). Real RAM does not share most of these characteristic drawbacks, and going through a (usually quite heavy) software layer that simply pretends to add some of them back is just silly.

[% One obvious exception is when the RAM-disk is being used for testing the file system software. Here, even in theory, the RAM drive is useful. :-) ]

In practice, the reason for using a RAM disk is that the stubborn software (written by some stubborn and/or long-gone programmer) insists on using the file system interface, and the system provides no way to short-circuit this. For instance, in C -- to get at least marginally back on topic -- one might have a program (or large library) that does all its I/O to a "stdio" stream. Since ISO C has no way to "open memory as a stream", one is often forced to supply an actual, on-disk file. To get decent performance, one may wish to use an "on-RAM-disk file" instead of "on-actual-disk".

Note that it can make sense to use a "RAM file system" rather than a "RAM disk" in this case. Going through a \*real\* file system, you may do a whole lot of extra work trying to avoid talking too much to a slow (real) disk -- i.e., spend large amounts of CPU time to avoid waiting for the device. But if the "device" is fake, and actually is very fast (relative to real disks), this CPU time is being wasted. It may be better to go through a fake file system (one which uses RAM), instead of a real file system on a fake disk.

Of course, it would be even better still if C had some sort of memory-oriented stdio streams -- or something like the 4.4BSD "funopen", which is even more general. (I am pretty sure that funopen() did get proposed for C99; but we might consider ourselves lucky to have gotten even snprintf(). :-) )

--

In-Real-Life: Chris Torek, Wind River Systems  
Salt Lake City, UT, USA (40°39.22'N, 111°50.29'W) +1 801 277 2603  
email: forget about it <http://web.torek.net/torek/index.html>  
Reading email is like searching for food in the garbage, thanks to spammers.

.