

Tentamensskrivning - Realtidsprogrammering **2005-06-01, kl 8.00-13.00**

Anvisningar

Tillåtna hjälpmedel: inga utöver Java snabbreferens.

DAT040: En uppdelning av betyg 4 för G resp VG kommer att ske, f.ö. se EDA040.

EDA040: *För godkänt betyg* krävs att större delen av de 7 första uppgifterna (teori) *samt uppgiften 8 (den första av de båda konstruktionsuppgifterna)* behandlas *nöjaktigt*. För högre betyg krävs dessutom en acceptabel lösning till uppgift 9 (design), och för högsta betyg ett tillräckligt (c:a 30p) totalt poängantal.

Senast 2005-06-03 anslås på institutionens anslagstavla vilka som deltagit i tentamen men som, enligt institutionens noteringar, ännu inte redovisat övningarna, laborationerna eller projektet. Deras skrivningsresultat anslås eller registreras inte förrän rättelse skett eller dispens erhållits från ansvarig lärare. Rättelse skall göras senast 2005-06-08.

På anslagstavlan anslagen restlista samt resultat av tentamen kommer även att publiceras via kursens hemsida (www.cs.lth.se/EDA040), i tillämpliga fall lösenordsskyddat (id=eda040, pwd=ht04).

Uppgifter

1. Om man vid implementeringen av ett realtidssystem använder sig av s.k. statisk schemaläggning (static scheduling) för samtliga trådar/processer, vad innebär detta för de systemrutiner som utgör stöd för ömsesidig uteslutning (synchronized, wait, notify, samt motsvarande rutiner i aktuell JVM och/eller OS)? (1p)

2. Hasse Hacker har utvecklat ett realtidssystem som i den just nu aktuella tillämpningen fungerar korrekt både logiskt (rätt resultat beräknas), jämlöpande (parallelliteten i tillämpningen hanteras), och tidsmässigs (acceptabla svarstider). Systemet är utvecklat och testat för operativsystemet VxWorks som tillhandahåller strikta prioriteter. Vid användning av Hasse:s system i en större simulering av det totala systemet i skrivbordsmiljö visar det sig dock att just Hasse:s del inte fungerar som avsett, vilket man lyckas spåra till att det OS som användes för simuleringen inte hade strikta prioriteter.

a) Vad kallar vi den sorts fel som Hasse:s programvara är behäftad med? (1p)

b) Följdaktligen, vad innebär detta för hur testning av realtidsprogramvara behöver utföras för att korrekthet och portabilitet skall kunna verifieras? (1p)

3. Olika varianter av semaforer stöds i kursen med några olika semaforklasser.

a) Nämn minst en anledning att skilja på en semafor för ömsesidig uteslutning (MutexSem) och en vanlig räknande semafor för exempelvis signalering (CountingSem)? (1p)

b) För att ta en semafor finns metoderna take och tryTake med olika typer av argument, men det finns ingen metod för att avläsa värdet av en (t.ex. räknande) semafor. Varför vore det en dålig ide att införa sådan funktionalitet? (1p)

4. Om man implementerar en periodisk aktivitet med en viss periodtid utan drift (t.ex. en sekundtickande tråd som i väckarklockslaborationen), så krävs viss omsorg i det fall att man bara har Java:s vanliga klasser att tillgå. Problemet är att väntan på tid (`sleep`) bara kan begäras för en tid relativt anropstidpunkten (och inte till en absolut tid), samtidigt som variationer i belastning och schemaläggning inte får orsaka drift hos tidsräkningen. Visa med några programrader hur man med vanlig Java-kod och system-metoderna

```
void Thread.sleep(long delay);
long System.currentTimeMillis();
```

implementerar en sådan sekundräkning utan drift. (3p)

5. De tre studenterna Emil-David Filipsson (EDF), Rose-Marie Strid (RMS) och Dan-Martin Svensson (DMS) planerar sina självstudier enligt olika principer. EDF jobbar alltid först klart med den uppgift (labförberedelse, tentaläsning, etc.) som först i tiden måste vara klar, och därefter nästa, o.s.v. RMS utgår istället från att alla typer av uppgifter är återkommande med en viss periodtid (tentor var åttonde vecka, labbar varannan vecka, läsa kursbok dagligen etc.), och jobbar sedan den mest frekventa uppgiften först, o.s.v. DMS gör på ett liknande sätt men utgår från vilken tid varje periodisk aktivitet tar att utföra. Betrakta fallen att de tre studenterna kan veta hur lång tid varje studiemoment kommer att ta, respektive att de helt enkelt jobbar på enligt preliminära uppskattningar som sedan ofta överskrids vid utförandet.

