

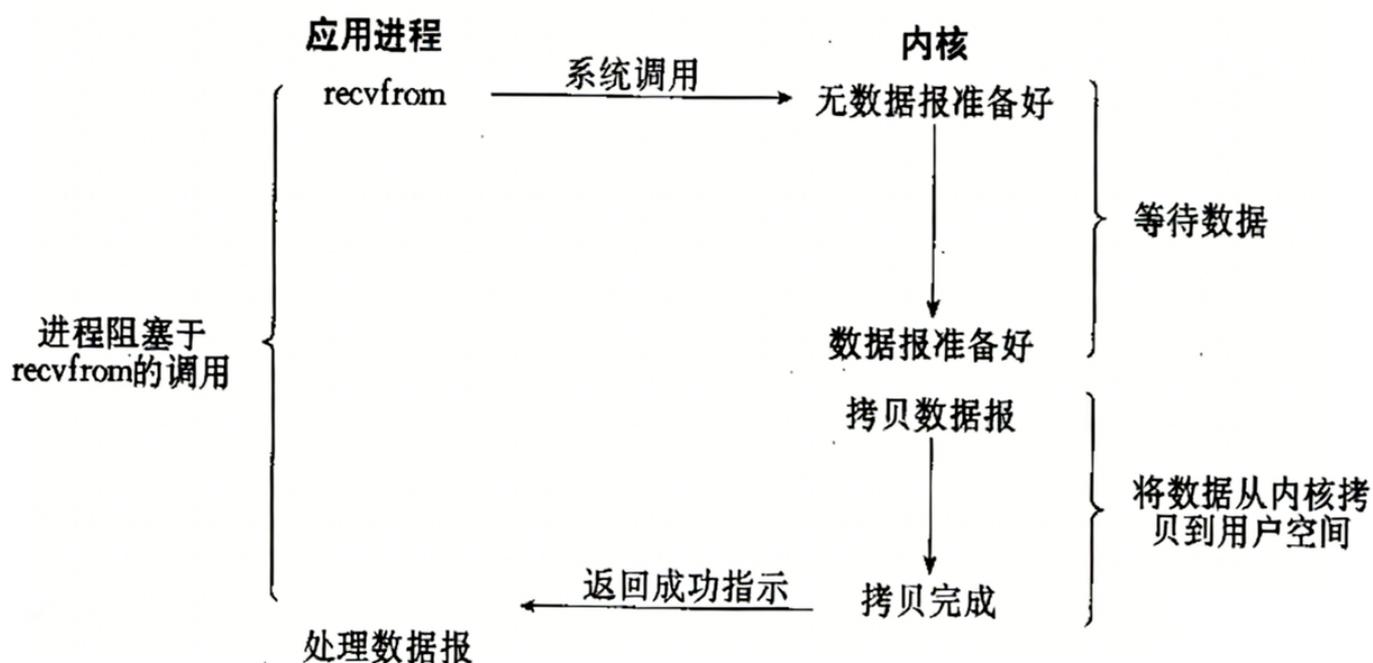
# 五种IO模型与非阻塞IO

## 五种IO模型

### [钓鱼例子]

· 阻塞IO: 在内核将数据准备好之前, 系统调用会一直等待. 所有的套接字, 默认都是阻塞方式.

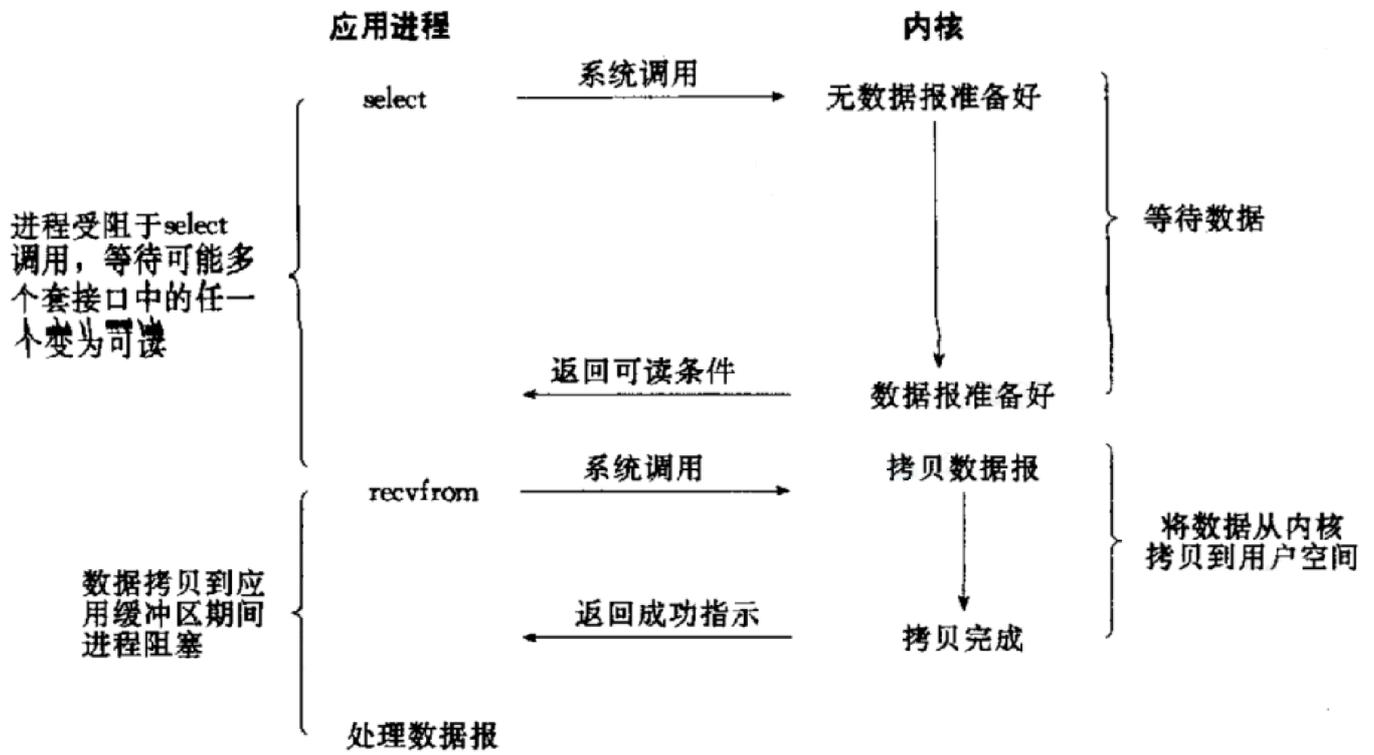
阻塞IO是最常见的IO模型.



