```
  b->flags |= B_VALID;
  b->flags &= ~B_DIRTY;
  wakeup(b);                          // wake waiting process
  if ((ide_queue = b->qnext) != 0) // start next request
    ide_start_request(ide_queue);  // (if one exists)
  release(&ide_lock);
}
```

图 36.5 xv6 的 IDE 硬盘驱动程序（简化的）

该协议的大部分可以在 xv6 的 IDE 驱动程序中看到，它（在初始化后）通过 4 个主要函数来实现。第一个是 ide_rw()，它会将一个请求加入队列（如果前面还有请求未处理完成），或者直接将请求发送到磁盘（通过 ide_start_request()）。不论哪种情况，调用进程进入睡眠状态，等待请求处理完成。第二个是 ide_start_request()，它会将请求发送到磁盘（在写请求时，可能是发送数据）。此时 x86 的 in 或 out 指令会被调用，以读取或写入设备寄存器。在发起请求之前，开始请求函数会使用第三个函数 ide_wait_ready()，来确保驱动处于就绪状态。最后，当发生中断时，ide_intr()会被调用。它会从设备中读取数据（如果是读请求），并且在结束后唤醒等待的进程，如果此时在队列中还有别的未处理的请求，则调用 ide_start_request()接着处理下一个 I/O 请求。

36.9 历史记录

在结束之前，我们简述一下这些基本思想的由来。如果你想了解更多内容，可以阅读 Smotherman 的出色总结[S08]。

中断的思想很古老，存在于最早的机器之中。例如，20 世纪 50 年代的 UNIVAC 上就有某种形式的中断向量，虽然无法确定具体是哪一年出现的[S08]。遗憾的是，即使现在还是计算机诞生的初期，我们就开始丢失了起缘的历史记录。

关于什么机器第一个使用 DMA 技术也有争论。Knuth 和一些人认为是 DYSEAC（一种“移动”计算机，当时意味着可以用拖车运输它），而另外一些人则认为是 IBM SAGE[S08]。无论如何，在 20 世纪 50 年代中期，就有系统的 I/O 设备可以直接和内存交互，并在完成后中断 CPU。

这段历史比较难追溯，因为相关发明都与真实的、有时不太出名的机器联系在一起。例如，有些人认为 Lincoln Labs TX-2 是第一个拥有向量中断的机器[S08]，但这无法确定。

因为这些技术思想相对明显（在等待缓慢的 I/O 操作时让 CPU 去做其他事情，这种想法不需要爱因斯坦式的飞跃），也许我们关注“谁第一”是误入歧途。肯定明确的是：在人们构建早期的机器系统时，I/O 支持是必需的。中断、DMA 及相关思想都是在快速 CPU 和慢速设备之间权衡的结果。如果你处于那个时代，可能也会有同样的想法。

36.10 小结

至此你应该对操作系统如何与设备交互有了非常基本的理解。本章介绍了两种技术，