

## Lab 2 - Socket programming

### Blocking-IO vs. Non-Blocking IO

ในกรณีที่ไม่ต้องการให้รอกระบวนการใดกระบวนการหนึ่งให้เสร็จก่อน สามารถใช้วิธีการ callback ได้ เช่นการอ่านไฟล์ text.txt โดยใช้ fs library จากตัวอย่างต่อไปนี้

#### 2.1 สร้าง ไฟล์ text.txt มีข้อความดังนี้

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser.

#### Blocking IO

สร้าง block.js

```
fs = require("fs");
data = fs.readFileSync('text.txt');
console.log(data.toString());
console.log("Finished");
```

#### Output

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser.

Finished

#### Non-blocking IO

```
fs = require("fs");
fs.readFile('text.txt', (err, data) => {
  if (err)
    return console.error(err);
  console.log(data.toString());
});
console.log("Finished");
```

#### Output

Finished

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser.

ตัวอย่างการ รับ input ที่เป็น non-blockingIO

## 2.2 การรับค่า 2 ค่า โดยใช้ callback ซ้อนกัน

โปรแกรม input1.js

```
console.log('a: ')
let stdin = process.openStdin()
stdin.addListener("data", (a) => {
  // console.log(a.toString().trim())
  console.log('b: ')
  stdin.addListener("data", (b) => {
    console.log(a*b)
    stdin.destroy()
  })
})
})
```

จงบอกความแตกต่างระหว่าง BlockIO กับ Non-BlockingIO

ตอบ

## 2.3 เขียนใหม่ แบบ ประกาศฟังก์ชันก่อน ดังนี้

โปรแกรม input2.js (ปรับปรุงจาก input1.js)

```
let stdin = process.openStdin()
let a = 1;

const main = () => {
  console.log('a: ')
  stdin.addListener("data", getA)
}

const getA = (num) => {
  a = num.toString().trim()
  console.log('a = ' + a)
  console.log('b: ')
  stdin.addListener("data", getB )
}

const getB = (b) => {
  console.log('a = ' + a.toString().trim())
  console.log('b = ' + b.toString().trim())
  console.log('a*b = ' + a*b)
  stdin.destroy()
}

main()
```

จากโปรแกรมข้างบน ได้ผลการทำงานคือ

```
a:
2
a = 2
b:
3
a = 3
b:
a = 3
b = 3
a*b = 9
```

จงอธิบายว่าเหตุใด  $a*b$  จึงไม่ได้  $2*3$ ?

ตอบ

2.3 การใช้ loop เดียว และ ใช้เงื่อนไขมาช่วย ในการรับค่า 2 ตัวแปร  
input3.js

```
console.log('a: ')
let stdin = process.openStdin()
let a, b, i=1
stdin.addListener("data", (num) => {
  // console.log(High.toString().trim())
  console.log('b: ')
  if ( i === 1)
    a = num
  else if ( i === 2 ){
    b = num
    console.log("a*b: " + a*b)
    stdin.destroy()
  }
  i++
})
```

## Event

ใช้ event ในการรองรับการเรียกผ่าน emit()

```
# Event: https://nodejs.org/api/events.html#events\_events
const EventEmitter = require('events');
class MyEmitter extends EventEmitter {}
const myEmitter = new MyEmitter();
myEmitter.on('event', (a,b) => {
  console.log(a,b, 'an event occurred!')
});
myEmitter.emit('event', 'a', 'b')
```

## Socket programming

### ข้อกำหนด

- ให้ ทดลอง TCP/UDP Server กับ TCP/UDP Client เพื่อทดลองส่งข้อมูลระหว่างกัน
  - สังเกตค่า HOST และ PORT ที่ใช้ในการเชื่อมต่อ
- หากเป็นการส่งข้อมูลระหว่างเครื่อง จำเป็นจะต้องใช้สิทธิ์ admin ในการปิด firewall ของ windows ก่อน

## TCP Socket

### TCP Server1

```
var net = require('net');

var HOST = '127.0.0.1';
var PORT = 6969;
net.createServer(function(sock) {
    console.log('CONNECTED: ' + sock.remoteAddress + ':' + sock.remotePort);
    sock.on('data', function(data) {
        console.log('DATA ' + sock.remoteAddress + ': ' + data);
        sock.write('You said "' + data + '"');
    });

    sock.on('close', function(data) {
        console.log('CLOSED: ' + sock.remoteAddress + ' ' + sock.remotePort);
    });
}).listen(PORT, HOST);

console.log('Server listening on ' + HOST + ':' + PORT);
```

### TCP Server2

```
var net = require('net');

var HOST = '127.0.0.1';
var PORT = 6969;
var server = net.createServer();
server.listen(PORT, HOST);
server.on('connection', function(sock) {
    console.log('CONNECTED: ' + sock.remoteAddress + ':' + sock.remotePort);
    sock.on('data', function(data) {
        console.log('DATA ' + sock.remoteAddress + ': ' + data);
        sock.write('You said "' + data + '"');
    });

    sock.on('close', function(data) {
        console.log('CLOSED: ' + sock.remoteAddress + ' ' + sock.remotePort);
    });
});
```

