# Tentamen
# EDAF85 – Realtidssystem

### 2022-04–27, 08.00–13.00

Det är tillåtet att använda Java snabbreferens och miniräknare. Det går bra att använda både engelska och svenska uttryck för svaren.

Den här tentamen består av två delar: *teori*, fråga 1-5 (15 poäng), och *programmeringsuppgifter*, fråga 6-7 (15 poäng). För godkänd (betyg 3) krävs preliminärt sammanlagt hälften av alla 30 möjliga poäng samt en tredjedel av poängen (5) på varje del för sig.

*Formelsamling*

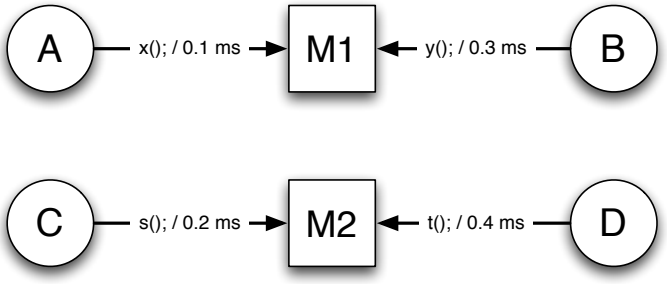
$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$$

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

**1.** Vad är ett kontextbyte (eng. *context switch*)? Förklara med en eller två meningar. *(1p)*

**2.** Vad menar vi med kapplöpning (eng. *race condition*) i ett realtidssystem? Förklara kortfattat. *(1p)*

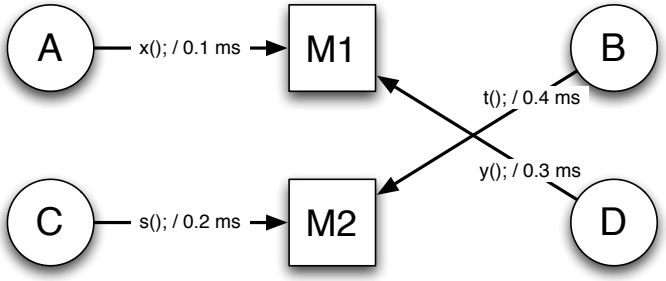
**3.** De fyra trådarna A, B, C och D, som har de högsta prioriteterna i ett realtidssystem, kommunicerar med varandra med hjälp av monitorerna M1 och M2 enligt följande figur:




Monitoroperationerna `x`, `y`, `s` och `t` anropas av trådarna (en gång varje gång en tråd körs) på det sätt som visas i figuren. För varje monitoroperation visas den maximala exekveringstiden i figuren. Tråd A har högst prioritet, B näst högst, C näst lägst och D lägst prioritet. Trådarna kommunicerar inte med trådar med lägre prioritet.

a) Vad är den maximala blockeringstiden för var och en av de fyra trådarna? *(2p)*

b) Tänk dig att vi ändrar figuren så att den ser ut enligt nedanstående:




Vad är den maximala blockeringstiden för var och en av trådarna nu? *(2p)*

**4.** Betrakta ett system bestående av tre oberoende trådar (dvs att de inte blockerar varandra) som exekverar periodiskt med följande egenskaper (C = värstafallsexekveringstid, D = deadline, T = period).

| Thread | C (ms) | D (ms) | T (ms) |
|---|---|---|---|
| A | 3 | 4 | 10 |
| B | 1 | 8 | 8 |
| C | 2 | 6 | 6 |

a) Är systemet schemaläggningsbart, dvs kommer systemet att klara alla sina deadlines, om vi använder Rate Monotonic Scheduling? För att få poäng på uppgiften måste du motivera svaret. Ett enkelt ja/nej ger inga poäng. *(2p)*

b) Vad händer om vi i stället använder oss av Deadline Monotonic Scheduling? Är systemet schemaläggningsbart i detta fall? För att få poäng på uppgiften måste du återigen motivera svaret. *(2p)*

