LUNDS TEKNISKA HÖGSKOLA Institutionen för datavetenskap

Tentamen
Nätverksprogrammering
Del 1

2010–06–01, 08.00–13.00

Tillåtna hjälpmedel för denna del av tentamen: *inga*. Kurslitteratur och andra hjälpmedel för del 2 av tentamen skall förvaras på golvet bredvid bordet eller vid salens vägg.

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 1. När du löst uppgifterna i denna del av tentamen lämnar du in din lösning i det vita tentamensomslaget varvid du erhåller del 2 av tentamen tillsammans med ett nytt, färgat, tentamensomslag som skall användas vid inlämning av din lösning på del 2 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 mer, så gör så många uppgifter du kan.

---

1. Ange för varje Javaklass nedan huruvida man normalt använder den i samband med att man kommunicerar över nätverket med hjälp av TCP, UDP, eller om den kan vara relevant för både TCP och UDP.

   a) InetAddress

   b) Socket

   c) DatagramPacket

   d) ServerSocket

   (2p)

2. Det finns tre basprotokoll som Internet Engineering Task Force och Internet Society har definierat för att hantera elementär mediaströmning: real-time transport protocol (RTP), real-time control protocol (RTCP) och real-time streaming protocol (RTSP).

   Förklara kortfattat vad syftet med vart och ett av protokollen är, vilken funktionalitet de erbjuder samt hur de hänger ihop med varandra.

   (3p)

3. Vad är huvudskillnaden mellan XHTML och HTML?

   (1p)

4. I samband med tekniken RMI talar man ibland om *marshalling*. Vad innebär detta?

   (1p)

5. I HTTP-protokollet förekommer bland annat de tre kommandona GET, POST och HEAD. Förklara vad de tre olika kommandona gör och hur de skiljer sig från varandra.

   (2p)

---

6. Vilka av följande sex påståenden är korrekta?
   a) För att upprätta en TCP-förbindelse till ett program som kör på en annan dator i nätverket räcker det med att vi känner till den andra datorns IP-nummer samt vilket portnummer det andra programmet lyssnar på.
   b) Portnummer 1-1023 är normalt reserverade för UDP-trafik.
   c) En socket representerar i TCP-sammanhang en enkelriktad förbindelse över nätverket.
   d) Om ett program lyssnar efter UDP-paket på ett visst portnummer kan ett annat program på samma dator starta en TCP-server som accepterar TCP-uppkopplingar på samma portnummer.
   e) När vi skickar data via en TCP-förbindelse delas vår data automatiskt upp i mindre bitar som skickas i form av UDP-paket. När UDP-paketen tas emot sätts datan i UDP-paketen automatiskt ihop igen till en kontinuerlig dataström.
   f) Multicast kan sägas vara en variant av UDP där varje meddelande skickas till flera mottagare samtidigt istället för bara till en enda (unicast).

   (3p)

7. En binär semafor är en mekanism som ibland används för att åstadkomma ömsesidig uteslutning i ett program. Den kan beskrivas som ett objekt som har två metoder: `take()` och `give()`. När ömsesidig uteslutning önskas anropar en tråd metoden `take()` och när ömsesidig uteslutning inte längre är nödvändig anropar tråden `give()`. Om någon annan tråd försöker anropa `take()` under denna tid kommer den att blockera inuti anropet av `take()` tills dess att `give()` anropats av den första tråden.

   Programmeraren Multitråds-Martin har skrivit följande klass som implementerar en binär semafor:

```
public class BinarySemaphore {
    boolean locked = false;

    public void take() {
        while(locked) {
            // Do nothing - just wait for locked to be false
        }
        locked = true;
    }

    public void give() {
        locked = false;
    }
}
```

   a) Vad brukar man kalla den i take() ovan använda tekniken för att vänta på ett villkor?

   (1p)

   b) Denna metod har en stor nackdel som gör att den oftast bör undvikas. Vilken?

   (1p)

   c) Förutom problemet som söktes i uppgift b) lider lösning dessutom av problem med så kallad kapplöpning (race condition). Om vi har en väldig otur med hur systemet väljer att schemalägga de olika trådarna finns det en risk för att flera trådar släpps förbi anropet av `take()` samtidigt.

   Skriv om klassen `BinarySemaphore` ovan så att nackdelen i uppgift b) elimineras helt tillsammans med risken för kapplöpning. Använd Javas monitorbegrepp för att åstadkomma detta. Klassen ska ur de anropande trådarnas synvinkel fungera likadant som tidigare.

   (3p)

8. I Javas klass `InputStream` finns en operation `read` med följande signatur (motsvarande operation finns även i C/C++):

```
public int read(byte[] input,int offset, int length) throws IOException;
```

Antag att vi vill använda operationen för att läsa in 100 byte från en nätverksström `input` till en byte-vektor `buffer` (med start i position 0). Förutsatt att det finns en omgivande try-catch-sats som fångar eventuella exceptions tycker Kodnings-Kristofer först att det verkar lockande att skriva så här:

```
input.read(buffer,0,100);
```

a) Dock inser Kodnings-Kristoffer att det är en olämplig lösning. Varför är det ovan beskrivna sättet att läsa in 100 byte olämpligt?

(1p)

b) Skriv en korrekt kodsekvens som använder operationen `read` som den är deklarerad ovan för att läsa in 100 byte.

(2p)

Slut på del 1 – lämna in och hämta ut del 2!