

Re: When does `write` return?

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.misc/2006-01/msg00982.html>

- *From:* "Andrew Xue" <xueruini@xxxxxxxxxx>
 - *Date:* 13 Jan 2006 16:56:42 -0800
-

Jean-David Beyer wrote:

> Andrew Xue wrote:

>> Jean-David Beyer ^{TMS}

>>

>>

>>> It depends on a coupla things. If you have much more than 1GBytes of

>>> ram, the write could return as soon as the data get to a memory buffer.

>>> Possibly some of the cache would have to be paged out, but not

>>> necessarily.

>>>

>>

>> My ram is a little bit large, 4GB. And my biggest test file is 1.8GB, so

>> I think it's cached entirely.

>>

>>

>>> It also depends if file fd is a raw file or not (if you are using raw

>>> file systems, you probably already know enough to answer your

>>> questions; so probably not a raw file). If it is, I believe you must

>>> wait until the write completes, but OTOH, you do not need to take the

>>> time to find system buffer space and copy from user space to system

>>> space, and you might be able to do it in a giant "spiral write" as DEC

>>> used to call it. Depends on hardware and the low level driver, I would

>>> expect.

>>

>> Could you please explain it in more details? I could not find anything

>> about "spiral write" by google. Does it mean something of copy-on-write?

>>

> Sorry. That was a DEC thing on their hard drives in the very late 1960s, at

> least on the PDP-11 and VAX-11 series machines. It has nothing to do with

> copy-on-write.

>

> Their disks had 512-byte sectors and UNIX at the time normally read in one

> sector at a time, and if it detected sequential reading, it might read the

> next block as well.

>

> But the hardware controller for the disk drives allowed you to read any

> multiple of 512 bytes you wanted. I do not remember the numbers, so I will

> make them up here, but you will get the idea. Imagine there are 19 tracks

Re: When does `write' return?

> per cylinder and 20 sectors per track. If you started at block zero and read
> 512 * 20 * 19 bytes, you would read the whole cylinder.
> But if you read 512 * 20 * 29 bytes, the controller would accept that, read
> the whole cylinder and do an automatic seek with no intervention from the OS
> and read about half the next cylinder as well. You could go as much as you
> wanted, up to almost 65536 bytes (which would use up all the data space in
> any one process).

Thank you so much for the useful information. However, I am still
confused whether
the kernel write some data to the disk while it's buffering them? If
so, how much is
written?

>

>

> --

> .~. Jean-David Beyer Registered Linux User 85642.
> /V\ PGP-Key: 9A2FC99A Registered Machine 241939.
> /(\) Shrewsbury, New Jersey <http://counter.li.org>
> ^^--^^ 22:35:00 up 14 days, 13:21, 3 users, load average: 4.19, 4.28, 4.21

• **References:**

- ◆ **When does `write' return?**
 ◇ From: Andrew Xue
- ◆ **Re: When does `write' return?**
 ◇ From: Jean-David Beyer
- ◆ **Re: When does `write' return?**
 ◇ From: Andrew Xue
- ◆ **Re: When does `write' return?**
 ◇ From: Jean-David Beyer

- Prev by Date: **[A new reader? Welcome to comp.os.linux.misc, read this first if you're new here \(FAQ\)](#)**
- Next by Date: **[Re: uninterruptible sleep](#)**
- Previous by thread: **[Re: When does `write' return?](#)**
- Next by thread: **[sar: difference paging and swapping – paging without swap space](#)**
- Index(es):
 - ◆ **[Date](#)**
 - ◆ **[Thread](#)**