DAM2 – M9

UF3: Sòcols i serveis

Pràctica 1

Format i data de lliurament

S'ha de lliurar la solució en un arxiu .ZIP que inclogui el document principal en format PDF i els arxius de codi font.

L'arxiu amb la solució tindrà el format: NomCognom_M9_UF3_P1.zip

La data límit del lliurament és el dia 27/03/2019

Les pràctiques lliurades fora del termini tindran una penalització del 10% de la nota per cada dia de retard.

Criteris de valoració

Raoneu la resposta en tots els exercicis.

Si es detecta una còpia de la pràctica aquesta serà avaluada amb un 0.

Les respostes sense justificació, que siguin una còpia d'una font d'informació i/o que no continguin les referències utilitzades, no rebran cap puntuació.

Es valorarà la correcció ortogràfica i la correcte presentació.

Indicacions

Tots els codis que se us lliuren (independentment del protocol usat) tenen un comportament equivalent (model client/servidor):

- El client rep com a únic paràmetre el nom de la màquina on es troba el servidor (localhost si ho feu des de la mateixa màquina). El servidor no rep cap paràmetre.

- El client es connecta al servidor i li va enviant missatges de text. Quan el client envia la cadena fi es tanca la connexió.

- El servidor accepta la connexió d'un client, i va mostrant per pantalla els missatges que rep d'ell.

Exercicis

Exercici 1

1. Comenteu el codi proporcionat, de manera que mostreu que enteneu el seu funcionament, tant en TCP com en UDP. Poseu els comentaris dins el mateix fitxer de codi.

2. Contesteu les següents preguntes:

TCP

a) Quina crida serveix per crear el *socket*? Quins paràmetres rep i què significa cadascun d'ells? Existeix alguna diferència entre les crides que realitzen el client i el servidor?

Client: Al client fem servir *Socket(address , port).* On «address» es l'adreça del servidor i port es per on sortira la connexió del client.

Servidor: Al servidor fem servir *ServerSocket(port).* On «port» es el port per on ha d'escoltar el servidor per fer connexió amb el client

b) Quin crida fa el servidor per admetre la connexió d'un nou client? Què retorna aquesta crida? Què té de particular i per a què és necessari aquest comportament?

Primer creem una instacia de tipus socket. I amb *serverSocket.accept()* permetrem la connexió i podrem crear l'instancia comentada abans.

c) Tal com està programat el servidor, quants clients concurrents poden estar connectats al mateix moment?

Només soporta/aceptar un sol client, en cas d'executar un segon no fara res.

d) En executar els programes, comproveu amb netstat que s'han creat els sockets corresponents i adjunteu-ne una captura.

```
C:\Users\34682>netstat -an | find ":6001"
  TCP    0.0.0.0:6001           0.0.0.0:0              LISTENING
  TCP    127.0.0.1:6001         127.0.0.1:54092        ESTABLISHED
  TCP    127.0.0.1:54092        127.0.0.1:6001         ESTABLISHED
  TCP    [::]:6001              [::]:0                 LISTENING
```

UDP

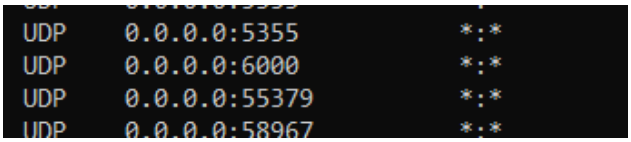
a) Quina és la unitat d'intercanvi de dades entre el client i el servidor? Quantes dades pot transportar, a partir del que s'observa en el codi, cada paquet de dades?
Es crea una instancia de *DatagramPacket(message_bytes,message.lenght(),adress,port).*
Address i port ja ho he explicat abans. «message_bytes» es la quantitat de bytes que pot transportar tant d'entrada o sortida.
Oh crearem de la següent manera: *bytes[] message_bytes = new byte[256]*

b) Atès que en UDP no s'estableix connexió entre el client i el servidor, on defineix el client l'adreça i el port del servidor al qual es volen enviar les dades?
L'adreça i el port és el tercer i quart parametre de *DatagramPacket.*

c) Quines crides usa el client per enviar dades? Quines crides usa el servidor per rebre dades a través dels *sockets* UDP? Quins paràmetres rep cada crida?
El client enviar el paquet amb *packet.send(),* previament amb el *DatagramPacket* creat.
El servidor fa servir amb el *DatagramPacket c*reat, *socket.recieve(paquet)*

d) Com pot saber el servidor qui envia un paquet?
Amb *packet.geAddress()* el servidor rep l'ip del client

e) En executar els programes, comproveu amb netstat que s'han creat els *sockets* corresponents i adjunteu-ne una captura.



