

进程池

- 定义数据结构: struct 结构体, 保存子进程信息: 子进程ID, 管道, 进程状态。

❖ 流程

- 增量编写, 每一步完成之后检查一下, 没有问题才继续写后面的功能

❖ 父进程

- 创建管道, 创建子进程并初始化, 记录子进程的信息 (管道, pid, busy)
- 初始化tcp: tcpInit (socket, bind, listen)
- 创建epoll, epoll监听所有管道和socketFd。

▼ 等待描述符就绪

- 如果客户端连接, newFd交给非忙碌子进程, 并标记为忙碌态。
- 如果管道可读, 说明子进程完成任务, 读管道, 并标记为空闲态。

❖ 子进程

- 读管道, 如果管道中没数据, 阻塞, 等待父进程发任务。
- 管道可读后, 拿到客户端newFd, 发文件给客户端。
- 发完文件后, 断开连接写管道, 通知父进程任务完成。继续等待下次任务

❖ 传输文件

❖ 流程

- 先发文件名
- 再发文件内容, 先read, 再send (newFd) recv,write
- 内容发完后, 发送文件结束标志

❖ ● 注意问题

▼ TCP粘包问题

- 问题描述:发送方多次send的数据停留在接收方的接收缓冲区里, 导致多次send的数据不能清晰的分辨出来, 造成混乱。

可以理解为发送速度快, 接收速度慢。

▼ 举例:

- 我们希望的流程是: 发送("hello"),接收("hello"),发送("world"),接收("world");
- 实际的流程可能是: 发送("hellowo"),接收("hellowo"),发送(rld"),接收("rld");

▼ 解决方法:

- 小火车: 增加控制信息, 火车头记录发送字节数, 火车车厢装真正要发的数

据。

- 接收方接收的时候，先接火车头，按照火车头记载的长度精确去接后面的数据。这样就不会把多次发送的数据接到一起，造成混淆

▼ 发送速度慢，接收速度快问题：

- 问题描述：接收方想接1000个字节，但是发送方发送速度慢，接收方的接收缓冲区中只有300个字节，recv读了300个字节之后，就会返回。

▼ 举例

- 发送方：本来每次要发4个字节火车头+1000个字节火车内容，但是第一次只发了4个字节+300个字节，第二次发了700个字节，第三次发4个字节火车头+1000字节内容。
- 接收方：本来每次要接4个字节火车头+1000个字节火车内容，但是发送方发的慢，第一次到缓冲区取数据的时候，缓冲区只有4个字节头部和300个字节内容，此时虽然没有接完1004个字节，但是缓冲区中有数据，recv也会返回，返回值为104，下一次再接的时候还是先接火车头的4个字节，但是此时缓冲区中可能只有第二次发过来的700个字节内容，因此会把这700个字节里的前4个字节当做火车头来接。这就导致数据错乱。

▼ 解决方法：recvCycle(int fd,void *buf, int recvLen)

- 定义一个函数，只有真正从缓冲区读了recvLen个字节的数据后，才返回，代替recv，这样就能够保证接到足够的数据。

▼ 断开：

- 服务器给客户端发送文件，如果发送过程中，客户端断开，服务器仍在发送文件，第一次send会返回-1，第二次send时调用send的子进程会收到SIGPIPE信号，子进程收到信号终止之后，父子进程间的管道会变成就绪状态，父进程的epoll会一直唤醒。为了避免子进程收到信号终止，每次send判断send的返回值是否为-1，如果是-1，就停止发送，直接返回。

▼ 提升效率：“零”拷贝

更加现实的情况是，系统的性能瓶颈往往是网络IO

- read+send的缺点：先从内核缓冲区读到用户态缓冲区，又从用户态缓冲区拷贝到socket缓冲区，多次拷贝，效率低。
- mmap方式：将内核缓冲区与用户共享，这样能减少内核缓冲区到用户缓冲区的之间的复制。

▼ sendfile：

- sendfile在内核里把数据从一个描述符缓冲区拷贝到另一个描述符缓冲区，不需要经历，从内核拷贝到用户空间的过程，因此效率更高。
- sendfile的一个限制是只能用于发送端，另一个限制是不能发送超过2G的文件。

▼ splice：

- splice可以从两个文件描述符之间拷贝数据而不用经过用户地址空间

▼ 进程池的退出：

- 给父进程发信号，通知父进程退出，父进程收到信号后，在信号处理函数里通过写管道的方式，唤醒epoll，epoll监听的管道读端可读，通知子进程退出。
- ▼ 子进程退出方式1：
 - 遍历的方式给子进程发信号，子进程收到信号终止，父进程回收子进程资源，然后退出。
 - 通过管道，给子进程发一条信息。子进程每次读管道时，如果发现退出标志位是1，就退出