设计说明

一. 概述

本程序主要是演示客户端与服务器端的通信，其中使用的传输层协议是TCP，客户端向服务器端发送一个字符串，服务器端接收到字符串后将其输出以展示收到了该字符串。为了能允许尽可能多的客户端同时向服务器发送数据，服务器端的设计采用多进程以及多路 IO 转接的技术来提高系统容量。其中，进程的数量可以根据系统中 CPU 的数量动态决定，多路 IO 转接由最近引入的 epoll 系列系统调用来实现。

服务器主进程首先建立网络套接字，然后产生大约 2 倍于系统 CPU 数量的子进程，每个子进程通过一个 UNIX 域套接字与父进程通信，最后主进程调用 accept 系统调用进入等待客户端发起连接请求的状态。每当有一个连接请求被服务器端的主进程接收到，主进程则通过轮询的方式把该连接请求的描述符通过 UNIX 域套接字发送给某个子进程，然后又进入等待连接请求的状态。

服务器子进程首先把绑定到标准输入的 UNIX 域套接字加入到 epoll 系统调用的等待事件集中，然后调用 epoll_wait()进入阻塞状态，当有新的套接字描述符通过标准输入传给该子进程时，该子进程从标准输入中读取套接字描述符，并把该套接字描述符加入到 epoll 的等待事件集中。依此类推，从而使得在一个进程中便可以处理很多的套接字链接，避免了建立一个套接字链接就开一个进程所带来的大量进程切换造成的开销。

二. 程序的编译与运行方法

1. 客户端程序的编译({软件目录}/tecent/src/下)

```
gcc client.c -o client
```

2. 服务器端程序的编译({软件目录}/tecent/src/server/下)

子进程程序的编译：

```
gcc helper.c channel.c load_server.c -o load_server
```

主进程程序的编译：

```
gcc helper.c channel.c proxy_server.c main.c -o qqserver
```

3. 运行环境的配置

在/etc/services 配置文件中加入如下一行

```
tecent   62000/tcp   tecent  # demo for high performance server
```

4. 程序的运行
    1. 开启服务器程序({软件目录}/tecent/src/server/下)

```
./qqserver
```

2. 生成大量客户端链接({软件目录}/tecent/src/下)

```
./hugeload.sh
```

三. 客户端设计说明

客户端程序只是简单的向服务器请求一个链接，然后向该链接发送一个字符串，因此，程序结构非常简单，在此不再做进一步说明。

四. 服务器端设计说明

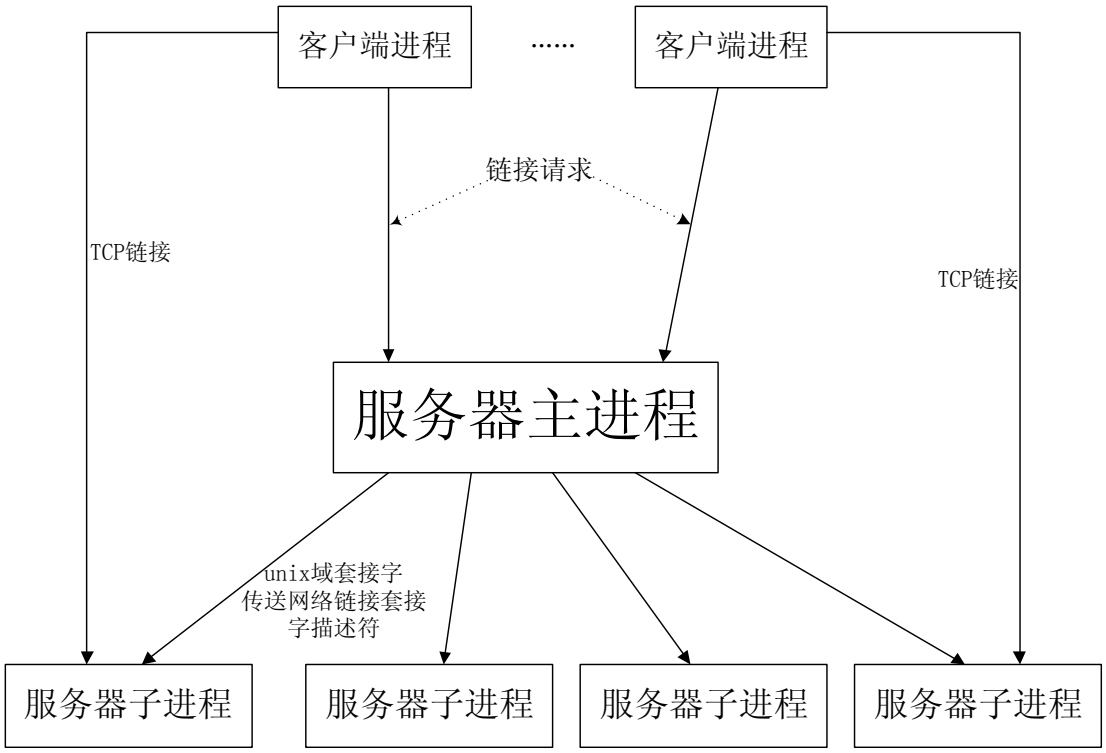

图-1 服务器架构说明

服务器主进程由 prox_server.c 实现，主要由 init_server()和 serve()两个函数，其中函数 init_server()实现因特网套接字的建立，地址的绑定等操作，函数 serve()则负责服务器子进程的生成，与子进程通信的 UNIX 域套接字的建立，以及监听客户端的请求，并把与客户端建立的 TCP 链接转移给某个子进程，由子进程实际负责与客户端通信。

服务器子进程由 load_server.c 实现，主要是有 epoll 系列系统调用实现 IO 多路转接(IO Multiplexing)。从而使得一个子进程可以与大量的客户端进程高效通信。

传送文件描述符的工作主要有 channel.c 实现，其中 send_fd()发送文件描述符，recv_fd()接受文件描述符。

最后整个服务器程序由 main.c 函数驱动。