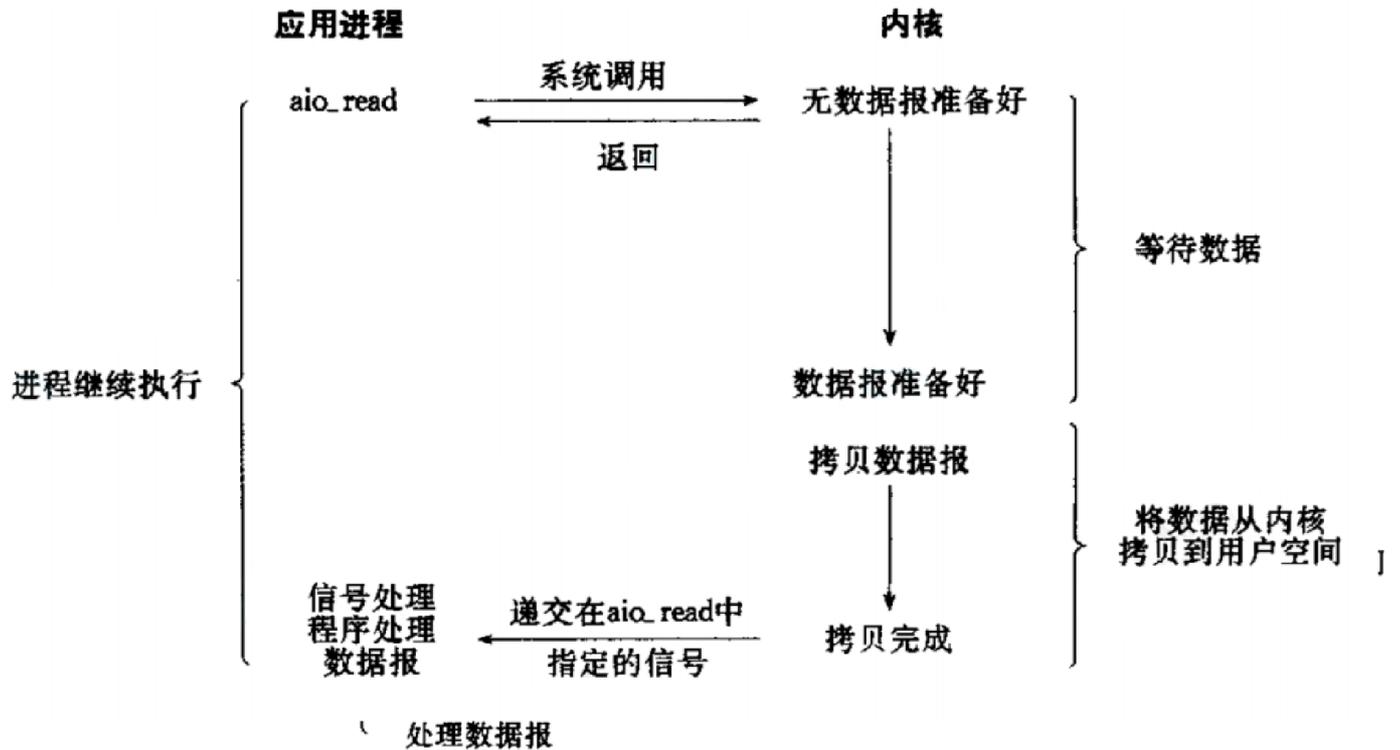
· 非阻塞IO: 如果内核还未将数据准备好, 系统调用仍然会直接返回, 并且返回EWOULDBLOCK错误码.

