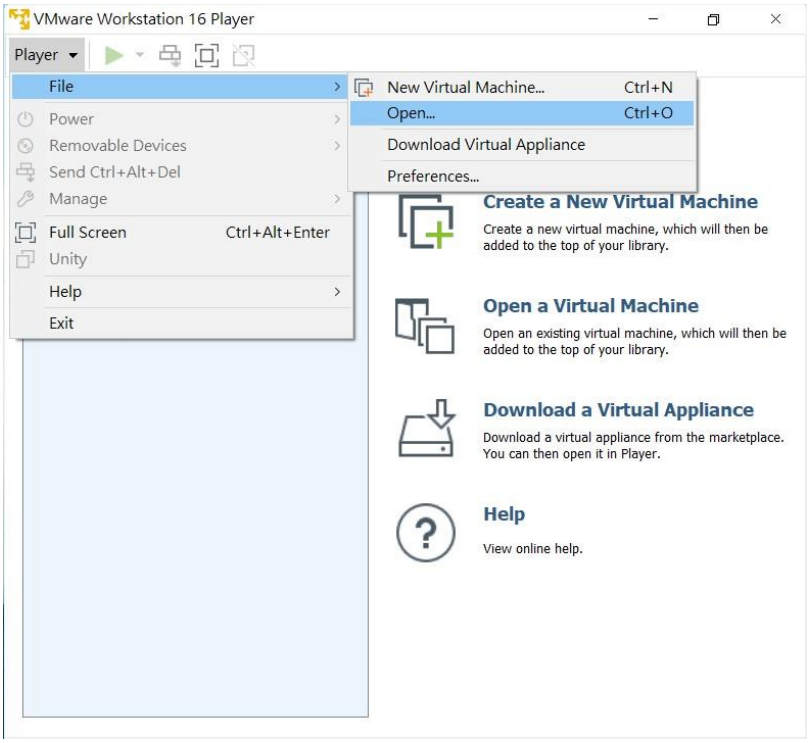
EECS302002 計算機網路概論

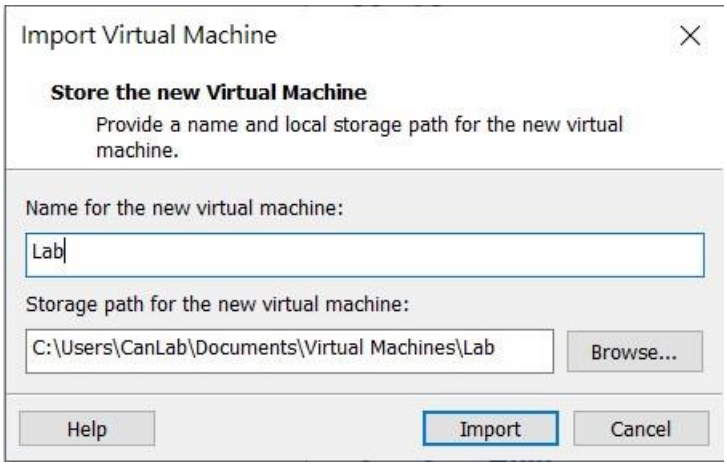
Linux socket Tutorial

You need to install Linux prepared by TA in Step 1.