**5.** I ett Javaprogram hittar vi tre trådinstanser, R, S och T, som i sina respektive run()-metoder exekverar nedanstående linjära sekvenser av semaforoperationer[1] på de fem mutexsemaforerna A, B, C, D och E (mellanliggande kod som är beroende av semaforerna representeras av funktionsanropen på formen `useXY()`', där ”XY” avser att semaforerna X och Y måste vara tagna när koden utförs).

a) Rita en resursallokeringsgraf för systemet. Förklara (motivera) varför systemet skulle kunna råka ut för dödläge och i vilken situation eller situationer detta skulle kunna inträffa. *(3p)*

b) Föreslå ändringar i ordningen som semaforerna tas som garanterar att systemet inte råkar ut för dödläge. Fortfarande gäller att för varje anrop av ”useXY();” måste motsvarande semaforer vara tagna. Eftersom `useA()` och `useB()` tar lång tid att exekvera ska inga onödiga semaforer vara låsta när de exekveras. *(2p)*

```
1  TRÅD T
2  ========
3  C.acquire();
4  D.acquire();
5  A.acquire();
6  useCDA();
7  A.release();
8  D.release();
9  C.release();
10
11 TRÅD S
12 ========
13 B.acquire();
14 useB();
15 D.acquire();
16 useBD();
17 B.release();
18 E.acquire();
19 useDE();
20 E.release();
21 D.release();
22
23 TRÅD R
24 ========
25 E.acquire();
26 D.acquire();
27 useED();
28 D.release();
29 E.release();
30 A.acquire();
31 useA();
32 B.acquire();
33 useAB();
34 B.release();
35 A.release();
```

[1] För dig som läst kursen före 2020: Semaforoperationerna `take()` och `give()` som användes då du läste kursen ersattes 2020 med operationerna `acquire()` och `release()` i samband med att vi övergick till att använda Javas standardbibliotek för semaforer. Betydelsen är dock densamma – det är bara namnen som skiljer.

## 6. Styrning av tillverkningsmaskin

En tillverkningsmaskin i en fabrik utför en serie av olika moment för att tillverka en produkt. Momenten utförs efter varandra i en bestämd ordning och ibland delvis parallellt. Varje enskilt moment använder separata delar av maskinen för att utföra sitt arbete och för varje moment finns det i maskinens styrprogram en separat tråd som hanterar just det momentet. När respektive tråd får order om att utföra sitt arbete så utför den en iteration av sitt arbete och inväntar sedan en ny order innan den upprepar samma moment igen. Dessutom finns det en central tråd som koordinerar alla styrtrådarnas arbeten så att alla moment utförs i korrekt ordning och ser till att timingen mellan momenten är korrekt. Genom att skicka ut en sekvens av startorder vid rätt tillfällen i tiden till de olika styrtrådarna kan koordinatortråden se till att de olika delarna av tillverkningsmaskinen samverkar på ett korrekt sätt.

För att underlätta kommunikationen mellan den centrala koordinatortråden och de många separata styrtrådarna som hanterar de enskilda tillverkningsmomenten finns en javaklass `Selector` som implementerar interfacet `SelectorInterface` enligt nedan. De olika styrtrådarna identifieras med ett icke-negativt heltal (0 eller större).

```
interface SelectorInterface {
    /** Ger styrtråd id order om att utföra en iteration av sitt arbete. Metoden
         blockerar tills den aktuella styrtråden har tagit emot ordern och
         påbörjat sitt arbete (dvs har lämnat sitt anrop av awaitTrigger().
         Endast den centrala styrtråden anropar denna metod. */
    public void trigger(int id);

    /** Anropas av de olika styrtrådarna när de är redo att ta emot en order
        om att påbörja sitt arbete. Den anropande tråden blockerar tills den
        centrala styrtråden anropar trigger() med samma id som awaitTrigger()
        anropades med. När styrtråden har utfört en iteration av sitt arbete
        anropas denna metod återigen. */
    public void awaitTrigger(int id);
}
```

