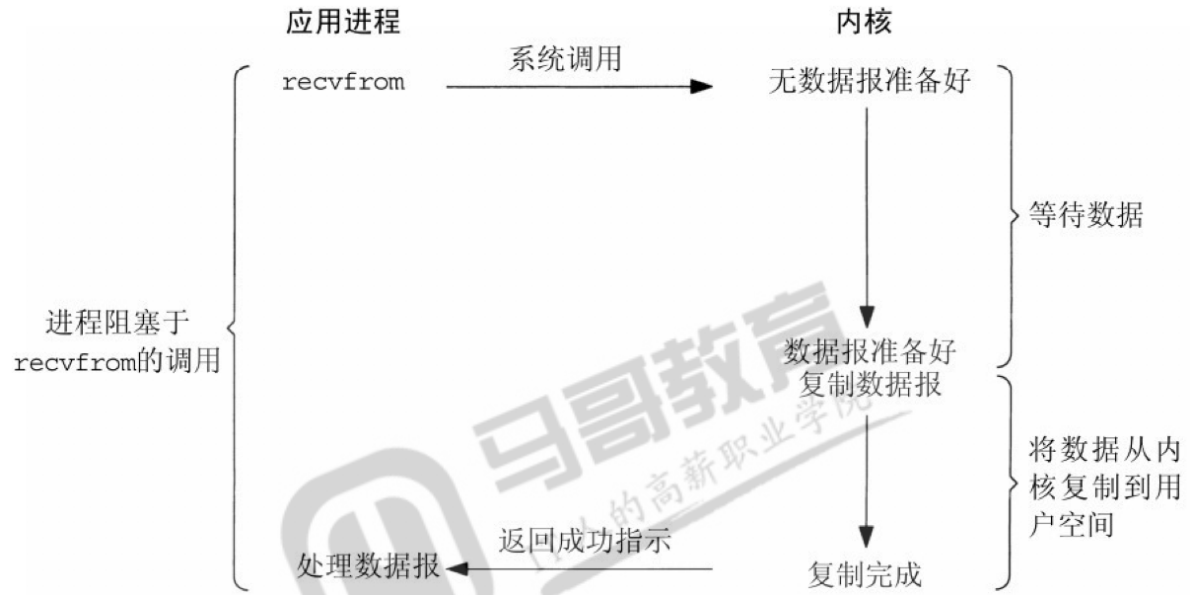
IO模型

IO两个阶段

IO过程分两阶段： 1、数据准备阶段。从设备读取数据到内核空间的缓冲区（淘米，把米放饭锅里煮饭） 2、内核空间复制回用户空间进程缓冲区阶段（盛饭，从内核这个饭锅里面把饭装到碗里来）

