# Chapter 5: 回声客户端的完美实现：

### 为什么服务端没有问题？

服务端代码：

**while((str\_len = read(clnt\_sock, message, BUF\_SIZE)) != 0)**

**write(clnt\_sock, message, str\_len);**

由于TCP不存在数据边界

可能出现的情况：

情况一：

服务端有可能从服务器端收到多个字符串

情况二：

服务端有可能从服务器端收到一个字符串的部分（由于从客户端传输的数据包过大）

这样，服务端回传的message 可能大于或小于期望中字符串的长度。

但是如果客户端 write 字符串时已经记录下来 字符串的长度， 那么 客户端 read 的字符串总会是发出的字符串。

### 为什么是 while( recv\_len<str\_len) 而不是 while( recv\_len< strlen(message))？

**str\_len=write(sock, message, strlen(message));**

当一切顺利的时候， str\_len == strlen(message)， 但是 str\_len 有可能返回 -1（出错），此时 recv\_len< strlen(message) 恒成立，由于客户端向服务端 write 没有成功， 客户端 read 也就收不到值，一直处于阻塞状态。即使收到值，也会陷入无限循环中。

# Chapter 10: 进程服务器端：

### 什么是僵尸进程？

**一个进程使用fork创建子进程，如果子进程退出，而父进程并没有调用wait或waitpid获取子进程的状态信息，那么子进程的进程描述符仍然保存在系统中。这种进程称之为僵死进程。**

子进程的终止方式：

调用exit函数

执行return语句

# Chapter 11: 进程间通信：

### 什么是进程间通信?分别从概念上和内存的角度进行说明？

进程间通信( lnter Process Communication )意味着两个不同进程间可以交换数据，为了完成

这一点，操作系统中应提供两个进程可以同时访问的内存空间。

### 进程间通信需要特殊的IPC机制，这是由操作系统提供的。进程间通信时为何需要操作系统的帮助?

进程具有完全独立的内存结构。就连通过fork函数创建的子进程也不会与父进程共享内存空间。因此，进程间通信只能通过其他特殊方法完成

### “管道"是典型的IPC技法。关于管道，请回答如下问题。

1. **管道是进程间交换数据的路径。如何创建此路径?由谁创建?**

管道并非属于进程的资源，而是和套接字一样，属于操作系统(也就不是fork函数的复制对象)。所以，两个进程通过操作系统提供的内存空间进行通信

1. **为了完成进程间通信， 2个进程需同时连接管道。那2个进程如何连接到同一管道?**

**pipe 函数提供的接口中，参数为一个长度为2的int数组filedes， filedes[0] 通过管道接收数据时使用的文件描述符，即管道出口，filedes[1]通过管道传输数据时使用的文件描述符，即管道入口。**

1. **管道允许进行2个进程间的双向通信。双向通信中需要注意哪些内容?**

**数据进入管道后成为无主数据。也就是通过read函数先读取数据的进程将得到数据。即使该进程将数据传到了管道。向管道传递数据时，先读的进程会把数据取走。**

# Chapter 12: I/O 复用：

### 请解释复用技术的通用含义，并说明何为I/O复用。

**在1 个通信频道中传递多个数据(信号) 的技术。**

### 多进程并发服务器的缺点有哪些? 如何在I/O复用服务器端中弥补?

**创建进程时需要付出极大代价。这需要大量的运算和内存空间。**

**由于每个进程都具有独立的内存空间，所以相互间的数据交换也要求采用相对复杂的方法(IPC属于相对复杂的通信方法)。**

**I/O复用服务器弥补：**

**无论连接多少客户端，提供服务的进程只有1**

### (3) 复用服务器端需要select函数。下列关于select函数使用方法的描述错误的是?

### a. 调用select函数前需要集中I/O监视对象的文件描述符。

### b. 若己通过select函数注册为监视对象， 则后续调用select函数时无需重复注册。

### C. 复用服务器端同一时间只能服务于1 个客户端，因此，需要服务的客户端接入服务器端后只能等待。

# 第13 章 多种I/O 函数

### (1) 下列关于MSG\_OOB可选项的说法错误的是?

a. MSG\_OOB指传输Out-of-band数据，是通过其他路径高速传输数据。

