Readme

第五組

組長：509411026 程毓盛

組員：509411008 蔡宜紋、509411011 魯恆均、509411018 莊詩婷

1 TABLE OF CONTENTS

2 運作說明：.....1
3 步驟:.....1
4 使用函式庫：.....2
4.1 socket.....2
4.2 bind.....2
4.3 listen.....3
4.4 connect.....3
4.5 accept.....3
4.6 send.....3
4.7 recv.....4
4.8 close.....5
4.9 fflush.....5
4.10 bzero.....5

2 運作說明：

使用 LINUX 環境 C 語言寫一對程式，分別為 TCP 的 client 端與 server 端，server 端在收到 client 端的字串後，會自動回傳相同的字串給 client 端

3 步驟:

1. 執行 server 程式, 等候
2. 執行 client 程式, 輸入傳送至 server 字串, ex.[ABC]
3. Server 端接收到字串[ABC],再回傳相同字串[ABC]給 Client 端
4. Client 端收到[ABC]字串

4 使用函式庫：

4.1 SOCKET

int socket(int domain,int type, int protocol)

返回值:

非負值成功, -1 出錯

其中 family 指明瞭協議族/域,通常 AF_INET、AF_INET6、AF_LOCAL 等;Type 是套介面型別,主要 SOCK_STREAM、SOCK_DGRAM、SOCK_RAW;

protocol 一般取為 0。成功時,返回一個小的非負整數值,與檔案描述符類似。

4.2 BIND

Int bind(int sockfd,const struct sockaddr* myaddr,socklen_t addrlen)

返回值:

0 成功, -1 出錯

當 socket 函式返回一個描述符時,只是存在於其協議族的空間中,並沒有分配一個具體的協議地址(這裡指 IPv4/IPv6 和埠號的組合),bind 函式可以將一組固定的地址繫結到 sockfd 上。

其中:

sockfd 是 socket 函式返回的描述符;

myaddr 指定了想要繫結的 IP 和埠號,均要使用網路位元組序-即大端模式;

addrlen 是前面 struct sockaddr(與 sockaddr_in 等價)的長度。

為了統一地址結構的表示方法,統一介面函式,使得不同的地址結構可以被 bind()、connect()、recvfrom()、sendto()等函式呼叫。但一般的程式設計中並不直接對此資料結構進行操作,而使用另一個與之等價的資料結構 sockaddr_in。

通常伺服器在啟動的時候都會繫結一個眾所周知的協議地址,用於提供服務,客戶就可以通過它來接連伺服器;而客戶端可以指定 IP 或埠也可以都不指定,未分配則系統自動分配。這就是為什麼通常伺服器端在 listen 之前會呼叫 bind(),而客戶端就不會呼叫,而是在 connect()時由系統隨機生成一個。

4.3 LISTEN

Int listen(int sockfd,int backlog)

返回值:

0 成功, -1 出錯

4.4 CONNECT

int connect(int sockfd,conststruct sockaddr *addr, socklen_t addrlen)

返回值:

0 成功, -1 出錯

通過此函式建立於 TCP 伺服器的連線,實際是發起三次握手過程,僅在連線成功或失敗後返回。引數 sockfd 是本地描述符,addr 為伺服器地址,addrlen 是 socket 地址長度。

4.5 ACCEPT

Int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen)

返回值:

非負值成功, -1 出錯

4.6 SEND

send(int sockfd,constvoid *buf, size_t len,int flags)

返回值:

>0 - 成功拷貝至傳送緩衝區的位元組數(可能小於 len),

-1 - 出錯,並置錯誤號 errno.

其中:

sockfd:傳送端套接字描述符(非監聽描述符)

buf:應用要傳送資料的快取

len:實際要傳送的資料長度

flag:一般設定為 0

每個 TCP 套介面都有一個傳送緩衝區,它的大小可以用 SO_SNDBUF 這個選項來改變。呼叫 send 函式的過程,實際是核心將使用者資料拷貝至 TCP 套介面的傳送緩衝區的過程:若 len 大於傳送緩衝區大小,則返回-1;否則,檢視緩衝區剩餘空間是否容納得下要傳送的 len 長度,若不夠,則拷貝一部分,並返回拷貝長度(指的是非阻塞 send,若為阻塞 send,則一定等待所有資料拷貝至緩衝區才返回,因此阻塞 send 返回值必定與 len 相等);若緩衝區滿,則等待發送,有剩餘空間後拷貝至緩衝區;若在拷貝過程出現錯誤,則返回-1。關於錯誤的原因,檢視 errno 的值。

如果 send 在等待協議傳送資料時出現網路斷開的情況,則會返回-1。注意:send 成功返回並不代表對方已接收到資料,如果後續的協議傳輸過程中出現網路錯誤,下一個 send 便會返回-1 傳送錯誤。TCP 給對方的資料必須在對方給予確認時,方可刪除傳送緩衝區的資料。否則,會一直快取在緩衝區直至傳送成功(TCP 可靠資料傳輸決定的)。

4.7 RECV

recv(int sockfd,void *buf, size_t len,int flags)

其中:

sockfd:接收端套接字描述符;

buf:指定緩衝區地址,用於儲存接收資料;

len:指定的用於接收資料的緩衝區長度;

flags:一般指定為 0

表示從接收緩衝區拷貝資料。成功時,返回拷貝的位元組數,失敗返回-1。阻塞模式下,recv/recvfrom 將會阻塞到緩衝區裡至少有一個位元組(TCP)/至少有一個完整的 UDP 資料報才返回,沒有資料時處於休眠狀態。若非阻塞,則立即返回,有資料則返回拷貝的資料大小,否則返回錯誤-1,置錯誤碼為 EWOULDBLOCK。

4.8 CLOSE

close 預設功能是將套接字作“已關閉”標記,並立即返回到呼叫程序,該套接字描述符不能再為該程序所用:即不能作為 read 和 write(send 和 recv)的引數,但是 TCP 將試著傳送傳送緩衝區內已排隊待發的資料,然後按正常的 TCP 連線終止序列進行操作(斷開連線 4 次握手-以 FIN 為首的 4 個 TCP 分節)。

4.9 FFLUSH

fflush:是把 c 庫中的緩衝呼叫 write 函式寫到磁碟[其實是寫到核心的緩衝區

4.10 BZERO

sin_zero 用來將 sockaddr_in 結構填充到與 struct sockaddr 同樣的長度，可以用 bzero()或 memset()函數將其置為零。