read-user, read-supervisor, write-user, and write-supervisor. Memory "gets" use the read indices and "sets" use the write indices. Thus, a switch between user mode and supervisor mode just requires changing two references in the memory system, as shown in Figure 9-5.

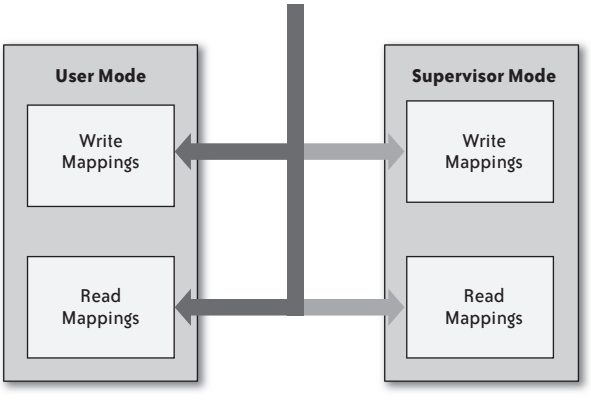

FIGURE 9-5. Protection level switching in the linear address space

NOTE

This fits in nicely with the Linux kernel's approach to paging. In Linux, every user-mode process has the kernel pages mapped in its linear address space. In hardware this prevents a context switch on transferring to kernel code for routine maintenance operations or system calls. Our switches to kernel space are just the flipping of array references, which is also a low-cost process.

This still leaves us with the problem of page faults and protection violations. Our attitude toward dealing with these conditions is, in some ways, an indication of an acceptance of the true nature of exceptions, both in the low-level processor sense and in the Java language. This attitude can be summed up as follows:

> An `Exception` is an exception, and an `Error` is an error.

More verbosely: an `Exception` should represent rare or exceptional conditions, but not necessarily fatal ones; an `Error` should represent fatal or certainly seriously undesirable situations.

Considering the extent to which exception handling is implemented in the Java language, it is in some ways curious that many programmers are reluctant to use the exception-handling mechanism to their advantage. Exceptions represent the most efficient and cleanest way for software to handle rare but correctable conditions. Throwing an exception will bias the execution cost by effectively optimizing for the common path, and concurrently will increase the cost involved in the inevitable exceptional situation. It is obvious from this discussion that