OSM, G2

Anders Iversen
Mikkel Mastek
Anders Bjerg Pedersen

23. februar 2008

G2.1 En ring af tråde

I denne trådstafet anvender vi to mutex'er og en condition variabel til hver af trådene. Én mutex bruges til at låse vores count-variabel, der holder styr på det antal gange, stafetten har været rundt, og én til en global variabel ready_threads, der tæller, hvor mange tråde der er klar til at modtage arbejde. Condition-variablene bruges til at signalere til næste tråd i rækken. Tråden med id=0 (den første) tæller altid stafetten én ned, indtil den når 0. Når en tråd ser, at stafetten er nået til 0, sender den stafetten videre (pånær sidste tråd, hvor signalet blot går tabt) og terminerer derefter.

Variablen ready_threads er indført, for at main()-metoden ikke starter stafetten, før trådene er klar. Anvendelsen af ready_threads er ikke 100% holdbar, da der potentielt kunne ske det, at første tråd bliver signaleret af main()-metoden, før den er nået til pthread_cond_wait(). Vi har ikke umiddelbart kunnet finde en bedre løsning på dette.

Herunder ses output af kørsel af ring.c. Det ses, at stafetten løber 5 gange igennem i korrekt rækkefølge, og at trådene derefter terminerer, igen i korrekt rækkefølge:

```
app-1 >./ring
work(): thread 0, stafet count = 4. Signalerer tråd 1
work(): thread 1, stafet count = 4. Signalerer tråd 2
work(): thread 2, stafet count = 4. Signalerer tråd 3
work(): thread 3, stafet count = 4. Signalerer tråd 4
work(): thread 4, stafet count = 4. Signalerer tråd 0
work(): thread 0, stafet count = 3. Signalerer tråd 1
work(): thread 1, stafet count = 3. Signalerer tråd 2
work(): thread 2, stafet count = 3. Signalerer tråd 3
work(): thread 3, stafet count = 3. Signalerer tråd 4
work(): thread 4, stafet count = 3. Signalerer tråd 0
work(): thread 0, stafet count = 2. Signalerer tråd 1
work(): thread 1, stafet count = 2. Signalerer tråd 2
work(): thread 2, stafet count = 2. Signalerer tråd 3
work(): thread 3, stafet count = 2. Signalerer tråd 4
work(): thread 4, stafet count = 2. Signalerer tråd 0
work(): thread 0, stafet count = 1. Signalerer tråd 1
work(): thread 1, stafet count = 1. Signalerer tråd 2
work(): thread 2, stafet count = 1. Signalerer tråd 3
work(): thread 3, stafet count = 1. Signalerer tråd 4
work(): thread 4, stafet count = 1. Signalerer tråd 0
work(): thread 0, stafet count = 0. Terminerer tråd.
work(): thread 1, stafet count = 0. Terminerer tråd.
work(): thread 2, stafet count = 0. Terminerer tråd.
work(): thread 3, stafet count = 0. Terminerer tråd.
work(): thread 4, stafet count = 0. Terminerer tråd.
Main(): Done.
app-1 >
```

G2.2 Trådsikret arbejdskø med trådpulje

I opgaven har vi brugt den udleverede kø-løsning fra G1.
Vi er opmærksomme på, at der er noget galt med prioriteten i køen - se kommentar i `main()`, men det har ingen praktisk indflydelse på løsningen af denne opgave.
Når flere tråde kan tilgå delt data samtidigt (f.eks. en arbejdskø), skal man sørge for at adgangen til dataen sker udeleligt. Derfor har vi sørget for at inden vi tilgår arbejdskøen - enten for at fjerne, indsætte eller tjekke for om den er tom - så låser vi med en mutex, så ingen andre tråde kan ændre noget ved køen imens. Vi har implementeret en tæller (`workcounter`), der holder øje med om der er nogen tråde der arbejder. Den bruger vi i det tilfælde hvor køen er tom, men der stadig er nogle tråde der arbejder, og potentielt kan indsætte mere arbejde i køen: er køen tom, men `workcounter > 0`, så sætter vi tråden til at vente.
Når en tråd har opdaget at der ikke er nogen andre der arbejder, og køen er tom, så afslutter den. For at evt. andre ventende tråde også afslutter, så signalere tråden med et broadcast, lige inden den afslutter.

