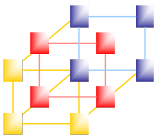
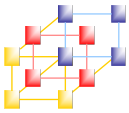


Unit 2
Client/Server Model

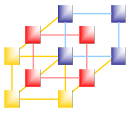
1-machine_info.py
2-remote_addr.py



Machine's name and IP address

- Defined in `socket` library
- `gethostname()` – Return the current host name
- `gethostbyname()` – Return the string of IP address for a host

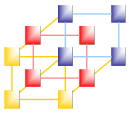
```
>>> import socket
>>> host_name = socket.gethostname()
>>> print('Host name: %s' % (host_name))
Host name: JackdeMac
>>> print('IP address: %s' %
(socket.gethostbyname(host_name)))
IP address: 127.0.0.1
>>> fcuwww = socket.gethostbyname('www.fcu.edu.tw')
>>> print('FCU WWW IP: %s' % (fcuwww))
FCU WWW IP: 52.76.146.103
```



Convert IPv4 address format

- `socket.inet_aton()` – string to int
- `socket.inet_ntoa()` – int to string

```
>>> import socket
>>> ipstr = '192.168.2.1'
>>> ipv4 = socket.inet_aton(ipstr)
>>> print(ipv4)
b'\xc0\xa8\x02\x01'
>>> string = socket.inet_ntoa(ipv4)
>>> print(string)
192.168.2.1
```

Host/Network order converting

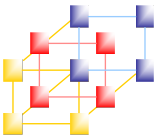
- `socket.ntohl(), socket.ntohs()`
  - network byte order to host byte order
- `socket.htonl(), socket.htons()`
  - host byte order to network byte order

```
0x0A0B0C0D ←→ 168496141
0x0D0C0B0A ←→ 218893066
>>> Int32=168496141
>>> socket.htonl(Int32)
218893066
>>> socket.ntohl(218893066)
168496141
```

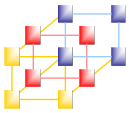
0x0A0B0C0D

Little Endian	0D	0C	0B	0A
Address:	x	x+1	x+2	x+3
Big Endian	0A	0B	0C	0D
Address:	x	x+1	x+2	x+3