b. MSG\_OOB指通过其他路径高速传输数据，因此，TCP中设置该选项的数据先到达对方主机。

c. 设置MSG\_OOB使数据先到达对方主机后，以普通数据的形式和顺序读取。也就是说，只是提高了传输速度，接收方无法识别这一点。

1. MSG\_OOB元法脱离TCP的默认数据传输方式。即使设置了MSG-OOB ，也会保持原有传输顺序。该选项只用于要求接收方紧急处理。

### 利用readv& writev函数收友数据有何优点?分别从函数调用次数和I/O缓冲的角度给出说明。

**通过writev函数可以将分散保存在多个缓冲中的数据一井发送，通过readv函数可以由多个缓冲分别接收。因此，适当使用这2个函数可以减少I/O 函数的调用次数**

### 通过recv函数验证输入缓冲是否存在数据时(确认后立即返回时)，如何设置recv函数最后一个参数中的可选项?分别说明各可选项的含义。

**同时设置MSG-PEEK选项和MSG-DONTWAIT选项，以验证输入缓冲中是否存在接收的数**

**据。设置MSG PEEK选项并调用recv 函数时，即使读取了输入缓冲的数据也不会删除。因此.该选项通常与MSG DONTWAIT合作，用于调用以非阻塞方式验证待读数据存在与否的函数**

# 第14 章 多播与广播

### TTL的含义是什么?请从路由器的角度说明较大的TTL值与较小的TTL值之间的区别及问题。

**TTL是Time to Live的简写，是决定"数据包传递距离"的主要因素。TTL用整数表示， 并且每经过1 个路由器就减1。TTL变为0时，该数据包无法再被传递，只能销毁。因此， TTL的值设置过大将影响网络流量。当然，设置过小也会无法传递到目标. 需要引起注意。**

### 多播与广播的异同点是什么?请从数据通信的角度进行说明。

**多播即使在跨越不同网络的情况下?只要加入多播组就能接收数据。相反，广播只能向同一网络中的主机传输数据。**

**多播与广播都是基于UDP 完成的。**

### (3) 下列关于多播的描述错误的是?

a. 多播是用来向加入多播组的所有主机传输数据的协议。

b. 主机连接到同一网络才能加入多播组，也就是说，多播组无法跨越多个网络。

c. 能够加入多播组的主机数并无限制，但只能有1个主机( Sender) 向该组发送数据。

d. 多播时使用的套接字是UDP套接字，因为多播是基于UDP进行数据通信的。

### (4) 多播也对网络流量有利，请比较TCP数据交换方式解释其原因。

**若通过TCP或UDP 向1000个主机发送文件， 则共需要传递1000次。即便将10 台主机合为1 个网络，使99%的传输路径相同的情况下也是如此。但此时若使用多播方式传输文件，则只需发送1 次。这时由1000台主机构成的网络中的路由器负责复制文件并传递到主机。**

# 第15 章 套接字和标准I/O

### 请说明标准I/O 函数的2个优点。它为何拥有这2个优点?

**标准I/O函数具有良好的移植性( Portability )。**

**标准I/O函数可以利用缓冲提高性能。**

**所有标准函数具有良好的移植性。因为，为了支持所有操作系统(编译器)，这些函数都是按照ANSI C标准定义的**

**需要传输的数据越多，有无缓冲带来的性能差异越大。可以通过如下两种角度说明性能的提高。传输的数据量 数据向输出缓冲移动的次数。**

### 利用标准I/O 函数传输数据时，下面的想法是错误的:

**"调用fputs 函数传输数据时，调用后应立即开始发送!”**

**为何说上述想法是错误的?为了达到这种效果应添加哪些处理过程?**

**标准I/O函数为了提高性能，内部提供额外的缓冲。因此，若不调用fflush函数则无法保证立即将数据传输到客户端**

# 第16 章 关于I/O流分离的其他内容

### (1) 下列关于FILE结构体指针和文件描述符的说法错误的是?

a. 与FILE结构体指针相同，文件描述符也分为输入描述符和输出描述符。

b. 复制文件描述符时将生成相同值的描述符，可以通过这2个描述符进行I/O 。

c. 可以利用创建套接字时返回的文件描述符进行I/O也可以不通过文件描述符，直接通过FILE结构体指针完成。

d. 可以从文件描述符生成FILE结构体指针，而且可以利用这种FILE 结构体进行套接字I/O。

解释:

