Linköpings universitet
Institutionen för datavetenskap 2020-06-02

Del 2: Teori

Uppgifter

1. **Notera:** För G på tentan behöver du få godkänt på denna uppgift.

   Tillsammans med det här dokumentet har du fått två lösningar på problemet i del 1 i filerna `a.c` och `b.c`. Dessa lösningar har olika styrkor och svagheter, och har i vissa fall inte ens lyckats synkronisera programmet korrekt.

   Välj en av de ovanstående lösningarna och jämför den med din lösning av del 1. Innan du börjar, fundera på vilken lösning som är mest lämplig att jämföra med. För att kunna svara bra på den här frågan vill du välja en lösning som är tillräckligt olik din lösning, annars kommer du inte kunna svara tillfredsställande på frågorna nedan. Om du inser att din lösning på del 1 innehåller fel, välj då en lösning som löser så många av felen i din lösning som möjligt och beskriv varför din lösning är fel och den givna lösningen är korrekt och vice versa.

   Jämför nedastående egenskaper i din lösning och i lösningen du har valt. Skriv sedan ner vad du kom fram till och hur du resonerade för att komma fram till ditt svar, gärna uppdelat i stycken eller rubriker som motsvarar punkterna nedan.

   1. Innehåller antingen din lösning eller den givna lösningen du har valt några synkroniseringsproblem? Beskriv vad som kan gå fel i den ena, och varför det inte kan hända i den andra.
   2. I vilka fall kan funktionen `connection_send` köras parallellt av olika trådar i din lösning och i den givna lösningen du har valt? Beskriv minst ett fall med mycket möjligt parallellism och ett med sämre möjlig parallellism.
   3. Använd de fyra villkoren för deadlock för att komma fram till om det finns risk för deadlock i din lösning, och i den givna lösningen du har valt.
   4. Vad kan gå fel ifall funktionen `send` inte är synkroniserad men anropas av flera trådar samtidigt? Om du löste bonusuppgiften: Hur ser du till att detta inte kan hända i din lösning?
   5. Givet dina svar på ovanstående punkter, vilken lösning tror du har bäst prestanda om man skickar många fakturor till ett fåtal anslutningsobjekt? Skulle det vara annorlunda ifall man visste att de flesta fakturor skickas till e-postaddresser på olika servrar, det vill säga, vi vet att det finns ungefär ett anslutningsobjekt per faktura?

   För godkänt på den här uppgiften vill vi se:

   - Att ditt svar är mellan 700 och 1500 ord, exklusive eventuella kodexempel.
   - Att du har diskuterat alla punkter som finns i uppgiften ovan.
   - Att du kan förklara varför din lösning är korrekt, eller med hjälp av en referenslösning kan förklara varför din lösning är felaktig och hur den kan korrigeras.
   - Att du kan förklara skillnaden mellan din lösning och den valda givna lösningen med avseende på korrekthet och möjlig parallellism.

2. I ett system körs tre processer, *P1, P2* och *P3*. I systemet finns det också tre typer av resurser, *A, B* och *C*. Tabell 1 visar hur många resurser som finns, hur systemets resurser är allokerade i nuläget och det maximala resursbehovet för varje process.

	A	B	C
P1	12	5	3
P2	3	1	8
P3	9	10	4

Maximal resursanvändning

	A	B	C
P1	9	3	2
P2	3	0	7
P3	6	6	3

Nuvarande resursanvändning

	A	B	C
P1	19	11	14

Totalt antal resurser

Tabell 1: Resurser i systemet.

   (a) Är systemet i ett säkert läge? Redovisa dina beräkningar. [1p]

   (b) Process *P3* begär en extra resurs av typ *B*. Ska begäran tillåtas enligt Banker's algorithm? Redovisa dina beräkningar. [1p]

   (c) Vad är tabellen med nuvarande resursanvändning efter föregående uppgift? [1p]

   (d) I systemet som kör Banker's, är det i allmäna fall möjligt att lägga till nya instanser av en resurs utan att det orsakar problem? Motivera ditt svar. [1p]

3. Ersätt din synkronisering i den givna koden i del 1 med lämpliga atomiska operationer från `wrap/atomics.h` (du behöver **inte** lösa bonusuppgiften från del 1). I den här uppgiften ska du alltså **inte** använda varken lås, semaforer eller condition variables.

   (a) Skriv i dokumentet du skickar in att du har löst uppgiften. Skicka in den modifierade versionen av `invoice.c` separat i Lisam (del 2, uppgift 3). [3p]

   (b) Varför är det inte möjligt att eliminera *busy-wait* med hjälp av enbart atomiska operationer? [1p]

   Notera: Det går att använda `atomic_swap` och `compare_and_swap` på pekarvariabler, trots att exempelimplementationerna i `wrap/atomics.h` endast hanterar heltal.

   Notera: Om du vill få koden att kompilera kan du tyvärr inte använda atomiska operationer för variabler av typen `bool`. Använd i så fall `int` i stället. Som i del 1 är det dock inget krav att koden du lämnar in kompilerar.