a) I fallet att varje studiemoment är välbestämt vad gäller tidsåtgången för respektive student, och att de jobbar på enligt sina principer utan att göra en detaljerad tidplan (bara en uppskattning av total arbetsbelastning), vem av studenterna kommer att hinna med en högre studietakt och vem kommer att i praktiken att få mera tid över till (det icke schemalagda) studentlivet i övrigt? Antag att varje person satsar på att studera så mycket som möjligt, och följer någon extra kurs om möjligt. (1p)

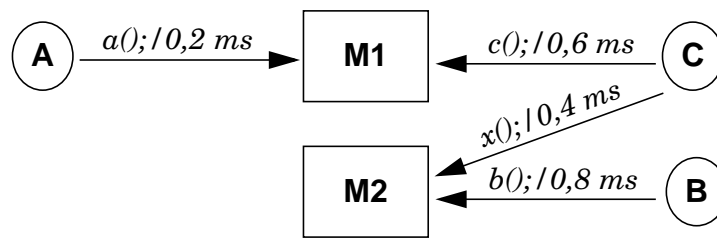
b) Om tidsuppskattningarna visar sig vara alltför optimistiska men tiden räcker fortfarande till för alla ordinarie studiemoment, vem kommer då att kunna hinna studera mest och åstadkomma sina resultat i tid? (1p)

c) Det visar sig att de ambitiösa studieplanerna (med eventuella extrakurser som beräknades hinnas med enligt uppgift a), i kombination med underskattad tidsåtgång för flera kurser resulterar i en omöjlig arbetsbelastning. Vem av de tre studenterna uppnår då det sämsta studieresultatet? (1p)

6. Ett realtidssystem är implementerat med hjälp av tre periodiska trådar kallade A, B och C. Tråd A har högst prioritet, B näst högst prioritet och C har lägst prioritet. Trådarnas värstafallsexekveringstid (C), period (T) och deadline (D) ges av följande tabell:

Tråd	C (ms)	T (ms)	D (ms)
A	2	40	40
B	3	50	50
C	4	60	60

Tråd A kommunicerar med tråd C via en monitor kallad M1. Tråd B kommunicerar med tråd C via monitorn M2. Varje gång A, B respektive C körs anropar de sina monitoroperationer (en och endast en gång per period). Monitoroperationerna och deras maximala exekveringstider framgår av följande figur.



Vilken maximal extra fördröjning kan respektive tråd råka ut för p.g.a. blockering (i den egna eller andra trådar)? Det gäller således fördröjningen utöver den egna och avbrytande tråders ordinarie exekveringstid, beroende på de i figuren angivna metoderna och deras respektive exekveringstider. (3p)

7. I ett Javaprogram hittar vi fem trådar, två stycken av typen T1, två stycken av typen T2 och en av typen T3, som i sina respektive `run()`-metoder exekverar följande linjära sekvenser av semaforoperationer på de fyra mutexsemaforerna A, B, C och D (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). Dessutom finns det en semafor S för signalering (synkronisering av exekvering, ej gällande reservering av resurs) mellan trådarna enligt följande:

T1	T2	T3
<code>A.take();</code>	<code>D.take();</code>	<code>A.take();</code>
<code>D.take();</code>	<code>B.take();</code>	<code>useA();</code>
<code>useAD();</code>	<code>useDB();</code>	<code>A.give();</code>
<code>D.give();</code>	<code>B.give();</code>	<code>B.take();</code>
<code>A.give();</code>	<code>D.give();</code>	<code>useB();</code>
<code>S.give();</code>	<code>S.give();</code>	<code>B.give();</code>
<code>C.take();</code>	<code>B.take();</code>	<code>S.take();</code>
<code>A.take();</code>	<code>C.take();</code>	<code>D.take();</code>
<code>useAC();</code>	<code>useBC();</code>	<code>useD();</code>
<code>A.give();</code>	<code>C.give();</code>	<code>D.give();</code>
<code>C.give();</code>	<code>B.give();</code>	<code>otherWork();</code>

a) Rita en resursallokeringsgraf för systemet. (2p)

b) Finns det risk för att systemet råkar ut för dödläge? Vilka trådar riskerar då att bli låsta i dödläget enligt resursallokeringsgrafen? (1p)

c) Vilka trådar vet vi att de kommer att köras? Kan vi veta huruvida `otherWork` kommer att anropas? (1p)