想要通过FILE结构体指针进行套接字的 I/O 操作，必须先将套接字文件描述符转换为读模式的FILE指针和写模式的FILE指针，然后才能使用不同模式的FILE指针进行套接字的 I/O 操作。因此，c 中的说法错误。

e. 若文件描述符为读模式，则基于该描述符生成的FILE结构体指针同样是读模式; 若文件描述符为写模式，则基于该描述符生成的F1LE结构体指针同样是写模式。

### （2) EOF的发送相关描述中错误的是?

a. 终止文件描述符时发送EOF 。

b. 即使未完全终止文件描述符，关闭输出流时也会发送EOF 。

c. 如果复制文件描述符，则包括复制的文件描述符在内，所有描述符都终止时才会发送EOF。

d. 即使复制文件描述符，也可以通过调用shutdown函数进入半关闭状态并发送EOF。

# 第17 章 优于select 的epoll

### 利用select面数实现服务器端时，代码层面存在的2个缺点是?

调用select函数后常见的针对所有文件描述符的循环语句。

每次调用select函数时都需要向该函数传递监视对象信息。

### 无论是select方式还是epoll方式，都需要将监视对象文件描述符信息通过函数调用传递给操作系统。请解释传递该信息的原因。

**套接字是由操作系统管理，所以select和epoll函数绝对需要借助于操作系统才能完成功能**

### select方式和epoll方式的最大差异在于监视对象文件描述符传递给操作系统的方式。请说明具体的差异，并解释为何存在这种差异。

**仅向操作系统传递1 次监视对象，监视范围或内容发生变化时只通知发生变化的事项。**

**select方式中为了保存监视对象文件描述符，直接声明了"fd\_set变量。但epoll方式下由操作系统负责保存监视对象文件描述符，因此需要向操作系统请求创建保存文件描述符的空间，此时使用的函数就是epoll\_create。**

### 虽然epoll是select的改进方案，但select也有自己的优点。在何种情况下使用select方式更合理?

大部分操作系统都支持select函数。

### epoll以条件触发或边缘触发方式工作。二者有何区别?从输入缓冲的角度说明这2种方式通知事件的时间点差异。

条件触发方式只要输入缓冲有数据就会一直通知该事件。

从上述对话可以看出，边缘触发中输入缓冲收到数据时仅注册1 次该事件。即使输入缓冲中还留有数据，也不会再进行注册。

### 采用边缘触发时可以分离数据的接收和处理时间点。说明其原因及优点。

即使输入缓冲收到数据(注册相应事件)，服务器端也能决定读取和处理这些数据的时间点，这样就给服务器端的实现带来巨大的灵活性。

# 第18 章多线程服务器端的实现

### 单CPU系统中如何同时执行多个进程?请解释该过程中发生的上下文切换。

因为系统将CPU时间分成多个微小的块后分配给了多个进程。为了分时使用CPU ，需要"上下文切换"过程。运行程序前需要将相应进程信息读入内存，如果运行进程A后需要紧接着运行进程B ，就应该将进程A相关信息移出内存，并读入进程B相关信息。这就是上下文切换。但此时进程A的数据将被移动到硬盘，所以上下文切换需要很长时间。即使通过优化加快速度，也会存在一定的局

### 为何线程的上下文切换速度相对线程间数据交换为何不需要类似IPC的特别技术?

上下文切换时不需要切换数据区和堆

### 请从执行流角度说明进程和线程的区别。

进程在操作系统内部生成多个执行流，那么线程就在同一进程内部创建多条执行流

### (4) 下列关于临界区的说法错误的是?

### a. 11伍界区是多个线程同时访问时发生问题的区域。

### b. 线程安全的函数中不存在临界区，即便多个线程同时调用也不会发生问题。

### C. 1个临界区只能由1 个代码块，而非多个代码块构成。换言之，线程A执行的代码块A和

### 线程B执行的代码块B之间绝对不会构成临界区。

### d. 临界区由访问全局变量的代码构成。其他变量中不会发生问题。

### (5) 下列关于线程同步的描述错误的是?

a. 线程同步就是限制访问临界区。

b. 线程同步也具有控制线程执行顺序的含义。

c. 互斥量和信号量是典型的同步技术。

d. 线程同步是代替进程IPC的技术。

### (6) 请说明完全销毁Linux线程的2种方法。

调用pthread\_join函数。

调用pthread\_detach函数。