知识点1【IO模型的分类】（了解）

阻塞式I/O、非阻塞式I/O、I/O复用、信号驱动式I/O、异步I/O

知识点2【IO模型的特点】（了解）

1、阻塞式I/O模型

默认 情形 下， 所有 套 接 字 都是 阻塞 的。

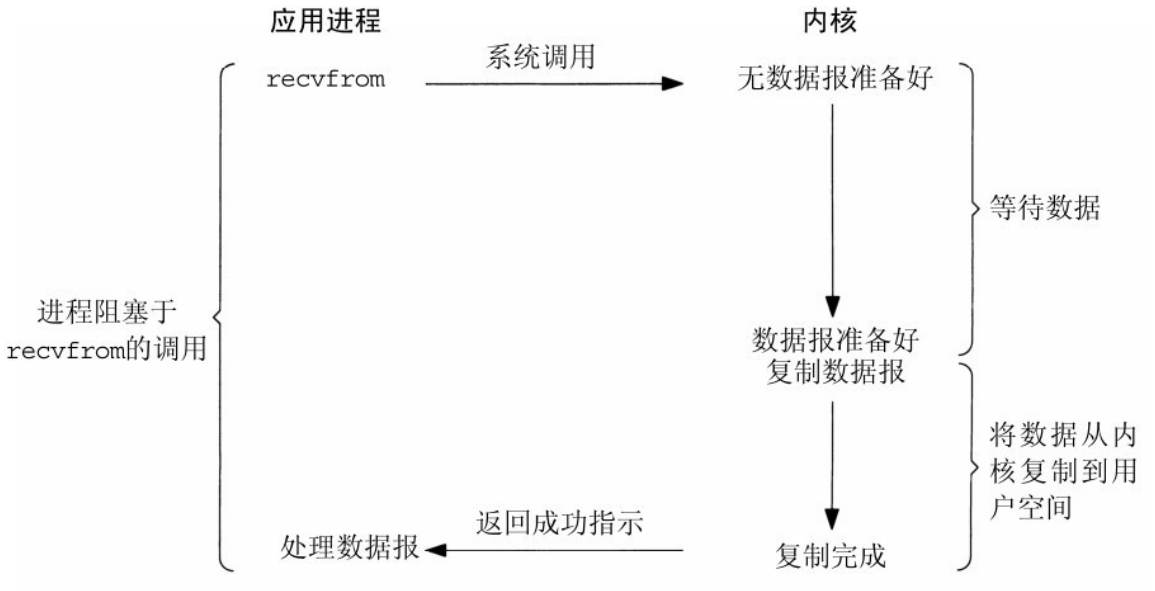


