

# Re: sleeping and waiting and tasklets

---

*Source:* <http://linux.derkeiler.com/Newsgroups/comp.os.linux.development.system/2006-01/msg00059.html>

---

- *From:* [bill.at.abcdefghijklmnopqrstuvwxyz](mailto:bill.at.abcdefghijklmnopqrstuvwxyz)
  - *Date:* Fri, 6 Jan 2006 03:48:33 GMT
- 

Peter,

I rely on Linux Device Drivers, 3rd Edition, an O'Reilly book a lot. This is what they have to say about copy\_to\_user():

```
=====
unsigned long copy_to_user(void __user *to,
const void *from,
unsigned long count);
unsigned long copy_from_user(void *to,
const void __user *from,
unsigned long count);
```

Although these functions behave like normal memcpy functions, a little extra care must be used when accessing user space from kernel code. The user pages being addressed might not be currently present in memory, and the virtual memory subsystem can put the process to sleep while the page is being transferred into place. This happens, for example, when the page must be retrieved from swap space. The net result for the driver writer is that any function that accesses user space must be reentrant, must be able to execute concurrently with other driver functions, and, in particular, must be in a position where it can legally sleep. We return to this subject in Chapter 5. The role of the two functions is not limited to copying data to and from user-space: they also check whether the user space pointer is valid. If the pointer is invalid, no copy is performed; if an invalid address is encountered during the copy, on the other hand, only part of the data is copied. In both cases, the return value is the amount of memory still to be copied.

```
=====
```

This edition of the book has been updated to include the 2.6 kernel. It looks like it agrees with you in some places and disagrees in others. As far as verifying the user-space pointers, they say there is validation but you both agree on what happens when there is a problem. You were much more detailed in your explanation. As a result of the answers here, I have a much better understanding of the "do not" issues. Thanks to all of you!!  
Bill

---