För att ytterligare tydliggöra hur klassen Selector är tänkt att fungera kan du betrakta följande enkla testprogram:

```
class SelectorTest {
    public static void main(String [] args) {
        SelectorInterface s = new Selector();
        for (int i=0;i<5;i++) {
            new Activity(s,i).start();
        }
        s.trigger(2);
        // Vänta en sekund innan vi ger order till aktivitet 4 att starta.
        try { Thread.sleep(1000);} catch(InterruptedException e) {}
        s.trigger(4);
        try { Thread.sleep(3000);} catch(InterruptedException e) {}
        s.trigger(1);
        try { Thread.sleep(2000);} catch(InterruptedException e) {}
        s.trigger(4);
        try { Thread.sleep(6000);} catch(InterruptedException e) {}
        s.trigger(3);
        // Låt aktivitet 0 löpa parallellt med aktivitet 3. Därför
        // behövs ingen fördröjning.
        s.trigger(0);
    }
}
```

```
class Activity extends Thread {
    private SelectorInterface s;
    private int id;

    public Activity(SelectorInterface s,int id) {
        this.s = s;
        this.id = id;
    }

    public void run() {
        while (true) {
            s.awaitTrigger(id);
            System.out.println("Activity "+id+" triggered!");
        }
    }
}
```

Huvudprogrammet (vilket motsvarar tillverkningsmaskinens koordinatortråd) startar fem aktivitetstrådar (motsvarande maskinens tillverkningstrådar) och triggar därefter deras aktiviteter enligt ett givet körschema. Resultatet blir (med varierande fördröjningar mellan utskriften av de olika raderna):

```
Activity 2 triggered!
Activity 4 triggered!
Activity 1 triggered!
Activity 4 triggered!
Activity 3 triggered!
Activity 0 triggered!
```

## Uppgift

Implementera klassen `Selector` som implementerar interfacet `SelectorInterface`! Använd Javas inbyggda monitorbegrepp för eventuell synkronisering av trådarna. *(8p)*

## 7. Temperaturstyrning

I den här uppgiften ska du skriva en del av styrprogrammet för reglering av temperaturen i en industriell process. Med hjälp av en mikrovågskälla kan temperaturen i processen ökas. Temperaturminskningar kan bara åstadkommas genom naturlig avsvalning. Temperaturen i processen ska styras så att den håller sig inom ett givet intervall och gränserna för intervallet får inte överskridas vare sig uppåt eller nedåt (annat än vid uppstart av systemet). Följande statiska metoder är tillgängliga för att mäta aktuell temperatur och för att sätta på och koppla ur mikrovågskällan:

```
public class ProcessControl {

    /**
     * Returns the current temperature (degrees Celsius).
     * @return the current temperature
    */
    public static double readTemp();

    /**
     * Switch the microwave source on or off.
     * @param on true if the microwave source should be switched on, false to switch off
    */
    public static void heating(boolean on);

    /**
     * Returns the target temperature.
     * @return the target temperature
    */
    public static double targetTemp();

}
```

### Uppgift

Implementera en trådklass som styr temperaturen. Temperaturen får aldrig överskrida temperaturen som returneras av `targetTemp()` och heller inte bli lägre än $5°C$ under tempåeraturen som returneras av `targetTemp()`. När mikrovågskällan är aktiverad kommer temperaturen att öka medy $5°C$ per sekund och när den är avstängd kommer temperaturen att avta med $1°C$ per sekund. Fördröjningar i systemet och begränsad noggrannhet vid mätningar kommer förstås i viss mån påverka den faktiska förmågan att klara kraven, men för uppgiftens skull ignorerar vi denna aspekt av problemet.

Din lösning måste vara rimligt effektiv i den meningen att styrtråden inte ska exekvera signifikant oftare än vad som nödvändigt för att hålla temperaturen inom de givna gränserna. För att undvika slitage på det elektromekaniska relät som styr mikrovågskällan bör det undvikas att slå på och av uppvärmningen oftare än nödvändigt. Gör en rimlig kompromiss mellan sampeltid och optimalt användande av det tillåtna temperaturintervallet. För full poäng förväntas du också motivera varför din lösning uppfyller kraven ovan och varför de avvägningar du gjort är rimliga.

*(7p)*