

Re: Programming in standard c

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2007-12/msg04178.html

- *From:* Eric Sosman <esosman@xxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 28 Dec 2007 12:04:57 -0500
-

Bart C wrote:

[...]

I don't know how to deal with /var/adm/messages or similar. Suppose I read byte-by-byte as recommended then someone updates the beginning of the file? Maybe it's foolhardy to even attempt making a copy of such a file. I'm not allowed to lock it because that's also frowned upon. What exactly can one do with such a file?

There are at least two problems to be faced when dealing with /var/adm/messages. First, the file grows as the system appends new log messages to it, an activity not coordinated with the actions of J. Random Program. If JRP queries the file size, then allocates a buffer of that size, and then tries to fill the buffer by reading blindly to EOF, the file may grow larger between the query and the reading. Thus, the queried size must be considered an estimate, not an absolute: use Chris Torek's approach, not Jacob Navia's.

The second problem is that the system does not allow the file to grow without limit. Every now and then, it renames the existing /var/adm/messages and creates a new, empty one where future messages are deposited. So the semantics of how the system identifies "a" file become important: Are the query and the reading directed to the same bunch of bytes, or only to the same file name? This is the (or a) reason POSIX systems have both the stat() and fstat() query operations: the first asks about the file associated with a given name, while the second asks about a file that's already open and whose name is no longer relevant.

So how does one process /var/adm/messages, or more broadly, how does one process a file that may be subject to change while the processing is in progress? The C language has almost no support for parallel activities; signals and volatile are about the extent of it, and neither is useful wit