图6-1 阻塞式I/O模型

2、非阻塞I/O模型

进程将套接字设置成非阻塞 就是告诉内核：用户操作套接字，如果套接字没有准备好，就不会阻塞，而是返回错误信息 继续执行

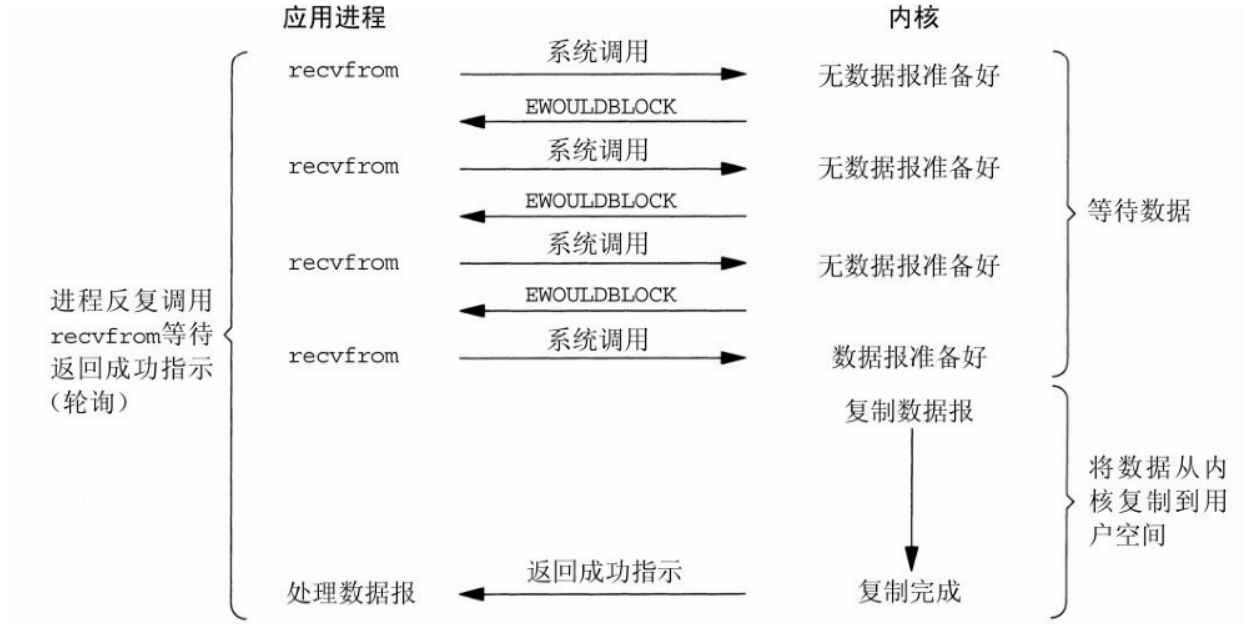
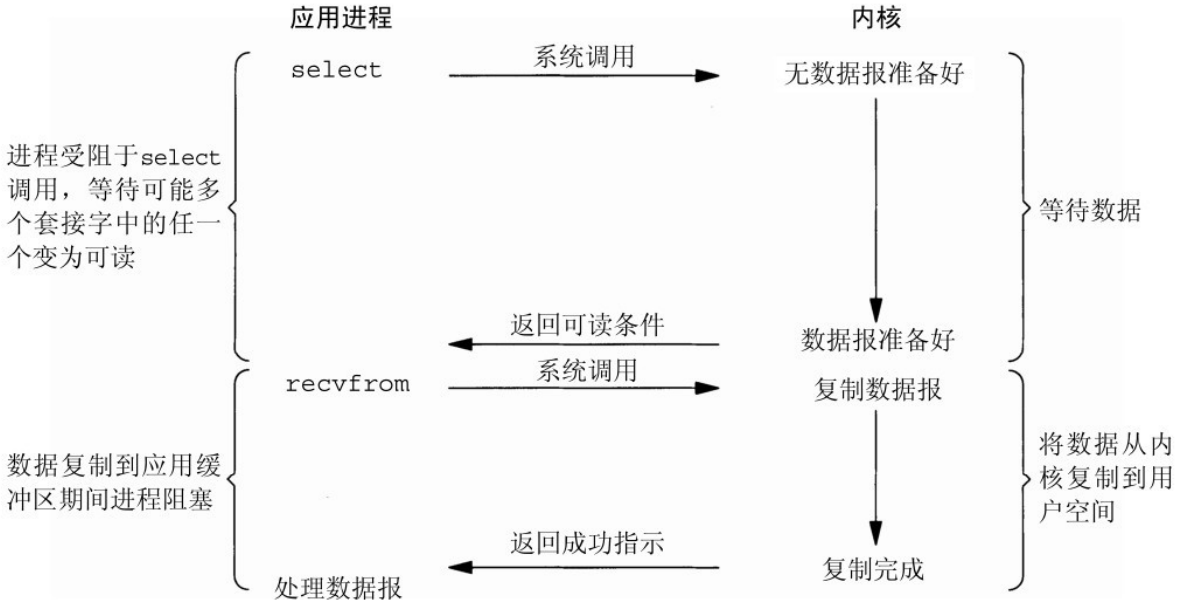