1. Install a virtual machine of Linux by following the instructions:
   - Download
   - Virtual disk file

   Download ubuntu 20.04 LTS（Linux 常用指令 - HackMD）

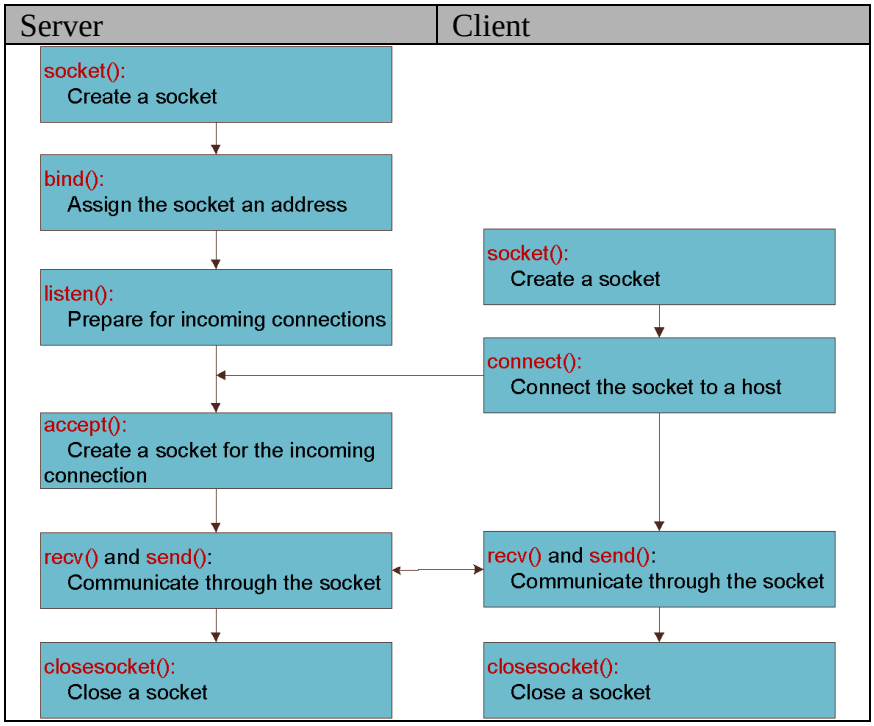




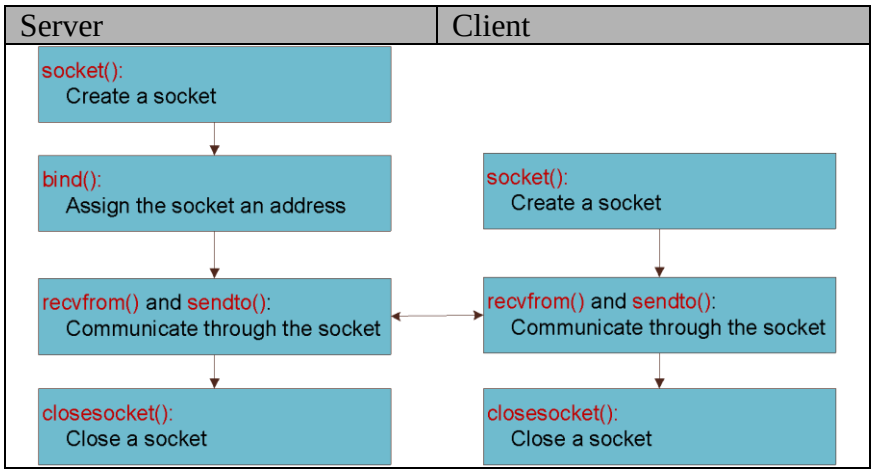
Password: canlab

2. Socket programming

a. TCP flow chart



b. UDP flow chart



c. Data structure of address

Structure	Usage

struct sockaddr_in {
sa_family_t sin_family;
in_port_t sin_port;
struct in_addr sin_addr;
};
	sin_family = AF_INET;
	sin_port = htons(8080);
	sin_addr.s_addr = INADDR_ANY; (for server)

*htons(unsign short port): host to network byte order for unsign short (16 bits)
*Use domain name instead of IP as the address

- struct sockaddr_in serverAddress
- struct hostent *h = gethostbyname(hostName)
- memcpy(&serverAddress.sin_addr, h->h_addr_list[0], h->h_length)