Herunder ses en test af vores implementation - nogenlunde som de 2 måder der er foreslået i opgaveteksten. I begge tilfælde starter vi 3 tråde i trådpuljen.
I første omgang tester vi for, om flere tråde kan udføre arbejde samtidigt. Vi indsætter to arbejdsopgaver med `print_int()` i køen. `print_int()` udskriver et tal og sover derefter et sekund - 5 gange efter hinanden. Som det ses af udskriften skiftes to af trådene til at arbejde, mens den tredje venter (da der kun er to arbejdsopgaver når den aldrig at arbejde).

```
app-4 > ./compile.sh
wqueue_ts_insert(): Der er indsat arbejde i køen
wqueue_ts_insert(): Køen var tom, men har nu noget i igen - Der broadcastes
wqueue_ts_insert(): Der er indsat arbejde i køen
wqueue_ts_run(): workcounter: 0. Tråd_id: -1210176624. Kø !tom?: 1
print_int(): 5
wqueue_ts_run(): workcounter: 1. Tråd_id: -1218569328. Kø !tom?: 1
print_int(): 2
wqueue_ts_run(): workcounter: 2. Tråd_id: -1226962032. Kø !tom?: 0
wqueue_ts_run(): Tråd -1226962032 venter på arbejde
wqueue_thread_pool(): Tråde oprettet - arbejd!...
print_int(): 5
print_int(): 2
print_int(): 5
print_int(): 2
print_int(): 5
print_int(): 2
print_int(): 5
print_int(): 2
wqueue_ts_run(): Tråd -1210176624 har udtaget og udført arbejde
wqueue_ts_run(): Tråd -1210176624 venter på arbejde
wqueue_ts_run(): Tråd -1218569328 har udtaget og udført arbejde
```

```
wqueue_ts_run(): Tråd_id: -1218569328 signing off
wqueue_ts_run(): Tråd -1226962032 er signaleret
wqueue_ts_run(): Tråd_id: -1226962032 signing off
wqueue_ts_run(): Tråd -1210176624 er signaleret
wqueue_ts_run(): Tråd_id: -1210176624 signing off
Done.
app-4 >
```

I anden omgang tester vi om der signaleres til ventende tråde, der venter fordi køen var tom, men en eller flere andre tråde er igang med at arbejde. De ventende tråde skulle gerne vågne op igen når de arbejdende tråde indsætter mere arbejde i køen. Vi starter med at indsætte kun en arbejdsopgave i køen med `calc2()`. Den starter med at sove lidt, så køen er tom og de andre tråde alle venter. Efter at `calc2()` har sovet lidt indsætter den tre nye arbejsopgaver i køen (`print_res()`). Som de ses af udskriften, så signalere vi med en broadcast til alle ventende tråde, ligeså snart der er indsat nye arbejdsopgaver i køen, og de ventende tråde tager fat.

```
app-4 > ./compile.sh
wqueue_ts_insert(): Der er indsat arbejde i køen
wqueue_ts_insert(): Køen var tom, men har nu noget i igen - Der broadcastes
wqueue_ts_run(): workcounter: 0. Tråd_id: -1209930864. Kø !tom?: 1
wqueue_ts_run(): workcounter: 1. Tråd_id: -1218323568. Kø !tom?: 0
wqueue_ts_run(): Tråd -1218323568 venter på arbejde
wqueue_ts_run(): workcounter: 1. Tråd_id: -1226716272. Kø !tom?: 0
wqueue_ts_run(): Tråd -1226716272 venter på arbejde
wqueue_thread_pool(): Tråde oprettet - arbejd!...
wqueue_ts_insert(): Der er indsat arbejde i køen
wqueue_ts_insert(): Køen var tom, men har nu noget i igen - Der broadcastes
wqueue_ts_insert(): Der er indsat arbejde i køen
wqueue_ts_insert(): Der er indsat arbejde i køen
wqueue_ts_run(): Tråd -1209930864 har udtaget og udført arbejde
wqueue_ts_run(): Tråd -1218323568 er signaleret
wqueue_ts_run(): workcounter: 0. Tråd_id: -1218323568. Kø !tom?: 1
print_res(): thread_id=-1209930862
wqueue_ts_run(): Tråd -1218323568 har udtaget og udført arbejde
wqueue_ts_run(): workcounter: 0. Tråd_id: -1218323568. Kø !tom?: 1
print_res(): thread_id=-1209930862
wqueue_ts_run(): Tråd -1218323568 har udtaget og udført arbejde
wqueue_ts_run(): workcounter: 0. Tråd_id: -1218323568. Kø !tom?: 1
print_res(): thread_id=-1209930862
wqueue_ts_run(): Tråd -1218323568 har udtaget og udført arbejde
wqueue_ts_run(): Tråd_id: -1218323568 signing off
wqueue_ts_run(): Tråd -1226716272 er signaleret
wqueue_ts_run(): Tråd_id: -1226716272 signing off
wqueue_ts_run(): Tråd_id: -1209930864 signing off
Done.
app-4 >
```