**Readme**

第七組

組長：509411026 程毓盛

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

Table of Contents

[2 運作說明： 1](#_Toc85576022)

[3 步驟: 1](#_Toc85576023)

[4 使用函式庫： 2](#_Toc85576024)

[4.1 socket 2](#_Toc85576025)

[4.2 bind 2](#_Toc85576026)

[4.3 listen 3](#_Toc85576027)

[4.4 connect 3](#_Toc85576028)

[4.5 accept 3](#_Toc85576029)

[4.6 send 3](#_Toc85576030)

[4.7 recv 4](#_Toc85576031)

[4.8 close 5](#_Toc85576032)

[4.9 fflush 5](#_Toc85576033)

[4.10 bzero 5](#_Toc85576034)

# 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

# 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

# 使用函式庫：

## 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。

## 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。

## listen

Int listen(int sockfd,int backlog)

返回值:

0成功, -1 出錯

## 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。

## accept

Int accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen)

返回值:

非負值成功, -1 出錯

## 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)。

## 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。

## 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)。

## 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

## bzero

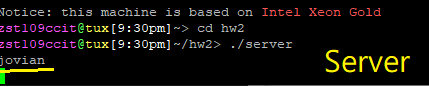
sin\_zero用來將sockaddr\_in結構填充到與struct sockaddr同樣的長度，可以用bzero()或memset()函數將其置為零。

中正資工 網路程式設計 HW2 509411018 莊詩婷 2021/10/21

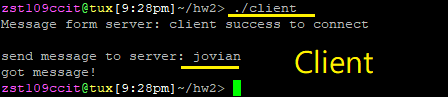
一．程式: 如附檔HW2\_Group7\_509411018.rar

執行結果

Server:



Client:



二．利用Wireshark抓到TCP三項交握的封包

