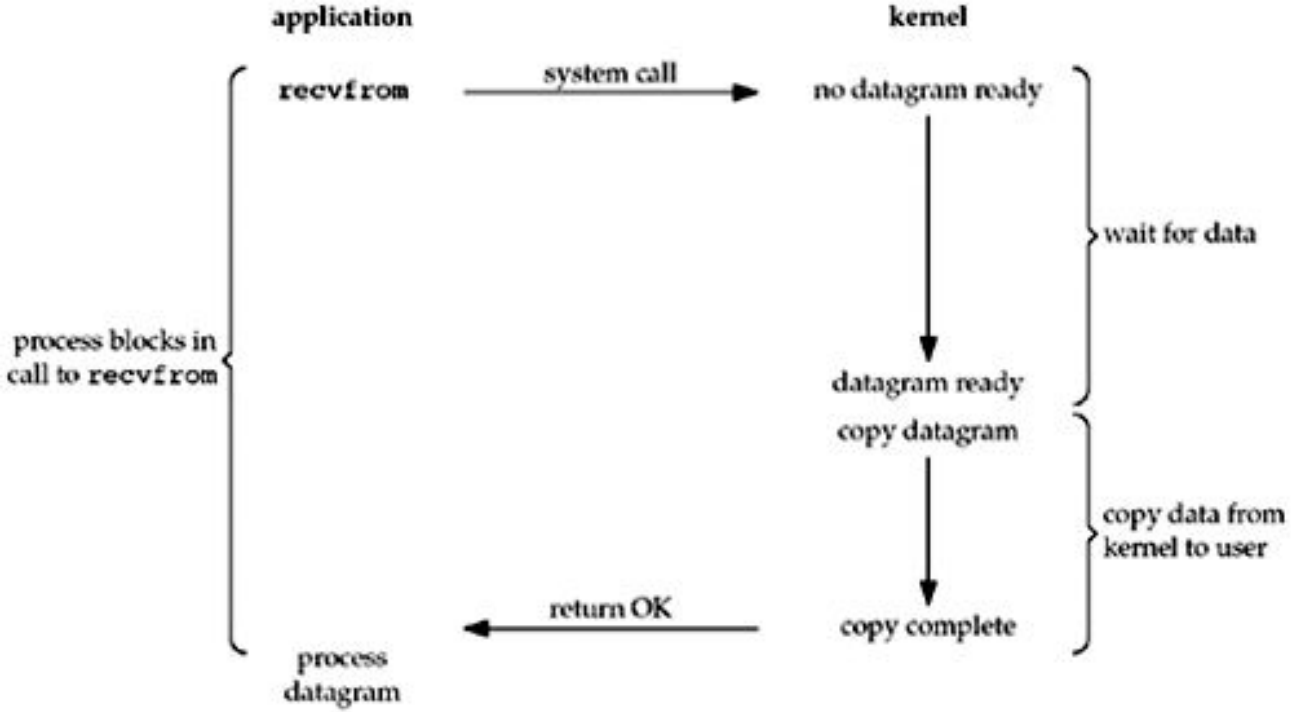
Многопоточное программирование на C++

Мультиплексирование ввода вывода

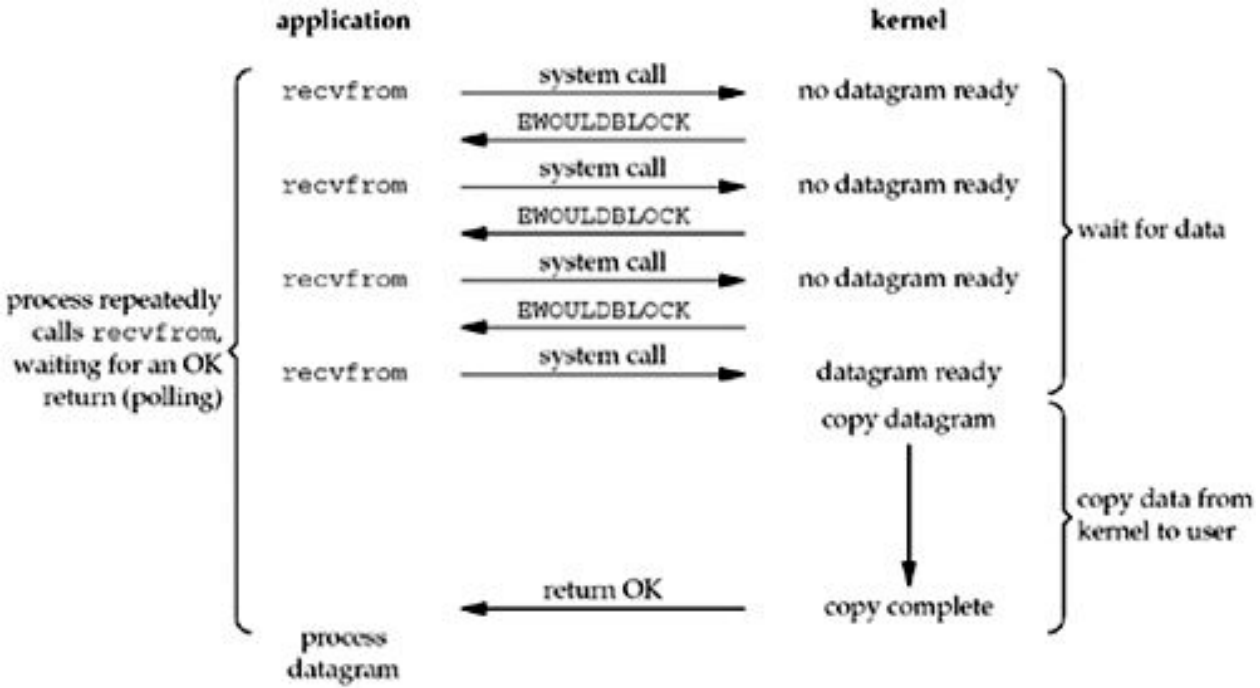
Когда?

1. Всегда, когда нужно использовать несколько дескрипторов *конкурентно* в рамках одного потока исполнения, к примеру:
   a. Клиент использует несколько дескрипторов: ввод с консоли и ввод/вывод из/в сокета
   b. Клиент с несколькими сокетами одновременно - web слиент
   c. Сервер, когда слушает сокет и одновременно обслуживает существующие соединения
   d. Если сервер, одновременно, обслуживает несколько транстпортов (к примеру TCP и UDP)
   e. Сервер обслуживает несколько сервисов/протоколов
2. Кроме того, мультиплексирование IO не огранниченно только сетью
   a. nodejs/golang используют асинхронную работу с файлами
   b. Некоторые базы данных также используют асинхронную работу с файлами