非阻塞IO往往需要程序员循环的方式反复尝试读写文件描述符, 这个过程称为轮询. 这对CPU来说是较大的浪费, 一般只有特定场景下才使用.





· 异步IO: 由内核在数据拷贝完成时, 通知应用程序(而信号驱动是告诉应用程序何时可以开始拷贝数据).



### 小结

· 任何IO过程中, 都包含两个步骤. 第一是等待, 第二是拷贝. 而且在实际的应用场景中, 等待消耗的时间往往都远远高于拷贝的时间. 让IO更高效, 最核心的办法就是让等待的时间尽量少.

# 高级IO重要概念

在这里, 我们要强调几个概念

## 同步通信 vs 异步通信(synchronous communication/ asynchronous communication)

同步和异步关注的是消息通信机制

- 所谓同步, 就是在发出一个**调用**时, 在没有得到结果之前, 该**调用**就不返回. 但是一旦调用返回, 就得到返回值了; 换句话说, 就是由**调用者**主动等待这个**调用**的结果;
- 异步则是相反, **调用**在发出之后, 这个调用就直接返回了, 所以没有返回结果; 换句话说, 当一个异步过程调用发出后, 调用者不会立刻得到结果; 而是在调用发出后, **被调用者**通过状态、通知来通知调用者, 或通过回调函数处理这个调用.

另外, 我们回忆在讲多进程多线程的时候, 也提到同步和互斥. 这里的同步通信和进程之间的同步是完全不相干的概念.

- 进程/线程同步也是进程/线程之间直接的制约关系
- 是为完成某种任务而建立的两个或多个线程, 这个线程需要在某些位置上协调他们的工作次序而等待、传递信息所产生的制约关系. 尤其是在访问临界资源的时候.

同学们以后在看到 "同步" 这个词, 一定要先搞清楚大背景是什么. 这个同步, 是同步通信异步通信的同步, 还是同步与互斥的同步.

## 阻塞 vs 非阻塞

阻塞和非阻塞关注的是程序在等待调用结果 (消息, 返回值) 时的状态.

- 阻塞调用是指调用结果返回之前, 当前线程会被挂起. 调用线程只有在得到结果之后才会返回.
- 非阻塞调用指在不能立刻得到结果之前, 该调用不会阻塞当前线程

## 理解这四者的关系

[妖怪蒸唐僧的例子]

## 其他高级IO

非阻塞IO, 纪录锁, 系统V流机制, I/O多路转接 (也叫I/O多路复用), readv和writev函数以及存储映射IO (mmap), 这些统称为高级IO.

我们此处重点讨论的是I/O多路转接

# 非阻塞IO

## fcntl

一个文件描述符, 默认都是阻塞IO.

函数原型如下.

代码块

```
1  #include <unistd.h>
2  #include <fcntl.h>
3  int fcntl(int fd, int cmd, ... /* arg */ );
```

传入的cmd的值不同, 后面追加的参数也不相同.

fcntl函数有5种功能:

- 复制一个现有的描述符 (cmd=F\_DUPFD) .
- 获得/设置文件描述符标记(cmd=F\_GETFD或F\_SETFD).
- 获得/设置文件状态标记(cmd=F\_GETFL或F\_SETFL).
- 获得/设置异步I/O所有权(cmd=F\_GETOWN或F\_SETOWN).
- 获得/设置记录锁(cmd=F\_GETLK,F\_SETLK或F\_SETLKW).

我们此处只是用第三种功能, 获取/设置文件状态标记, 就可以将一个文件描述符设置为非阻塞.

## 实现函数SetNoBlock

基于fcntl, 我们实现一个SetNoBlock函数, 将文件描述符设置为非阻塞

代码块

```
1  void SetNoBlock(int fd)
2  {
3      int fl = fcntl(fd, F_GETFL);
4      if (fl < 0)
5      {
6          perror("fcntl");
7          return;
8      }
9      fcntl(fd, F_SETFL, fl | O_NONBLOCK);
10 }
```

- 使用F\_GETFL将当前的文件描述符的属性取出来(这是一个位图).

· 然后再使用F\_SETFL将文件描述符设置回去. 设置回去的同时, 加上一个O\_NONBLOCK参数

## 轮询方式读取标准输入

代码块

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <fcntl.h>
4
5  void SetNoBlock(int fd)
6  {
7      int fl = fcntl(fd, F_GETFL);
8      if (fl < 0)
9      {
10         perror("fcntl");
11         return;
12     }
13     fcntl(fd, F_SETFL, fl | O_NONBLOCK);
14 }
15
16 int main()
17 {
18     SetNoBlock(0);
19     while (1)
20     {
21         char buf[1024] = {0};
22         ssize_t read_size = read(0, buf, sizeof(buf) - 1);
23         if (read_size < 0)
24         {
25             perror("read");
26             sleep(1);
27             continue;
28         }
29         printf("input:%s\n", buf);
30     }
31     return 0;
32 }
```