图6-2 非阻塞式I/O模型

3、I/O复用模型

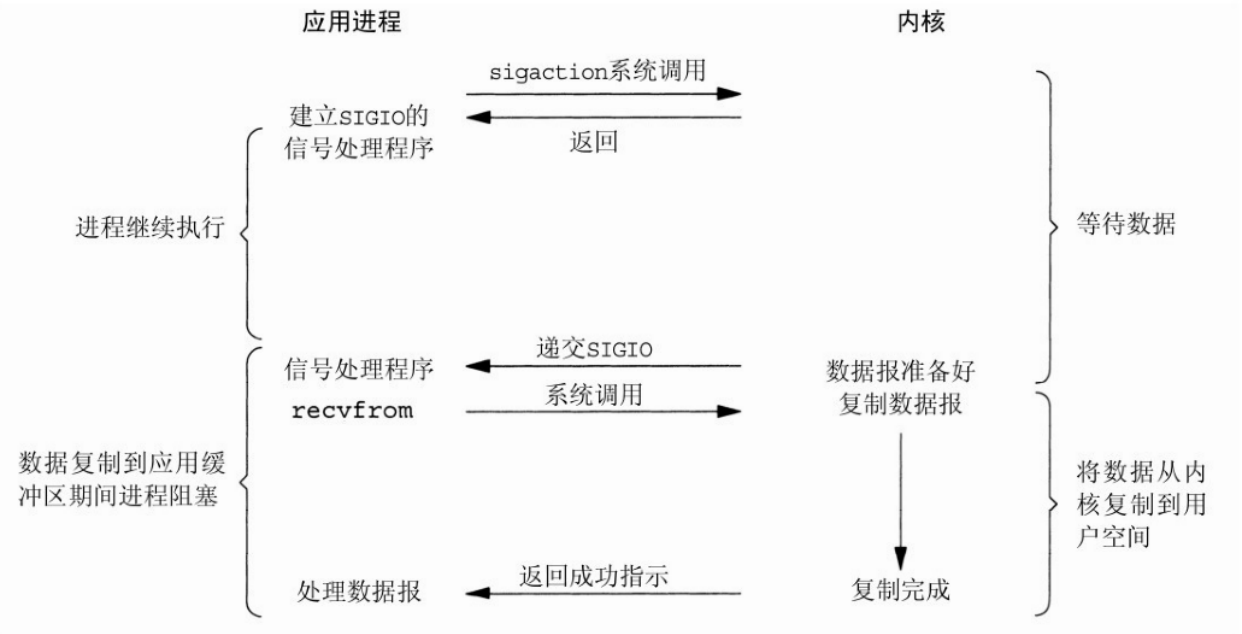
有了 I/ O 复 用（ I/ O multiplexing）， 我们 就可以 调用 select 或 poll， 阻塞 在这 两个 系统 调用 中的 某一个 之上， 而 不是 阻塞 在 真正 的 I/ O 系统 调用 上。



我们 阻塞 于 select 调用， 等待 数 据报 套 接 字 变为 可读。 当 select 返回 套 接 字 可读 这一 条件 时， 我们 调用 recvfrom 把 所 读数 据报 复制 到 应用 进程 缓冲区。