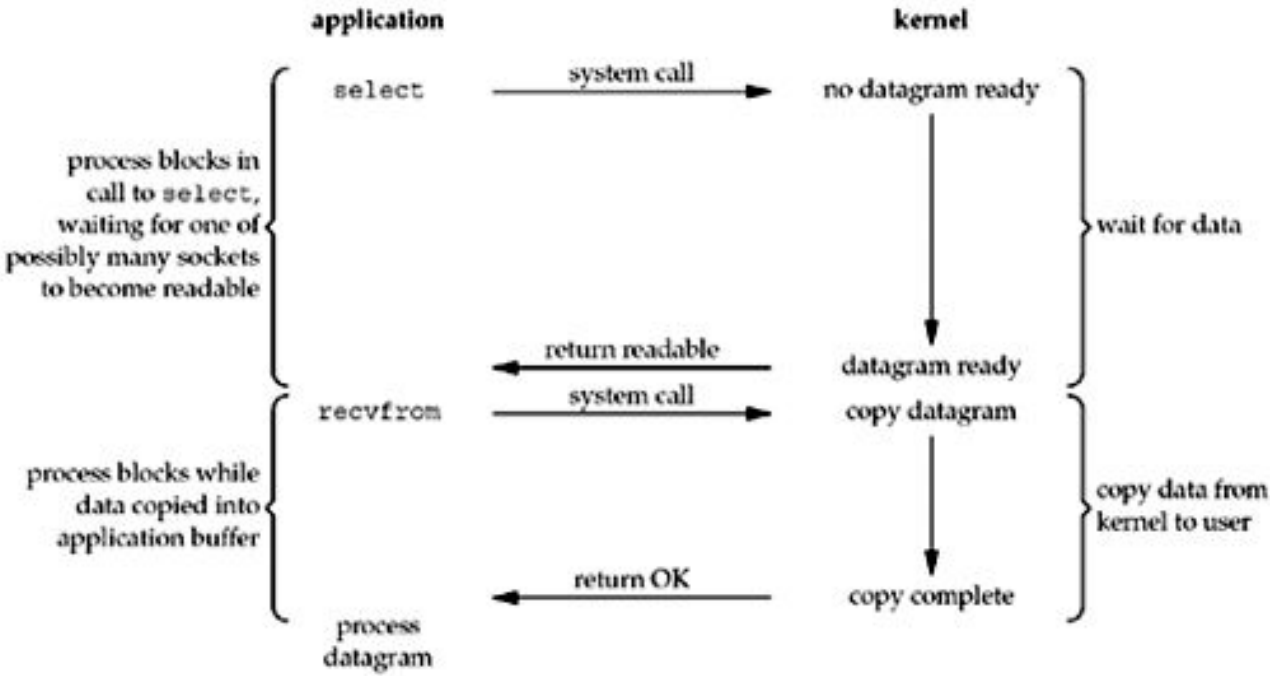
Read pipeline - blocking



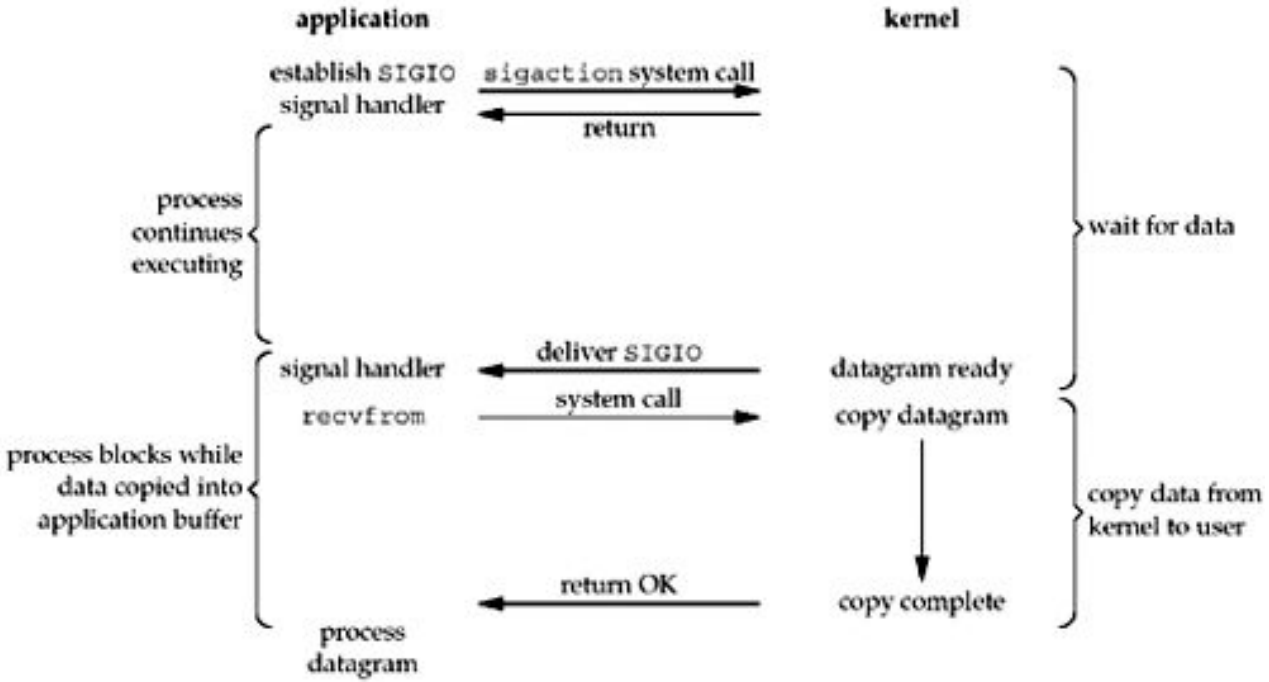
Read pipeline - nonblocking



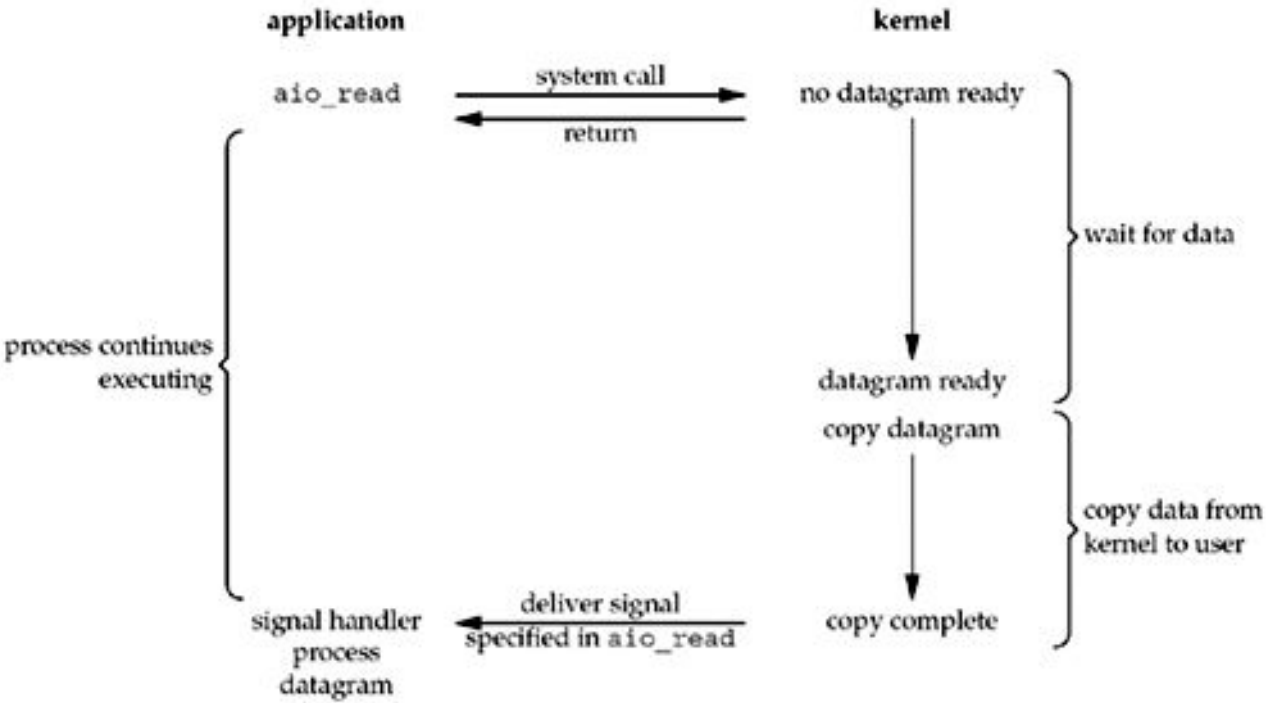
Read pipeline - multiplexed



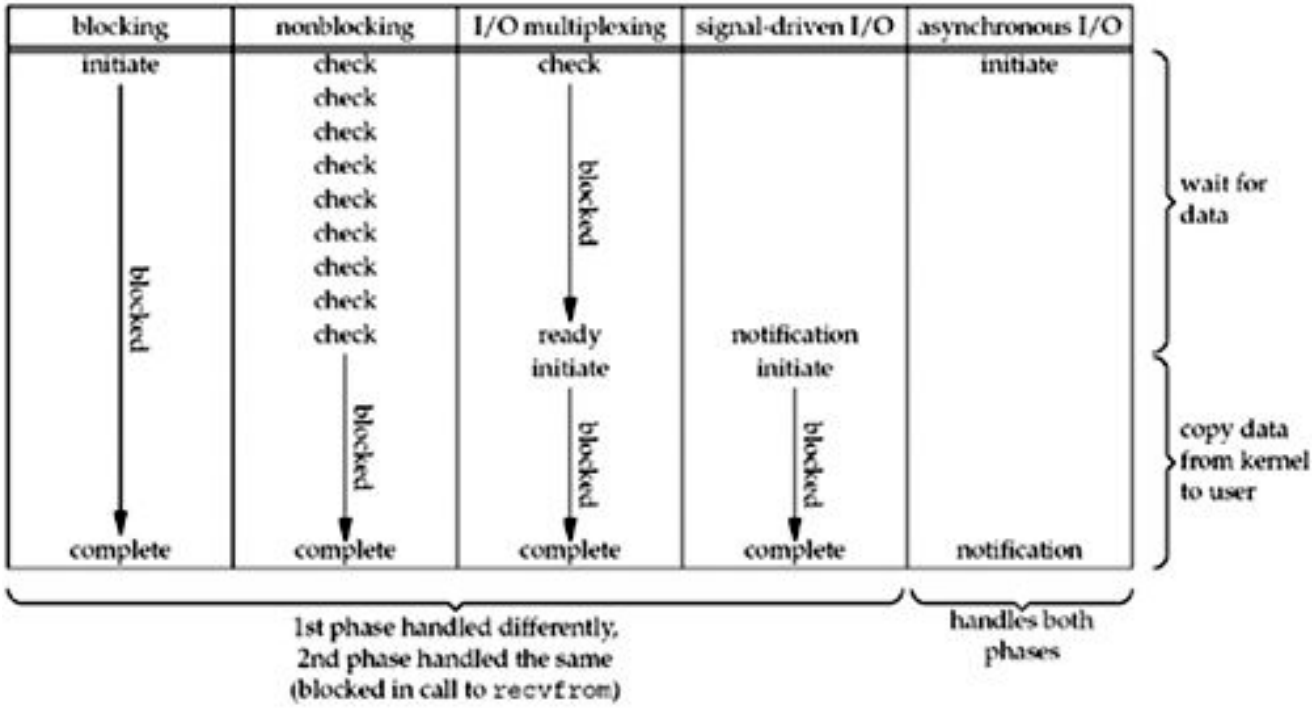
Read pipeline - signal driven



Read pipeline - async IO



Read pipeline - summary



Multiplexed IO

1. Неблокирующие сокеты
2. “Подписка” на события
    a. select() появился в 4.2BSD Unix в Августе 1983. В Linux доступен начиная с libc 5.4.28 (Май 1997)
    b. poll() добавили в SVR3 Unix, опубликованный в 1986. В Linux доступен начиная с 2.1.23 (Январь 1997)
    c. epoll() появился в ядре Linux 2.5.44(?). Создан для замены select/poll
    d. Kqueue() появился в FreeBSD 4.1, аналог epoll. Для Linux есть libkqueue обертка

Performance

# operations	poll	select	epoll
10	0.61	0.73	0.41
100	2.9	3.0	0.42
1000	35	35	0.53
10000	990	930	0.66