

Re: aio_read/write versus O_NONBLOCK

Source: <http://unix.derkeiler.com/Newsgroups/comp.unix.programmer/2008-05/msg00244.html>

- *From:* scott@xxxxxxxxxxxxxx (Scott Lurndal)
 - *Date:* Tue, 20 May 2008 23:49:36 GMT
-

phil-news-nospam@xxxxxxxx writes:

On Thu, 15 May 2008 21:56:27 GMT Scott Lurndal <scott@xxxxxxxxxxxxxx> wrote:
| phil-news-nospam@xxxxxxxx writes:
> On Thu, 15 May 2008 18:55:43 GMT Scott Lurndal <scott@xxxxxxxxxxxxxx> wrote:
> | phil-news-nospam@xxxxxxxx writes:
> |> On Wed, 14 May 2008 00:13:59 -0400 Barry Margolin <barmar@xxxxxxxxxxxxxx>
> |> wrote:
> |> | In article
> |> | <fe463120-697d-4100-9ca8-b74083491664@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>,
> |> | RazvanD <razvand@xxxxxxxx> wrote:
> |> |>
> |> |> Hi!
> |> |>
> |> |> Could someone point me to some articles or give me some hints on the
> |> |> advantages/disadvantages of asynchronous operations on files (aio_read/
> |> |> aio_read) versus normal operations (read/write) used with O_NONBLOCK
> |> |> when opening a file.
> |> |>
> |> |> aio_read/aio_write are indeed more flexible. But, at least on Linux/
> |> |> glibc, they are implemented using POSIX threads. Wouldn't a carefully
> |> |> designed program using O_NONBLOCK for files best a program using
> |> |> aio_read/aio_write? I think my question should be: are asynchronous I/
> |> |> O operations only more flexible or are they also faster?
> |> |>
> |> |> I'm not certain about Linux, but on many systems O_NONBLOCK has no
> |> |> effect on ordinary file streams.
> |> |>
> |> |> It has not had any effect on ordinary file streams in Linux in the programs
> |> |> I have written that tried it (a couple of them).
> |> |>
> |> |>
> |> |> It's pretty much of a waste of time. O_NONBLOCK doesn't make any sense
> |> |> for file descriptors that can't block on a read (i.e. disk-based files).
> |> |>
> |> |> Could you explain why you believe that to be the case?
> |> |>
> |> |>
|> |> |> It's simple. The disk is always there, so a read must always complete

Re: aio_read/write versus O_NONBLOCK

| in a bounded time. O_NONBLOCK was added to Unix to handle cases where
| a blocking read can be unbounded (serial ports, parallel ports, network
| ports).

But a bounded time is not zero time. There are things that can be done
while a disk is being read.

I don't think you really understand how file-based I/O works in linux.
In the majority of cases, by the time you issue the read(2)/pread(2) system
call, it is likely that the data has already been read from the disk.

In any case, what you want is asynchronous I/O, rather than non-blocking I/O.

See aio_read/aio_write/lio_listio.

Btw, if you're copying one file to another, use mmap/madvise
not read/pread/write/pwrite.

scott

.