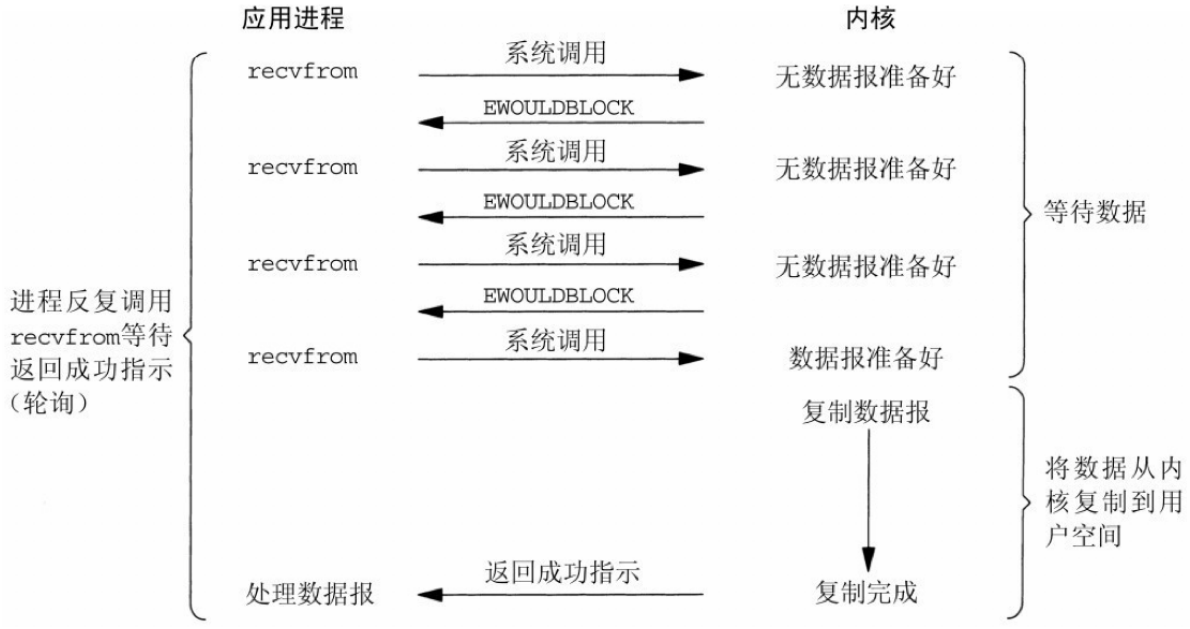
系统调用——read函数、recv函数等

同步阻塞IO



进程等待（阻塞），直到读写完成。（全程等待）

同步非阻塞IO

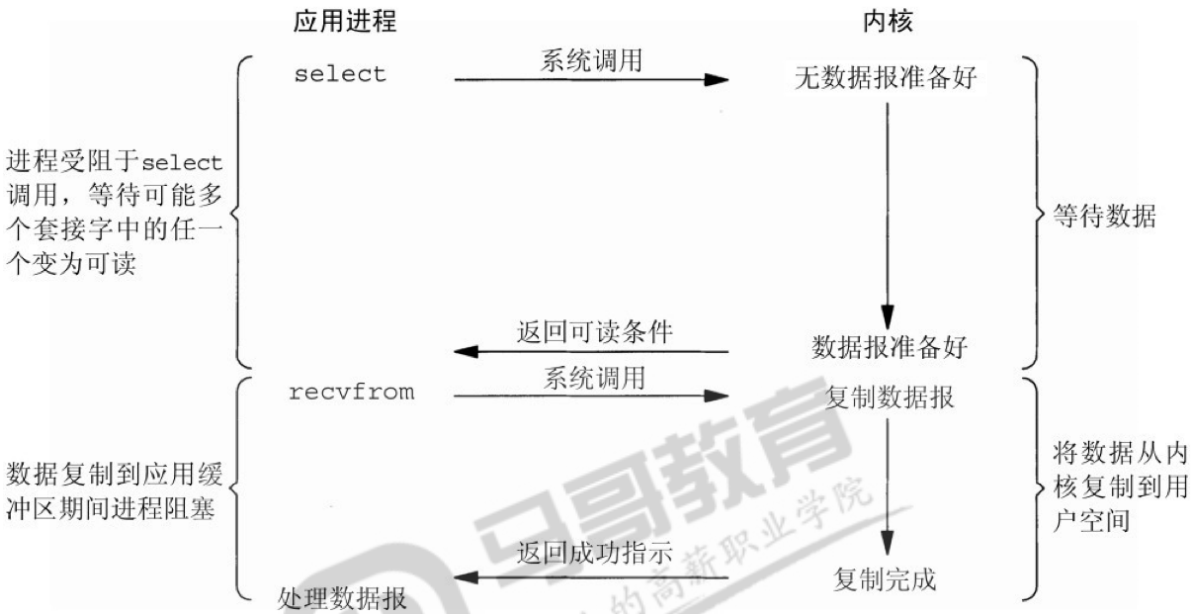


进程调用recvfrom操作，如果IO设备没有准备好，立即返回ERROR，进程不阻塞。用户可以再次发起系统调用（可以轮询），如果内核已经准备好，就阻塞，然后复制数据到用户空间。虽然不阻塞，但是不断轮询，CPU处于忙等。 淘米、蒸饭我不阻塞等，反复来询问，一直没有拿到饭。盛饭过程我等着你装好饭，但是要等到盛好饭才算完事，这是同步的，结果就是盛好饭。

IO多路复用

也称Event-driven IO。

所谓IO多路复用，就是利用操作系统提供的多路选择器（select/poll/epoll等）同时监控多个IO，称为多路IO，哪怕只有一路准备好了，就不需要等了就可以开始处理这一路的数据。这种方式提高了同时处理IO的能力。



select几乎所有操作系统平台都支持，poll是对的select的升级。 epoll，Linux系统内核2.5+开始支持，对select和poll的增强，在监视的基础上，增加回调机制。BSD、Mac平台有kqueue，Windows有iocp。

以select为例，将关注的IO操作告诉select函数并调用，进程阻塞，内核"监视"select关注的文件描述符fd，被关注的任何一个fd对应的IO准备好了数据，select返回。再使用read将数据复制到用户进程。

Epoll与select相比，解决了select监听fd的限制和O(n)遍历效率问题，提供回调机制等，效率更高。

实战：实现WEB服务器——IO多路复用版

- IO多路复用高级标准库selectors
- 注意selectors库对Windows只实现了select，效率不高，请在Linux、Unix系统运行

```
import selectors
import socket
from selectors import EVENT_READ

html = """\
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>magedu</title>
```

```
</head>
<body>
    <h1>马哥教育www.magedu.com -- Multiplexing</h1>
</body>
</html>\
""".encode()

response = """\
HTTP/1.1 200 OK
Date: Mon, 24 Oct 2022 20:04:23 GMT
Content-Type: text/html
Content-Length: {}
Connection: keep-alive
Server: wayne.magedu.com

""".format(len(html)).replace('\n', '\r\n').encode() + html

selector = selectors.DefaultSelector()
print(selector)  # Linux Epoll

def accept(server):
    conn, raddr = server.accept()
    conn.setblocking(False)  # 要非阻塞
    selector.register(conn, EVENT_READ, recv)

def recv(conn: socket.socket):
    try:
        data = conn.recv(1024)
        if not data:
            print(conn.getpeername(), "bye~~~")
            return selector.unregister(conn)
        conn.send(response)
    except Exception as e:
        print(e, "!!!!!!")
        selector.unregister(conn)

if __name__ == '__main__':
    server = socket.socket()
    server.setblocking(False)  # 要非阻塞
    laddr = ('0.0.0.0', 9999)
    server.bind(laddr)
    server.listen(1024)

    selector.register(server, EVENT_READ, accept)

    while True:
        for key, event in selector.select(): # 阻塞到有事件
            key.data(key.fileobj)
```

数数上例有几个线程?

马哥教育
IT人的高薪职业学院