Intel CPUs are little-endian

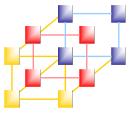


The TCP client-server model

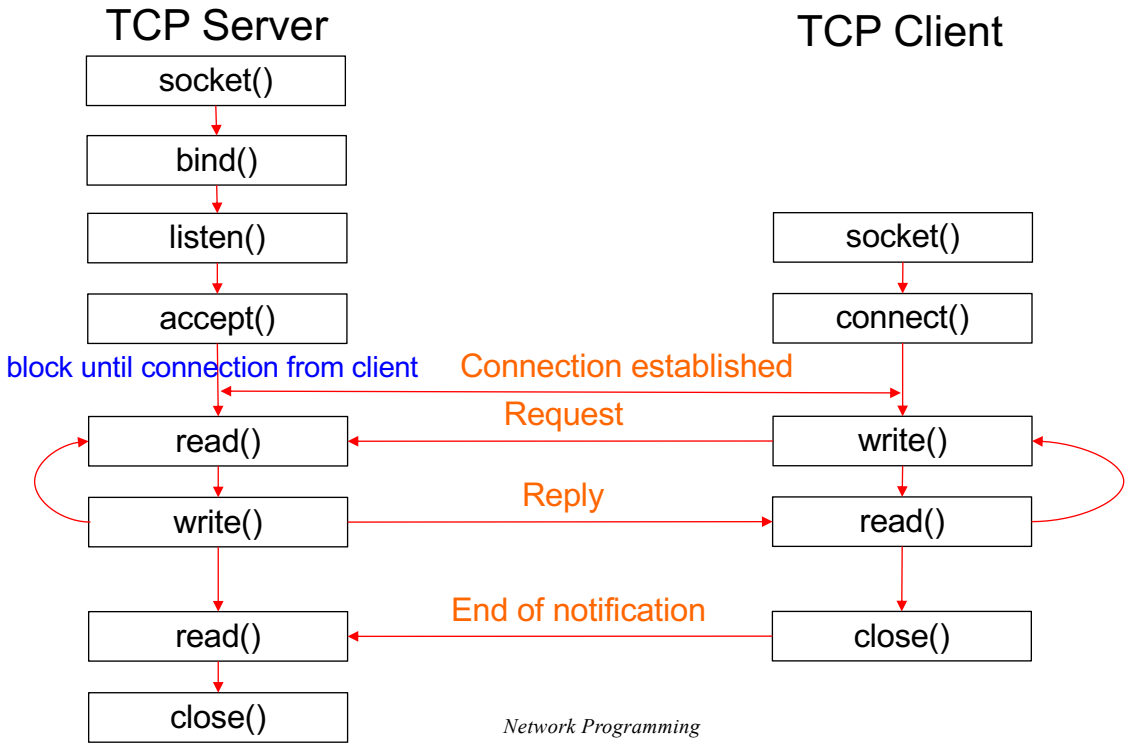


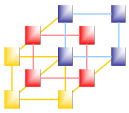
Socket programming

- LISTEN
- CONNECT
- SEND
- RECEIVE
- DISCONNECT

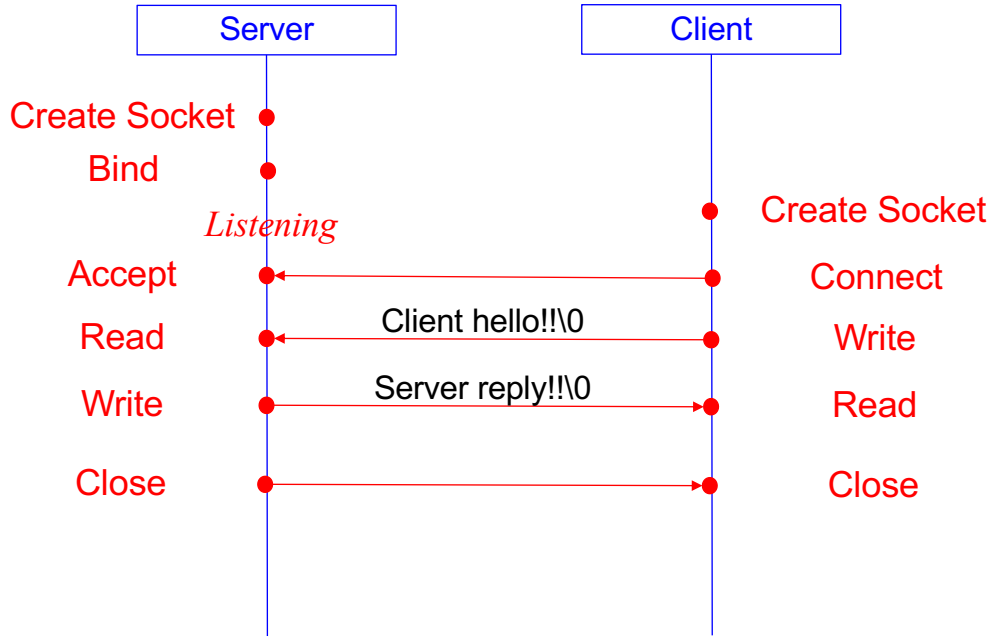


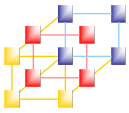
TCP client / server model



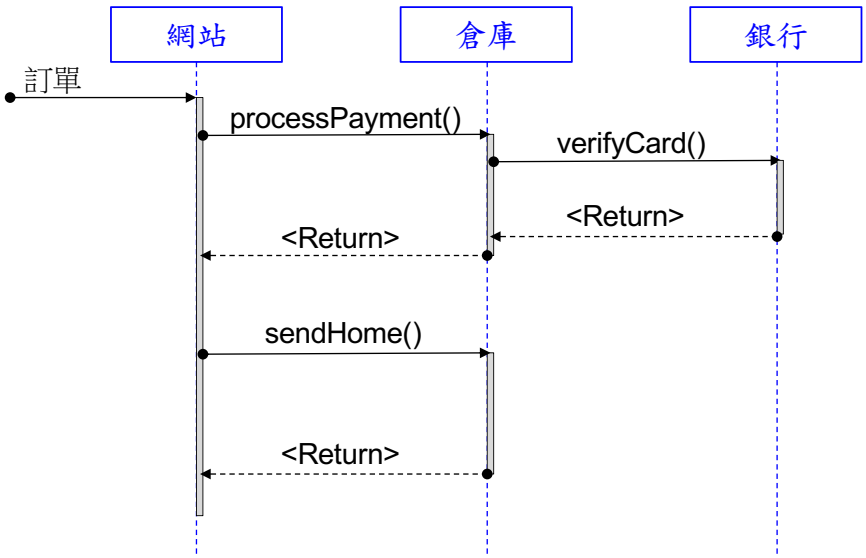


Sequence Diagram (序列圖、循序圖)

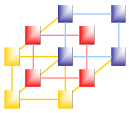




循序圖範例

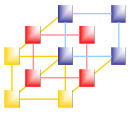



```
3-TCPServer.py
3-TCPClient.py
4-TCPServer.py
4-TCPClient.py
```



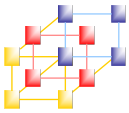
Socket module in Python (1/4)

- S = socket.socket(socket_family, socket_type, protocol=0)
  - socket_family
    - AF_UNIX – 多用於同一機器間不同 processes 的通訊
    - AF_INET – 用於 IPv4 網路
  - socket_type
    - SOCK_STREAM – TCP
    - SOCK_DGRAM – UDP
  - protocol: This is usually left out, defaulting to 0
- socket.setsockopt(SOL_SOCKET, SO_REUSEADDR , 1)
  - SOL_SOCKET – 正在使用的socket選項
  - SO_REUSEADDR – 當socket關閉後，本地端用於該socket的埠號立刻就可以被重用。通常來說，只有經過系統定義一段時間後，才能被重用
  - 1: SO_REUSEADDR = TRUE



