

# How to use "offset" parameter in mmap() system call

**Source:** <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/hackers/2005-09/0379.html>

---

**From:** Apache Xie ([apachexm\\_at\\_hotmail.com](mailto:apachexm_at_hotmail.com))

**Date:** 09/29/05

To: [freebsd-hackers@freebsd.org](mailto:freebsd-hackers@freebsd.org)

Date: Thu, 29 Sep 2005 04:43:59 +0000

Hi,

I wrote a device driver, it supports mmap(), but I met some problems recently, it seems relate to the "offset" parameter in mmap() system call, what I want to get help are:

How to use the offset parameter of mmap() correctly?

Can it be used freely?

My driver is a char device driver, when it is loaded, it allocates 64 MB contiguous physical memory using contigmalloc(), and I wrote my own memory management function to allocate memory blocks from this 64 MB memory. My driver can be used by several processes concurrently, each process allocates one block from the 64 MB memory, then mmap this block to user space.

Each process is assigned a sequence number when it opens the device. For example, assume process X gets the sequence number 0, and allocates memory block 0-2MB, and process Y gets the sequence number 1, and allocates memory block 4-6MB, we use offset 0, size 2MB in process X on calling mmap() system call, and use offset 64MB, size 2MB in process Y on calling mmap() system call. In driver's \_mmap() function, we check offset, when it is < 64MB, we know process X do the mmap, then we return physical address of memory block 0-2MB, when the offset is between 64MB and 128MB, we know process Y do the mmap, then we return physical address of memory block 4-6MB. In other word, the "offset" parameter in mmap() system call is just a indicator, we use it to compute the real offset in the 64MB memory to map the memory block the process allocated. My driver can support maximum 16 processes, so the "offset" parameter in mmap() system call can be set to near  $16 * 64MB = 1GB$ .

I wrote a test program, it does very simple thing:

1. open char device;
2. mmap driver's memory;
3. write to mmapped memory;

#### 4. unmap driver's memory.

It works fine when I run it manually many times, but after I put it in a while loop in a shell script, and run several scripts concurrently, I found sometimes the mapped user space address don't point to the correct memory block the process allocated in driver. We checked the physical address returned by my driver's `_mmap()` function, it is correct, but when I sent the mapped user space address back to kernel and use `pmap_extract()` to get the physical address, it is wrong, it may be another block in the 64MB memory. For example, process X allocate 0-2MB, the mapped user space address is pX, process Y allocate 4-6MB, the mapped user space address is pY, but when pX and pY is sent back to kernel, `pmap_extract(pX)` is 4-6MB, `pmap_extract(pY)` is 0-2MB, this is very strange.

Below is the simple shell script which shows the error:

```
#!/bin/sh
while true
do
    ./my_test_program
done
```

I checked some driver codes in the kernel which support `mmap()`, it seems in many driver, "offset" is used as indicator too, they use "offset" to find the correct memory block and return it's physical address too.

But in my driver, the "offset" parameter the user process used in the `mmap()` system call can be very large, even the mapped memory block is always in the 64 MB contiguous physical memory. But we can compute the real offset correctly.

I am not familiar with FreeBSD kernel, but I find "offset" is used by kernel too, so, I am not sure if offset can be set to the very large value as my driver do.

Any help is welcomed, thanks!

---

Don't just search. Find. Check out the new MSN Search!  
<http://search.msn.click-url.com/go/onm00200636ave/direct/01/>

---

freebsd-hackers@freebsd.org mailing list  
<http://lists.freebsd.org/mailman/listinfo/freebsd-hackers>  
To unsubscribe, send any mail to "freebsd-hackers-unsubscribe@freebsd.org"