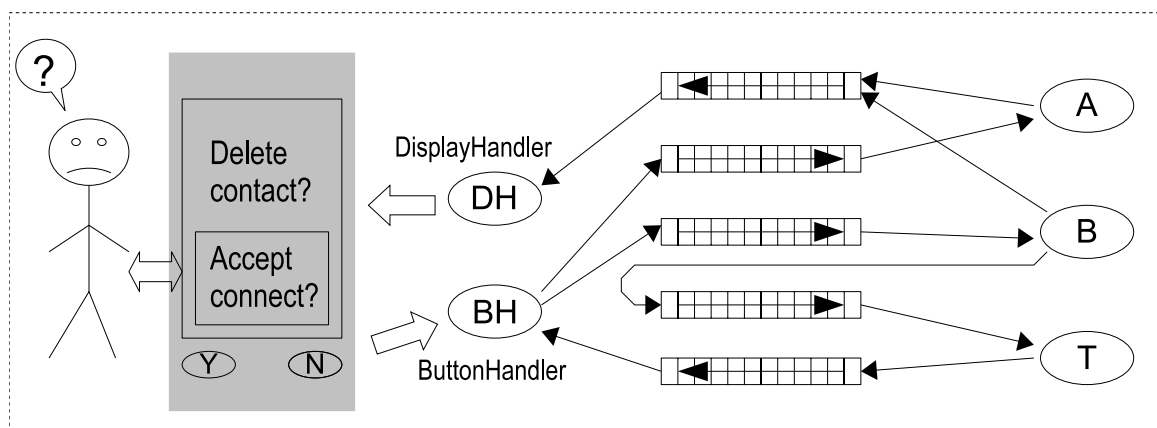
8. Realtidsinteraktion

En nyutvecklade mobiltelefon har utrustats med bra datorkraft och en påkostad grafisk färgdisplay. Användarinteraktionen innehåller (precis som på en PC) menyer, dialogrutor, jämlöpande tillämpningar som kan avbryta varandra, etc. Som en avsedd extra förbättring av interaktionen med användaren har dialogrutor med timeout införts. Dessa fungerar så att exempelvis en bakgrundstillämpning (såsom hantering av batteristatus, 3G/GPRS, detekterade Bluetooth-enheter, inkommande videosamtal, m.m.) först avbryter pågående dialog (t.ex. redigering av adressboken eller spelande av spel) med en dialogruta. Denna kan kvitteras direkt genom att trycka på avsedd knapp, men man kan också i vissa fall vänta en stund varefter dialogen försvinner (timeout, så man slipper den extra kvitteringen, vilket utgör den tänkta förbättringen).

Robert Robotsson, som (för att ge sina robotar förutsägbart beteende) alltid är vaksam vad det gäller kapplöpingsfenomen i realtidssystem, har börjat använda den nya mobilen. Han erfar direkt följande problem:

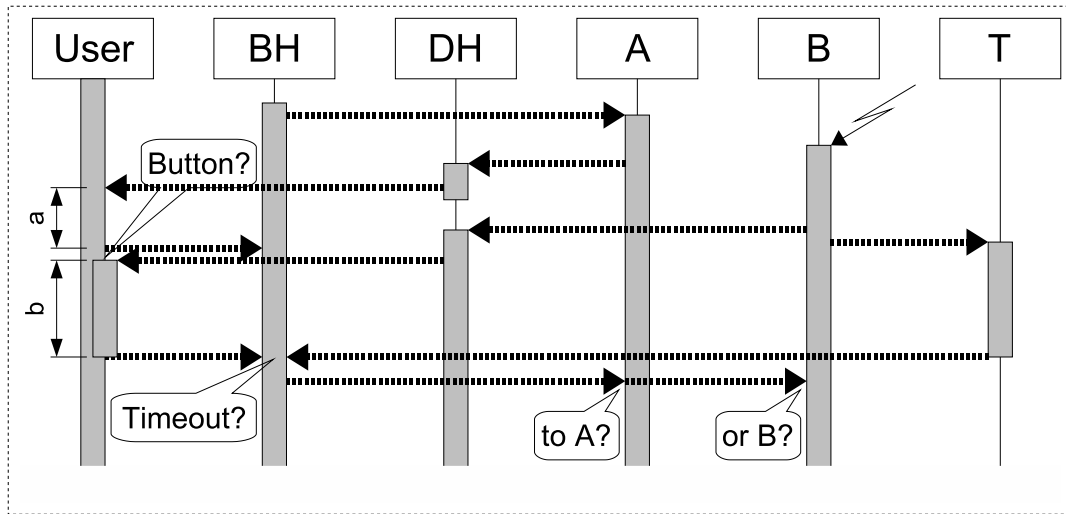
- Pågående dialog avbryts av en dialogruta, som kan kvitteras med samma knapp som den pågående dialogen. Exempelvis höll användaren på med att bläddra i telefonboken (lista med kontakter och dess tel.nr.) då menyknappen har innebörden Ring (vill inte trycka på denna), men då en annan dialog kommer upp får samma knapp innebörden Y (info om laddning el annan händelse).
- Ett scenario är att man just tryckte på den ursprungliga menyknappen när dialogrutan kom upp, och man råkar kvittera den senare (utan att hinna uppfatta texten). Detta motsvaras av att intervallen a resp b i sekvensdiagrammet på nästa sida kommer nära eller intill varandra, så att det blir tvetydigt vilken applikation som skall ha knapptryckningen, vilket indikeras av "Button?" i figuren.
- Ett annat scenario är att användaren efter att ha läst och förstått innebörden just ska till och kvittera med Y, men då finns det en maxtid (timeout) för dialogen som försvinner automatiskt just millisekunderna före det att kvittering sker, och knapptryckningen som var ämnad åt knappen Y hamnar på den ursprungliga knappen (och oönskat samtal rings, kontakt raderas av misstag, el.dyl.).