## TCP Client

```
var net = require('net');
var HOST = '127.0.0.1';
var PORT = 6969;

var client = new net.Socket();
client.connect(PORT, HOST, function() {
    console.log('CONNECTED TO: ' + HOST + ':' + PORT);
    client.write('I am Chuck Norris!');
});

client.on('data', function(data) {
    console.log('DATA: ' + data);
    client.destroy();
});

// Add a 'close' event handler for the client socket
client.on('close', function() {
    console.log('Connection closed');
});
```

## UDP Socket

### UDP Server

```
var dgram = require("dgram");
var server = dgram.createSocket("udp4");
server.on("message", function (msg, rinfo) {
    console.log("server got: " + msg + " from " +
        rinfo.address + ":" + rinfo.port);
});
server.bind(41234); // server listening 0.0.0.0:41234
```

### UDP Client

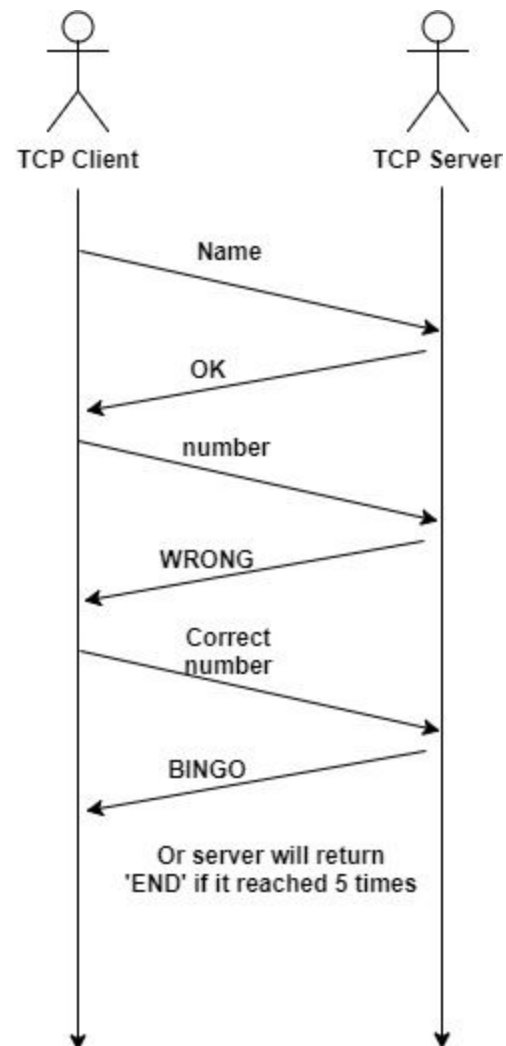
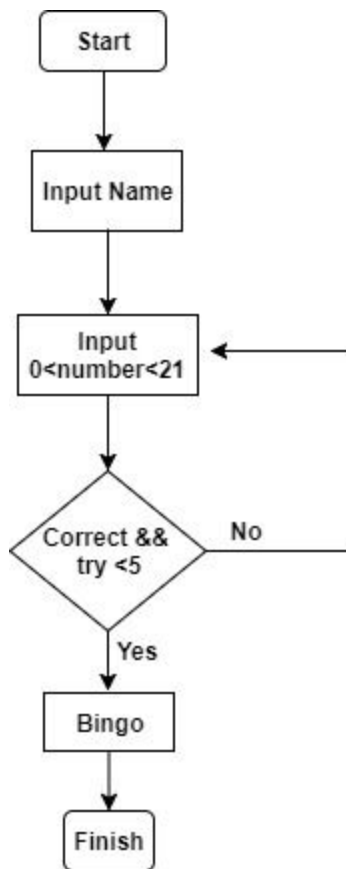
```
var dgram = require('dgram');
var message = new Buffer.from("My message", 'utf8');
var client = dgram.createSocket("udp4");
client.send(message, 0, message.length, 41234, "localhost",
    function(err, bytes) {
        client.close();
    }
);
```

จงอธิบายความแตกต่างในการส่งข้อมูล ระหว่าง TCP/UDP พร้อมข้อดี ข้อเสีย

ตอบ

ทดลองเขียนเกมทายตัวเลขฝั่ง Client โดยมีการทำงานดังรูป

ฝั่ง server รอรับการเชื่อมต่อ โดย client จะต้องส่งชื่อมา server ตอบ OK กลับไป และ client ส่งตัวเลข เพื่อทายตัวเลขที่ฝั่ง server กำหนดไว้ (ตัวเลขมีค่าอยู่ระหว่าง 0-21 เพื่อให้ไม่ทายยากเกินไป) ถ้าตอบผิด server ตอบกลับว่า WRONG ถ้าตอบถูก server ตอบ BINGO แต่ถ้าตอบผิดครบ 5 ครั้ง ให้ตัดการเชื่อมต่อ



การสุ่มตัวเลข สามารถใช้คำสั่ง `let answer=Math.floor(Math.random() * 21);` และใช้ `parseInt()` ในการแปลงข้อความเป็นตัวเลขจำนวนเต็ม

ตอบ

**Homework:** จงเขียนการทำงานฝั่ง Server