EDA095 FED

Tentamensskrivning — Nätverksprogrammering 2004-05-25, kl 8-13

DEL 2 - Praktiska programmeringsuppgifter

Anvisningar

Tillåtna hjälpmedel för denna del av tentamen:

- Java snabbreferens.
- Kursboken: Java Network Programming av Elliotte Rusty Harold eller An Introduktion to Network Programming with Java av Jan Graba.
- Valfri lärobok i Java.
- Utdelat komplement till läroboken ("Kompletterande kursmaterial").
- Utskrift av OH-bilder från föreläsningarna - ej separata exempelprogram.

Denna tentamen i kursen Nätverksprogrammering består av två delar - en del som innehåller frågor av teoretisk/principiell/utredande karaktär och en del som innehåller praktiska programmeringsuppgifter. Detta är del 2. Den ska du ha erhållit tillsammans med ett färgat tentamensomslag när du lämnade in din lösning på del 1 av tentamen.

För godkänt betyg på tentamen krävs sammanlagt minst 20 poäng på tentamen, varav minst 8 poäng på vardera deltentamen. För högre betyg krävs naturligtvis mer, så gör så många uppgifter du kan.

Senast 2004-05-28 anslås på institutionens anslagstavla vilka som deltagit i tentamen men som, enligt institutionens noteringar, ännu inte redovisat laborationerna eller projektet. Deras skrivning rättas inte (och kan på sikt komma att annulleras) såvida inte rättelse skett eller dispens erhållits från ansvarig lärare. Rättelse skall göras senast 2004-06-11.

Uppgifterna nedan kan innehålla en hel del beskrivningar på vad som ska hända i olika specialfall. Om du inte hinner med eller vet hur du ska hantera alla dess så är det bättre att du ignorerar dem och åstadkommer en lösning som fungerar i övrigt än att du inte löser uppgiften alls. En lösning kan ge mycket poäng även om alla specialfall inte är täckta.

(vänd)

1. TicTacToe, eller “tre-i-rad”.

Det välkända spelet TicTacToe, eller “tre-i-rad”, spelas på en spelplan bestående av 3 gånger 3 rutor. Spelarna turas om att sätta sitt tecken, ett kryss eller en ring, i en ledig ruta. Den vinner som först får tre rutor i rad med sitt eget tecken. Om ingen har tre rutor i rad när alla rutor är ifyllda blir det oavgjort. Din uppgift blir att skriva en server som man kan koppla upp sig mot via Internet och spela mot andra användare på nätverket. Servern ska implementeras enligt följande:

1. Spelare kopplar upp sig mot servern med hjälp av programmet Telnet. Användargränssnittet blir alltså helt textbaserat. Det medför också att inget speciellt klientprogram behöver utvecklas.
2. När en spelare kopplar upp sig ska han ställas i vänteläge tills en motspelare ansluter. Spelare paras ihop två och två i den ordning de ansluter till servern. När två spelare är ihopparade börjar spelet.
3. Flera spel ska kunna vara igång samtidigt och nya spelare ska kunna ansluta när som helst.
4. Servern ska vara så robust att den överlever att en förbindelse kopplas ned oväntat. Naturligtvis måste ett pågående spel avbrytas om förbindelsen till den ena spelaren bryts, och motspelaren ska informeras om detta, men andra pågående spel ska ej påverkas och nya spelare ska kunna fortsätta att ansluta och starta nya spel.
5. Ett spel ska gå till så att servern omväxlande ber spelarna att skriva in var de vill placera sitt tecken och presenterar löpande hur spelplanen ser ut. Om en spelare får tre märken i rad ska spelomgången avbrytas och spelarna informeras om utgången. Detsamma gäller om spelplanen blir full utan att någon har vunnit. När en spelomgång är slut ska spelplanen tömmas och en ny spelomgång börja. Spelarna ska dock ha något sätt att avsluta spelet och koppla ned förbindelsen till servern när de inte vill spela längre.
6. Välj själv lämpliga ledtexter till spelarna och hur de ska mata in var de vill placera sina tecken. Tänk dock på att Telnet skickar två tecken när man trycker på radslutstangenten - vagnretur (teckenkod 13) och radframmatning (teckenkod 10). Likaledes bör servern skicka dessa två tecken när man vill byta rad på utmatningen.
7. För att underlätta hanteringen av själva spelplanen och för att avgöra huruvida någon spelare har fått tre tecken i rad kan du använda klassen TicTacToeBoard som bifogas på nästa sida. Klassen kan förutom att hålla reda på var alla tecken finns och om någon har fått tre tecken i rad också returnera en sträng som representerar spelplanens aktuella utseende och som är lämplig att skicka direkt till en Telnetklient. Vill du ändra eller lägga till något i klassen går det bra.