Exercici 2

Modifiqueu el codi que se us ha donat de partida per tal que funcioni com a servidor de comandes d'una cadena de menjar ràpid, de la següent manera:

– El client rebrà com a paràmetres del programa el nom de la màquina on es troba el servidor (per exemple, localhost si executeu ambdós programes des de la mateixa màquina). El servidor no rebrà cap paràmetre.

– El client es connectarà al servidor i acte seguit li enviarà una salutació formada per la cadena "HELLO".

– El servidor l'analitzarà i mostrarà per pantalla el missatge "HELLO".

– El servidor retornarà la salutació, contestant també amb la cadena "HELLO".

– El client imprimirà per pantalla la cadena rebuda.

– A partir d'aquí, l'usuari introduirà per teclat un codi de producte i el client l'enviarà al servidor.

– El servidor afegirà al darrera de la cadena rebuda un espai en blanc i el preu del producte de la comanda, ho imprimirà per pantalla, i després ho retornarà al client.

– El client ho imprimirà per pantalla i tornarà a repetir el mateix procés (enviant un missatge al servidor, esperant que li contesti el servidor), tantes vegades com desitgi.

– Els preus dels productes estaran prefixats en el codi tal i com s'especifica en l'apartat *Notes* d'aquest enunciat.

– Quan l'usuari vulgui finalitzar el programa, escriurà "fi" per teclat. El client li ho indicarà al servidor, enviant-li un missatge de fi. El servidor li respondrà amb el mateix missatge i el client finalitzarà l'execució del programa després de mostrar-lo per pantalla.

– El servidor, en rebre fi, tancarà la connexió i es tornarà a posar a escoltar noves peticions de nous clients. Només cal que poseu la major part del codi del servidor dins un bucle i que després de tancar la connexió torni a l'inici del bucle.

– S'han de contemplar els casos en que tant lectures com escriptures sobre sockets retornen error, mostrant un missatge indicatiu per pantalla i, a continuació, finalitzant l'execució del programa.

– El client no ha de comprovar si les comandes introduïdes són o no correctes o existents.

– El servidor sí que ha de comprovar si la comanda correspon a un producte conegut, o a la comanda de salutació HELLO. En cas que el servidor rebi quelcom desconegut, retornarà sempre la cadena "ERROR".

Notes:

1-Els productes a tenir en compte són (i per tant comandes que rebrà el servidor):

- REF Refresc
- PAT Patates fregides
- CRO Croissant
- BOC Entrepà
- ENS Ensaladilla
- CAF Cafè

2-També existeix la comanda de salutació HELLO.

3-En analitzar la cadena rebuda, si voleu, només cal que analitzeu el primer caràcter per saber de quin producte es tracta (en el cas de CRO i CAF fixeu-se que heu d'agafar els dos primers). En cas que el servidor rebi quelcom desconegut, retornarà sempre la cadena "ERROR".

4-El preu dels productes anirà prefixat en el codi del servidor amb els següents valors en euros:

- REF 2
- PAT 3
- CRO 1
- BOC 2
- ENS 4
- CAF 1

5-Cal distingir entre majúscules i minúscules, és a dir, el vostre resultat ha de ser exactament igual que el d'aquest enunciat. Per exemple, si els productes estan en minúscula retornarem error.

Exemple:

A continuació es mostra una execució exemple del funcionament entre client i servidor. En blau teniu el que es mostra per pantalla, en verd el que s'introdueix pel teclat i en vermell el que s'envien entre ells.

Client	Servidor
HELLO ->	
	HELLO
	<- *HELLO*
HELLO	
REF	
REF ->	
	REF 2
	<- *REF 2*
REF 2	
PAT	
PAT ->	
	PAT 3
	<- *PAT 3*
PAT 3	
BOC	
BOC ->	
	BOC 2
	<- *BOC 2*
BOC 2	
boc	
boc ->	
	boc ERROR
	<- boc ERROR
boc ERROR	
XXX	
XXX ->	
	XXX ERROR
	<- XXX ERROR
XXX ERROR	
FI	
FI ->	
	FI
	<- *FI*
FI	

Exercici 3

Repetiu l'exercici 2 , però ara modificant la versió UDP del codi.