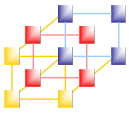
Socket module in Python (2/4)

- bind( )
  - bind the socket to an address
  - Ex: socket.bind((‘192.168.1.1’, 80)), ‘’ is any interface
- listen(backlog)
  - This method listens for the connection made to the socket
  - The backlog is the maximum number of queued connections that must be listened before rejecting the connection
- accept( )
  - This method is used to accept a connection
  - The return value is a pair (conn, address)
    - conn is a new socket object which can be used to send and receive data on that connection
    - address is the address bound to the socket on the other end of the connection



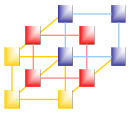
Socket module in Python (3/4)

- connect( )
  - To connect to a remote socket at an address. An address format(host, port) pair is used for AF_INET address family
- recv(bufsize[, flags])
  - Receive data from the socket
  - The return value is a bytes object representing the data received
  - The maximum amount of data to be received at once is specified by bufsize.
  - optional argument flags; it defaults to zero



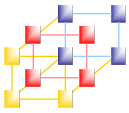
Socket module in Python (4/4)

- socket.send(bytes[, flags])
  - Send data to the socket
  - The optional flags argument has the same meaning as for recv() above
  - Returns the number of bytes sent
  - e.g. socket.send(b‘Hello, World!’)
- close()
  - close the socket connection



Binary data structure (1/4)

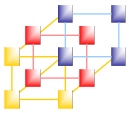
- Structs support packing data into strings, and unpacking data from strings using format specifiers made up of characters representing the type of the data and optional count and endianness indicators.
  - pack(fmt, v1, v2, ...)
    - 將變數轉成對應的格式
  - unpack(fmt, string)
    - 將對應的格式轉回數據



Binary data structure (2/4)

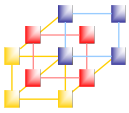
FORMAT	[C TYPE]	[PYTHON TYPE]	[STANDARD SIZE]	[NOTES]
x	pad byte	no value		
c	char	string of length	1	(1)
b	signed char	integer	1	(3)
B	unsigned char	integer	1	(3)
?	_Bool	bool	1	(1)
h	short	integer	2	(3)
H	unsigned short	integer	2	(3)
i	int	integer	4	(3)
I	unsigned int	integer	4	(3)
l	long	integer	4	(3)
L	unsigned long	integer	4	(3)
q	long long	integer	8	(2), (3)
Q	unsigned long	long integer	8	(2), (3)
f	float	float	4	(4)
d	double	float	8	(4)
s	char[]	string		
p	char[]	string		
P	void *	integer		(5), (3)

(1) q 和 Q 只在機器支持 64 位操作時有意思
(2) 每個格式前可以有一個數字，表示個數
(3) s格式表示一定長度的字符串，4s表示長度為4的字符串，但是p表示的是pascal字符串
(4) P用來轉換一個指標，其長度和機器字長相關
(5) 最後一個可以用來表示指標類型的，佔4個字節



Binary data structure (3/4)

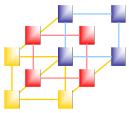
```
>>> import struct
>>> import binascii
>>>
>>> packed_data1 = struct.pack('I 2s f', 1, 'ab'.encode('utf-8'), 2.7)
>>> values = (1, 'ab'.encode('utf-8'), 2.7)
>>> s = struct.Struct('I 2s f')
>>> packed_data2 = s.pack(*values)
>>>
>>> print('Original values:', values)
Original values: (1, b'ab', 2.7)
>>> print('Format string  :', s.format)
Format string  : I 2s f
>>> print('Uses           :', s.size, 'bytes')
Uses           : 12 bytes
>>> print('Packed Value   :', binascii.hexlify(packed_data1))
Packed Value   : b'0100000061620000cdcc2c40'
```



Binary data structure (4/4)

```
>>> import struct
>>> import binascii
>>>
>>> packed_data =
binascii.unhexlify(b'0100000061620000cdcc2c40')
>>>
>>> s = struct.Struct('I 2s f')
>>> unpacked_data = s.unpack(packed_data)
>>> print('Unpacked Values:', unpacked_data)
Unpacked Values: (1, b'ab', 2.700000047683716)
```


5-TCPServer.py
5-TCPClient.py



Endianness

```
import struct
import binascii

values = (1, 'ab'.encode('utf-8'), 2.7)
print('Original values:', values)
endianness = [
    ('@', 'native, native'),
    ('=', 'native, standard'),
    ('<', 'little-endian'),
    ('>', 'big-endian'),
    ('!', 'network'),
]
for code, name in endianness:
    s = struct.Struct(code + ' I 2s f')
    packed_data = s.pack(*values)
    print()
    print('Format string  :', s.format, 'for', name)
    print('Uses           :', s.size, 'bytes')
    print('Packed Value   :', binascii.hexlify(packed_data))
    print('Unpacked Value :', s.unpack(packed_data))
```

Code	Meaning
@	Native order
=	Native standard
<	little-endian
>	big-endian
!	Network order