4、信号 驱动 式 I/ O 模型

我们 也可 以用 信号， 让 内核 在 描述 符 就绪 时 发送 SIGIO 信号 通知 我们。

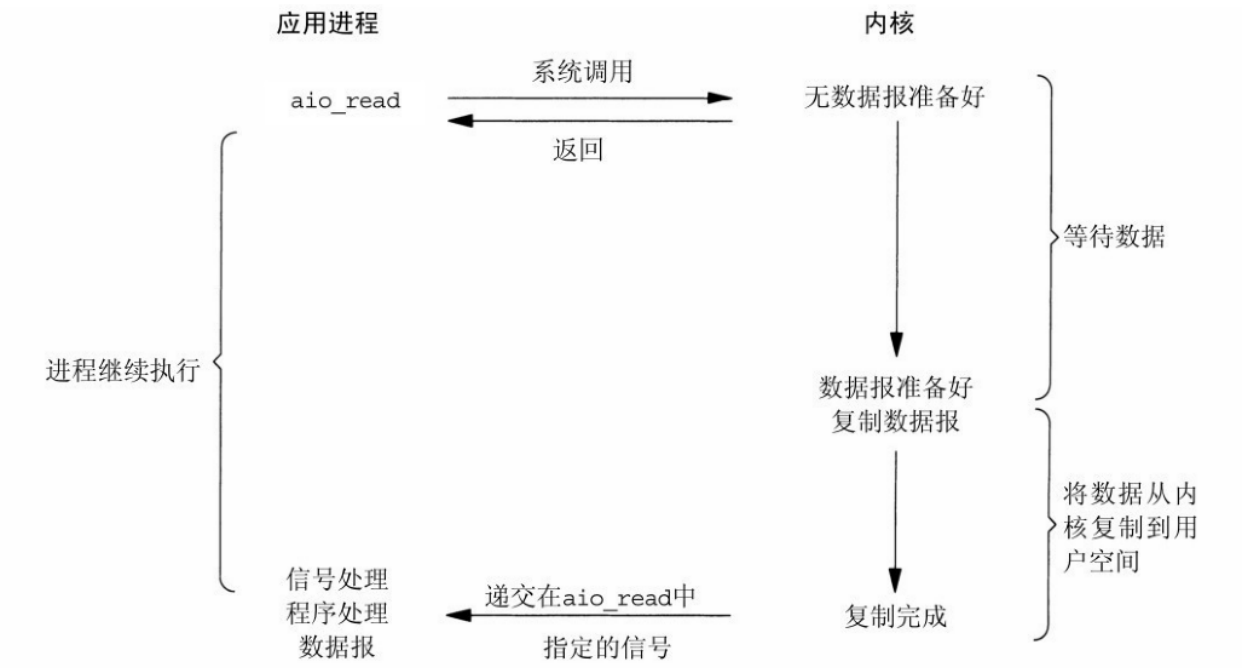


我们 首先 开启 套 接 字 的 信号 驱动 式 I/ O 功能， 并 通过 sigaction 系统 调用 安装 一个 信号 处理 函数。 该 系统 调用 将 立即 返回， 我们 的 进程 继续 工作， 也就是说 它 没有 被 阻塞。 当 数 据报 准备好 读取 时， 内核 就为 该 进程 产生 一个 SIGIO 信号。 我们

随后 既可 以在 信号 处理 函数 中 调用 recvfrom 读取 数 据报， 并 通知 主 循环 数据 已 准备好 待 处理， 也可以 立即 通知 主 循环， 让 它 读取 数 据报。

5、异步 I/ O 模型

工作 机制 是： 告知 内核 启动 某个 操作， 并 让 内核 在 整个 操作（ 包括 将 数据 从内 核 复制 到 我们 自己的 缓冲区） 完成 后 通知 我们。 这种 模型 与 前一 节 介绍 的 信号 驱动 模型 的 主要 区别 在于： 信号 驱动 式 I/ O 是由 内核 通知 我们 何时 可以 启动 一个 I/ O 操作， 而 异步 I/ O 模型 是由 内核 通知 我们 I/ O 操作 何时 完成。



我们 调用 aio_ read 函数（ POSIX 异步 I/ O 函数 以 aio_ 或 lio_ 开头）， 给 内核 传递 描述 符、 缓冲区 指针、 缓冲区 大小（ 与 read 相同 的 三个 参数） 和 文件 偏移（ 与 lseek 类似）， 并 告诉 内核 当 整个 操作 完成时 如何 通知 我们。 该 系统 调用 立即 返回， 而且 在 等待 I/ O 完成 期间， 我们 的 进程 不被 阻塞。 本 例子 中 我们 假设 要求 内核 在 操作 完成时 产生 某个 信号。 该 信号 直到 数据 已 复制 到 应用 进程 缓冲区 才 产生， 这一点 不同于 信号 驱动 式 I/ O 模型。