Implementering av programvaran sker på det aktuella mobilföretaget baserat på operativsystemet OSE, och interaktionen mellan trådarna sker med meddelandesändning (kallade signaler och programmerade i C, men motsvarande vår `RTEventBuffer` i Java). Följande figur illustrerar kommunikationen och de enligt denna design inbyggda kapplöpingsfenomenen.



Här har den ursprungliga tillämpningen modellerats med tråden A, den tillämpning som avbröt med dialogen motsvaras av tråden B. Ett antal brevlådor (en per tråd motsvarande de object av typen `RTEventBuffer` som vi normalt har för trådobjekten i kursen) illustrerar den meddelandebaserade kommunikationen. En tråd `DisplayHandler` hanterar Display:en och en tråd `ButtonHandler` hanterar knapparna. Tråden T är en timertråd som skickar ett meddelande efter en viss tid då dialogen skall försvinna.

De två knapparna Y och N delas av olika tillämpningar, och i detta exempel har A ställt frågan “Delete contact?” och sedan blivit avbruten av B som frågar “Accept connect?” gällande t.ex. en Bluetooth-enhet som kom i närheten. Med ett sekvensdiagram kan kapplöpningssproblemet i den befintliga implementeringen illustreras ytterligare:



Här visas den asynkrona kommunikationen via buffertobjekten som streckade pilar. Med den nuvarande meddelandebaserade utformningen av systemet kommer olika delar av telefonprogramvarans tillstånd att ligga fördelat på olika objekt, däribland de olika buffertarna i vilka de olika meddelandena befinner sig en varierande tid innan de når sin destination. Observera att dessa fördröjningarna inte visas i sekvensdiagrammet, utan beroende på aktuell sammanflätning av trådarnas exekvering så kan olika resultat erhållas såsom markerat av pratbubblorna i figuren. Notera vidare att:

- den ovan beskrivna meddelandebaserade implementeringen syftar bara till att beskriva problematiken;
- vi vet inte om detta är precis den aktuella implementeringen¹, och vi vill nu föreslå en helt annan implementering².

Uppgift: Föreslå en annan systemutformning som inte är behäftad med den ovan beskrivna osäkerheten i användarinteraktionen. Din lösning behöver inte vara helt generell utan det räcker om det framgår hur exemplet ovan hanteras. Implementera de delar som har med resurser, tidshantering och interaktion att göra. Förklara speciellt hur kapplöpningssproblemet undviks i din lösning. Du måste ha kvar och hantera den (antagligen förkastliga) timeout som finns för vissa dialoger. Antag att det bara förekommer en bakgrunds-tillämpning och en brådskande dialog åt gången. Beskriv egna antagande (såsom tidstämpling av knapptryckningar) och ytterligare klasser (för meddelanden, knappar, etc.).