(15p)

```
/**
 * A class representing a TicTacToe game board.
 * The squares are identified by their row and column coordinates, which
 * both ranges from 0 to 2.
 */
public class TicTacToeBoard {
     private int[][] b;

     /**
      * Create a new game board
      */
     public TicTacToeBoard() {
          b = new int[3][3];
     }

     /**
      * Clear the contents of the board
      */
     public void clear() {
          for(int r=0;r<3;r++) {
               for(int c=0;c<3;c++) {
                    b[r][c] = 0;
               }
          }
     }

     /**
      * Test if the square at row r and column c is free
      */
     public boolean isFree(int r,int c) {
          return b[r][c]==0;
     }

     /**
      * Place an X at the indicated coordinates
      */
     public void setX(int r,int c) {
          b[r][c] = 1;
     }

     /**
      * Place an O at the indicated coordinates
      */
     public void setO(int r,int c) {
          b[r][c] = 2;
     }

     /**
      * Test if the game has been won
      */
     public boolean gameWon() {
          if (b[0][1]==b[0][0] && b[0][2]==b[0][0]) return true;
          if (b[1][1]==b[1][0] && b[1][2]==b[1][0]) return true;
          if (b[2][1]==b[2][0] && b[2][2]==b[2][0]) return true;
          if (b[1][0]==b[0][0] && b[2][0]==b[0][0]) return true;
          if (b[1][1]==b[0][1] && b[2][1]==b[0][1]) return true;
          if (b[1][2]==b[0][2] && b[2][2]==b[0][2]) return true;
          if (b[1][1]==b[0][0] && b[2][2]==b[0][0]) return true;
          if (b[1][1]==b[0][2] && b[2][0]==b[0][2]) return true;
          return false;
     }

     /**
      * Creates a string that can be written into a Telnet terminal in
      * order to provide a "graphical" view of the game board.
      */
     public String toString() {
          StringBuffer s = new StringBuffer();
          for(int r=0;r<3;r++) {
               for(int c=0;c<3;c++) {
                    if (b[r][c]==0) {
                         s.append('.');
                    } else {
                         if (b[r][c]==1) {
                              s.append('X');
                         } else {
                              s.append('O');
                         }
                    }
               }
               s.append((char) 13);
               s.append((char) 10);
          }
          return s.toString();
     }
}
```

2. Internetradio via multicast

När kursansvarig för kursen Nätverksprogrammering började doktorera 1991 noterade han att någon på institutionen vid något tillfälle hade lött ihop en kabel som gjorde att man kunde koppla valfri audioutrustning som var försedd med ett hörlursuttag till ljudingången på de Sun 3/60-maskiner institutionen då hade. På hyllan bredvid en av dessa maskiner stod en gammal transistorradio.

Lite efterforskning visade dessutom att det var ganska lätt att använda sig av ljudingången (och ljudutgången) mjukvarumässigt. Den inbyggda hårdvaran omvandlade den analoga inkommande signalen till en ström av bytes med hastigheten 8192 bytes/sekund (8-bitars sampel med en samplingsfrekvens på ca 8 kHz). För att spela upp denna digitaliserade ljudsignal i den inbyggda högtalaren i datorn behövde man bara i sin tur skicka denna insamplade dataström tillbaka till ljudhårdvaran som omvandlade den till en analog signal som spelades upp i högtalaren.

Idén var given: Varför inte ansluta transistorradion till ljudingången på en maskin, skriva ett serverprogram (som körde på denna maskin) som läste in den digitaliserade ljudsignalen och skickade ut den via nätverket till alla på institutionen som ville lyssna på radio? Varje användare startade ett klientprogram som kopplade upp sig mot servern och tog emot ljuddata som sedan skickades till högtalaren.

Uppgiften består av att implementera ett klient/server-system i Java enligt beskrivningen ovan baserat runt *multicast*. Till din hjälp har du följande färdigskrivna klass som kan användas för att läsa in digitaliserat ljud till en ström av bytes eller omvänt skicka den digitala strömmen till högtalaren:

```
specification SoundIO {
    /** Skapar ett objekt med vars hjälp man kan läsa in en digitaliserad
        signal från ljudingången eller skicka motsvarande signal till
        högtalaren */
    public SoundIO();

    /** Läser in så många bytes som har hunnit samplas in sedan operationen
        anropades senast, och lagrar dessa i vektorn "data". Om fler bytes
        finns tillgängliga än vad som ryms i "data" lagras endast så många
        bytes. Resten sparas till nästa anrop av "read". Det antal bytes som
        verkligen lagrades i "data" returneras som resultat av anropet.
        Skulle inga data finnas tillgängliga blockerar anropet av "read" tills
        minst 128 bytes data finns tillgängliga. */
    public int read(byte [] data);

    /** Skickar de "n" stycken första bytes i vektorn "data" till högtalaren
        för omvandling till en analog signal och uppspelning. */
    public void write(byte [] data,int n);
}
```

Av effektivitetsskäl kan det vara lämpligt att försöka läsa och skriva 1024 bytes eller mer varje gång man anropar `read`/`write` ovan. Storleken på den vektor `data` som används som inparameter bör alltså vara minst 1024 bytes.

För att begränsa uppgiften något har vi redan skrivit klientdelen av programmet, vilken återfinns på nästa sida. Din uppgift blir alltså att skriva ett serverprogram som fungerar tillsammans med detta klientprogram. Låt servern kontinuerligt skicka ut ljuddata på nätet, oavsett om det finns någon klient som lyssnar eller inte.

(5p)

```
import java.net.*;
import java.io.*;

public class NetworkRadioClient{

    public static void main(String args[]) {
        SoundIO io = new SoundIO();
        try {
            MulticastSocket ms = new MulticastSocket(40099);
            InetAddress ia = InetAddress.getByName("all-systems.mcast.net");
            ms.joinGroup(ia);
            byte[] buf = new byte[65536];
            DatagramPacket dp = new DatagramPacket(buf,buf.length);
            while(true) {
                ms.receive(dp);
                io.write(dp.getData(),dp.getLength());
                dp.setLength(dp.getData().length);
            }
        } catch(IOException e) {
            System.out.println("Exception:"+e);
        }
    }
}
```
