LUND UNIVERSITY, Faculty of Engineering Department of Computer Science

Lösningar, EDAF85 Realtidssystem

2024–08–29, 14.00–19.00

1. Ett `ReentrantLock` är en semafor som är speciellt gjord för att åstadkomma ömsesidig uteslutning och kan därför inte användas till signalering. En fördel är att den kan upptäcka felaktiga användningssätt som den mera generella `Sempahore` inte kan, som t.ex. att det inte är samma tråd som låser upp semaforen som låste den. En `Semaphore` är en så kallad *räknande* semafor som förutom att åstadkomma ömsesidig uteslutning också kan användas till signalering mellan trådar.

2. A: $5 + max(3,1) = 8\mu s$ (direkt blockering i M1 och M2)
   B: $5 + 3 = 8\mu s$ (direkt blockering i M2 och indirekt blockering i M1)
   C: $3\mu s$ (indirekt blockering i M2)
   D: $0\mu s$ (lägst prioritet...)

3. Uppgiftens tabell innehöll vid tentamenstillfället ett tryckfel som gjorde att den beskrev ett förhållande mellan trådarna som inte är möjligt i verkligheten (den lägst prioriterade tråden hade en blockeringsfaktor>0) och som upptäcktes först vid rättningen. I denna version är tryckfelet rättat. Uppgiften rättades generöst med hänsyn till felet.

   a) $U = 1/5 + 1/20 + 2/8 + 3/12 = 0{,}2 + 0{,}05 + 0{,}25 + 0{,}25 = 0{,}75$

   b) $R_A^0 = 1 + 0{,}3 = 1{,}3$
   $R_A^1 = 1 + 0{,}3 = 1{,}3$

   $R_C^0 = 2 + 0{,}4 = 2{,}4$
   $R_C^1 = 2 + 0{,}4 + \left\lceil \frac{2{,}4}{5} \right\rceil \cdot 1 = 3{,}4$
   $R_C^2 = 1 + 0{,}4 + \left\lceil \frac{3{,}4}{5} \right\rceil \cdot 1 = 3{,}4$

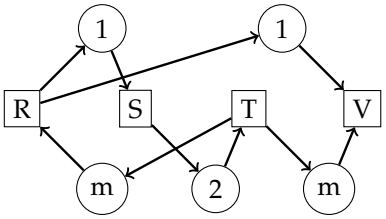
   $R_D^0 = 3 + 0{,}1 = 3{,}1$
   $R_D^1 = 3 + 0{,}1 + \left\lceil \frac{3{,}1}{5} \right\rceil \cdot 1 + \left\lceil \frac{3{,}1}{8} \right\rceil \cdot 2 = 6{,}1$
   $R_D^2 = 3 + 0{,}1 + \left\lceil \frac{6{,}1}{5} \right\rceil \cdot 1 + \left\lceil \frac{6{,}1}{8} \right\rceil \cdot 2 = 7{,}1$
   $R_D^3 = 3 + 0{,}1 + \left\lceil \frac{7{,}1}{5} \right\rceil \cdot 1 + \left\lceil \frac{7{,}1}{8} \right\rceil \cdot 2 = 7{,}1$

   $R_B^0 = 1 + 0 = 1$
   $R_B^1 = 1 + 0 + \left\lceil \frac{1}{5} \right\rceil \cdot 1 + \left\lceil \frac{1}{8} \right\rceil \cdot 2 + \left\lceil \frac{1}{12} \right\rceil \cdot 3 = 7$
   $R_B^2 = 1 + 0 + \left\lceil \frac{7}{5} \right\rceil \cdot 1 + \left\lceil \frac{7}{8} \right\rceil \cdot 2 + \left\lceil \frac{7}{12} \right\rceil \cdot 3 = 8$
   $R_B^3 = 1 + 0 + \left\lceil \frac{8}{5} \right\rceil \cdot 1 + \left\lceil \frac{8}{8} \right\rceil \cdot 2 + \left\lceil \frac{8}{12} \right\rceil \cdot 3 = 8$

   Svar: $R_A = 1{,}3ms$, $R_B = 8ms$, $R_C = 3{,}4ms$ och $R_D = 7{,}1ms$

   c) Eftersom alla värstafallssvarstider är mindre än respektive tråds deadline är systemet schemaläggningsbart.

4. a) Vi kallar den första tråden ”1”, den andra tråden ”2” och huvudprogrammet ”m”.



   b) Tråd 1: 11
   Tråd 2: 25
   Huvudprogrammet: 35

   c) Byt plats på rad 10 och 11.

5. Värdena som kan tänkas skrivas ut är 6, 7 eller 8 beroende på om (och i så fall hur) kapplöpning mellan trådarna uppstår.

6.

```
class Monitor {
    private final Queue<Request> studentQueue = new LinkedList<>();

    public synchronized void studentArrived(Request s) {
        studentQueue.add(s);
        notifyAll();
    }

    public synchronized String awaitStudent(Set<String> competence) throws InterruptedException
        while (true) {
            for (Request r : studentQueue) {
                if (competence.contains(r.getCourse())) {
                    studentQueue.remove(r);
                    return r.getStudentName();
                }
            }
            wait();
        }
    }
}

public class VideoMeetingApplication {
    private static final List<Set<String>> INSTRUCTOR_COMPETENCE = List.of(
        Set.of("EDAA45", "EDAA80", "EDAA85", "EDAA90", "EDAA01", "EDAP10"),
        Set.of("EDAA80", "EDAA85", "EDAA90"),
        Set.of("EDAA45", "EDAA01", "EDAP10"),
        Set.of("EDAA80", "EDAA85", "EDAA01"),
        Set.of("EDAA90", "EDAA01", "EDAP10")
    );

    public static void main(String[] args) {
        Monitor mon = new Monitor();

        new Thread(() -> {
            WebServer server = new WebServer();
            try {
                while (true) {
                    Request req = server.awaitStudentConnected();
                    mon.studentArrived(req);
                }
            } catch (InterruptedException e) {
                throw new Error(e);
            }
        }).start();

        for (int i = 0; i < INSTRUCTOR_COMPETENCE.size(); i++) {
            VideoMeeting meeting = new VideoMeeting(i);
            Set<String> comp = INSTRUCTOR_COMPETENCE.get(i);
            new Thread(() -> {
                try {
                    while (true) {
                        String s = mon.awaitStudent(comp);
                        meeting.acceptStudent(s);
                    }
                } catch (InterruptedException e) {
                    throw new Error(e);
                }
            }).start();
        }
    }
}
```