Ledning: Samla dina tillstånd, inklusive tidsangivelser för inträffade och önskade händelser, i en gemensam monitor (synkroniserat objekt). Knapptryckningar som inte med en marginal om minst 0.2s kan hänföras till en viss applikation ignoreras genom att de skickas som argument till den statiska metoden `UI.ignore` (som i sin tur kan generera ett pip eller annan info enligt användarexperternas förslag). För att synkronisera hanteringen av timeout med övriga tillstånd så finns som bekant metoden `wait(timeout)` (vilken dock som bekant inte informerar om huruvida det var en notify eller en timeout som inträffat). (9p)

1. Alla eventuella likheter med produkter från den lokala mobiltillverkaren är rent slumpmässiga.
 2. Motsvarande problem (men med mindre felfrekvens) finns även i de grafiska användargränssnitten för våra vanliga datorer.

9. Tunnelövervakning

En principiell konstruktionsutformning (design) av ett övervakningssystem för en biltunnel behöver utföras för projektering och ev beslut om att gå vidare med implementering och installation. En motsvarande produkt, som kanske kan kompletteras med det SDK som kan köpas till, finns exempelvis på <http://www.autoscope.com> där informationen till höger har hämtats. Klarlägg med figurer och text vilka trådar, resurser, kommunikationskanaler, etc. som behövs för ett system som uppfyller följande specifikationer:

1. En tunnel med dubbelriktad biltrafik skall övervakas i båda ändarna av tunneln.
2. Vid vardera tunnelmynning finns en kameraservert som har till uppgift att:
 - a) Med 1Hz skicka bilder till central operatörsdator.
 - b) Med 5Hz detektera om det förekommer någon rörelse i bilden.
 - c) Vid rörelse fånga 25Hz bildsekvenser med längden 1 sekund.
 - d) För varje fångad bildsekvens urskilja förekommande bilnummer och för detekterat fordon beräkna riktning och hastighet.
 - e) För varje fordon, som kan ha detekterats ur flera sekvenser, skall dess hastighet och riktning skickas till central dator.
 - f) Om fordon eller rörelse detekteras men inget reg.nr., så skickas bildsekvensen till central operatör (vid central dator).
3. Kommunikation mellan olika datorer stöder skickandet av hela objekt. All läsning av objekt från nätverket är blockerande.
4. Operatörsdatorn finns på central plats; inte vid någon av tunnelmynningarna.
5. Ett operatörsgränssnitt visar uppdaterade stillbilder med frekvensen 1Hz (idle), inkomna bildsekvenser, samt detekterade problemfordon enligt följande punkt.
6. Alla inkomna reg.nr. (med plats och riktning) skickas till en separat tråd i operatörsdatorn. Denna tråd eliminerar bilar som passerat genom tunneln på normalt sätt. Resterade fordon klassas med ett eller flera av attributen: Fortkörare, U-svängare, Bortkommen, Nyskapad. Resultatet loggas i fil och visas i användargränssnittet.

Utför design av systemet, men ingen implementering. Algoritmer antas finnas i färdiga klasser (som du själv kan namnge). (9p)

Autoscope Video Detection (product example)

In recent years, a number of aboveground technologies have emerged to complement or replace inground inductive loops, which are expensive to reconfigure, have limited capabilities, and fail frequently. These new technologies include video detection, radar, ultrasonic, infrared and laser. Of these, video detection has been the most successful, providing unsurpassed richness of data as well as video images, wider coverage areas and greater versatility for demanding applications. To determine the best technology for your applications, ask yourself these questions:

- Do you want wide area detection? Accuracy in measuring vehicle counts and speed?
- Do you want to detect stopped vehicles? Congestion? Vehicles going the wrong way?
- Do you want to be able to reconfigure the detectors easily during road construction or to reflect changes in road geometry?
- Do you want "snapshots" of traffic?

Video detection is the only technology that can provide all these functions and benefits. Video detection is now more cost-effective, accurate and reliable than ever, outperforming inground inductive loops and other aboveground detection technologies under all weather and light conditions. The Autoscope system is the leader in video vehicle detection.

Tunnels

Tunnels are safer than ever, thanks to video-based incident detection and traffic management. Autoscope systems can quickly signal an alarm when incidents are detected, enabling fast response by an operator. Autoscope systems are proving their unique value to this challenging environment by detecting: Stopped vehicles, Slow traffic, slow vehicles, Wrong way vehicles, etc.

Autoscope is being utilized with SCADA (Supervisory Control And Data Acquisition) for incident management in the Clyde Tunnel. In 1992, local authorities in Glasgow decided to adopt the new European tunnel guidelines for a highly driven tunnel and included Autoscope in their refurbishing program.

[www.autoscope.com]