d. Functions

socket(AF_INET, SOCK_STREAM, 0) Create a TCP socket (SOCK_STREAM) or UDP socket (SOCK_DGRAM)
bind(server_fd, (struct sockaddr *)&address, sizeof(address)) Assign serverSocket serverAddress
listen(server_fd, 3) Prepare for incoming connections (maximum 3 connections)
accept(server_fd, (struct sockaddr *)&address,(socklen_t*)&addrlen)) Create a socket for the incoming connection, and the address of the target host is stored in address
send(new_socket , hello , strlen(hello) , 0) Send buf(hello) of size len (TCP socket)
read(new_socket , buffer, 1024) read data of maximum size 1024, and store the data in buffer (TCP socket)
sendto(sockfd, (const char *)hello, strlen(hello),MSG_CONFIRM, (const struct sockaddr *) &cliaddr,len); Send buf of size len to client (UDP socket)
recvfrom(sockfd, (char *)buffer, MAXLINE,MSG_WAITALL, (struct sockaddr *) &cliaddr,&len); Receive data of maximum size MAXLINE, and store the data in buffer (UDP socket)

3. Examples
   a. TCP echo server and client

TCP echo server

```
// Server side C/C++ program to demonstrate Socket programming
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 8080
int main(int argc, char const *argv[])
{
    int server_fd, new_socket, valread;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char *hello = "Hello from server";

    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }
    // Forcefully attaching socket to the port 8080
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,&opt,
sizeof(opt)))
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons( PORT );
    // Forcefully attaching socket to the port 8080
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address))<0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0)
    {
        perror("listen");
        exit(EXIT_FAILURE);
    }
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address,(socklen_t*)&addrlen))<0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }
    valread = read( new_socket , buffer, 1024);
    printf("%s\n",buffer );
    send(new_socket , hello , strlen(hello) , 0 );
    printf("Hello message sent\n");
    return 0;
}
```

TCP echo client

```
// Client side C/C++ program to demonstrate Socket programming
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8080

int main(int argc, char const *argv[])
{
    int sock = 0, valread;
    struct sockaddr_in serv_addr;
    char *hello = "Hello from client";
    char buffer[1024] = {0};
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Socket creation error \n");
        return -1;
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    // Convert IPv4 and IPv6 addresses from text to binary form
    if(inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)<=0)
    {
        printf("\nInvalid address/ Address not supported \n");
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
        printf("\nConnection Failed \n");
        return -1;
    }
    send(sock , hello , strlen(hello) , 0 );
    printf("Hello message sent\n");
    valread = read( sock , buffer, 1024);
    printf("%s\n",buffer );
    return 0;
}
```

b. UDP echo server and client

UDP echo server

```
// Server side implementation of UDP client-server model
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT   8080
#define MAXLINE 1024

// Driver code
int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello from server";
    struct sockaddr_in servaddr, cliaddr;

    // Creating socket file descriptor
    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));

    // Filling server information
    servaddr.sin_family    = AF_INET; // IPv4
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    // Bind the socket with the server address
    if ( bind(sockfd, (const struct sockaddr *)&servaddr,sizeof(servaddr)) < 0 )
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    int len, n;

    len = sizeof(cliaddr);  //len is value/resuslt

    n = recvfrom(sockfd, (char *)buffer, MAXLINE,MSG_WAITALL, ( struct sockaddr *) &cliaddr,&len);
    buffer[n] = '\0';
    printf("Client : %s\n", buffer);
    sendto(sockfd, (const char *)hello, strlen(hello),MSG_CONFIRM, (const struct sockaddr *)
&cliaddr,len);
    printf("Hello message sent.\n");

    return 0;
}
```

UDP echo client

```
// Client side implementation of UDP client-server model
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT    8080
#define MAXLINE 1024

// Driver code
int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello from client";
    struct sockaddr_in      servaddr;

    // Creating socket file descriptor
    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));

    // Filling server information
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;

    int n, len;

    sendto(sockfd, (const char *)hello, strlen(hello),MSG_CONFIRM, (const struct sockaddr *)
&servaddr, sizeof(servaddr));
    printf("Hello message sent.\n");

    n = recvfrom(sockfd, (char *)buffer, MAXLINE,  MSG_WAITALL, (struct sockaddr *) &servaddr,
&len);
    buffer[n] = '\0';
    printf("Server : %s\n", buffer);

    close(sockfd);
    return 0;
}
```