
Del 2: Teori

Bedömning

För att få godkänt på tentan krävs att uppgift 1 bedöms som godkänd. På resterande uppgifter kan du samla bonuspoäng för högre betyg. Se preliminära gränser på försättsbladet.

Notera att denna del innehåller uppgifter som är unika för dig.

Inlämning

Denna del lämnas in som text. Använd ett program som låter dig skapa rubriker, styckeformaterad text och kodexempel, exempelvis LibreOffice eller Word. Det är också okej att skriva direkt i en textfil med en vanlig textredigerare. Använd i så fall en struktur som gör det tydligt vad som är rubriker, stycken och kodexempel, exempelvis Markdown eller Org-Mode i Emacs.

Dokumentet lämnas sedan in i Lisam senast klockan 12:00. Lämna in dina lösningar antingen som PDF, eller som en textfil beroende på vad du har använt för att skriva dina lösningar.

Din inlämning kommer att köras genom Urkund för plagiatkontroll efter inlämning.

Uppgifter

1. 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 datastrukturen 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. Kan funktionen `run_in_parallel` anropas av flera trådar samtidigt i din lösning och i den givna lösningen du har valt? Varför/varför inte?
3. Kan din lösning och den givna lösningen du valt köra funktionen som skickades till `run_in_parallel` parallellt, eller finns det något som förhindrar det?
4. Använd de fyra villkoren för deadlock för att komma fram till om det finns risk för deadlock i din lösning, eller i den givna lösningen du har valt.
5. Givet dina svar på ovanstående punkter, vilken lösning tror du har bäst prestanda om man kör funktioner som är dyra (= tar lång tid) att exekevera och varför? Skulle någon annan lösning vara bättre om funktionerna går snabbt att exekevera?

För godkänt på den här frågan vill vi se:

- Att ditt svar är mellan 1000 och 2000 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.

2. I ett system körs tre processer, $P1$, $P2$ och $P3$. I systemet finns det också tre typer av resurser, A , B och C , och 5 instanser av vardera. Tabell 1 visar hur systemets resurser är allokerade i nuläget och det maximala resursbehovet för varje process. Systemet är i ett säkert läge.

	A	B	C
P1	4	2	2
P2	3	2	3
P3	3	2	4

Maximal resursanvändning

	A	B	C
P1	1	0	1
P2	1	0	1
P3	1	1	1

Nuvarande resursanvändning

Tabell 1: Resurser i systemet.

- (a) Process $P3$ begär en extra resurs av typ A . Ska begäran tillåtas enligt Banker's algorithm? Redovisa dina beräkningar. [2p]
- (b) Vad ska göras då en begäran inte tillåts enligt Banker's algorithm? Motivera ditt svar. [1p]
1. Processen som begärde en extra resurs måste avslutas – det kommer **garanterat** bli deadlock om den fortsätter köra.
 2. Processen som begärde en extra resurs får vänta – det kommer **garanterat** bli deadlock om den fortsätter köra.
 3. Processen som begärde en extra resurs får vänta – det **kan** bli deadlock om den fortsätter köra.
 4. Processen som begärde en extra resurs får fortsätta – det är först vid nästa resursförfrågan som problem kan uppstå, och vi kan lösa problemen då.
3. Utgå från din lösning av uppgiften i del 1. Använd atomiska operationer i stället för lås, semaforer och condition-variables för att synkronisera koden. Din lösning får innehålla busy-wait. [5p]

Tillgängliga atomiska operationer finns i filen `wrap/atomics.h`.

Infoga koden i det dokument du skickar in till Lisam.