Godkendelsesopgave 2 i "Styresystemer og multiprogrammering"

1 Generelt

Denne ugeopgave skal afleveres senest torsdag den 21. februar 2008 klokken 6:00. Den kan løses af grupper på op til 3 personer. Besvarelsen af opgaven vil resultere i enten 0, $\frac{1}{2}$ eller 1 point. Pointene uddeles efter følgende retningslinjer:

- 0 point: besvarelsen har flere store mangler.
- $\frac{1}{2}$ point: besvarelsen opfylder i store træk kravene men har flere mindre mangler.
- 1 point: en god besvarelse der kun har få eller ingen mangler.

Det er en betingelse for at gå til eksamen på kurset at man har opnået mindst 4 point i alt, og at mindst fem ugeopgaver har fået mindst $\frac{1}{2}$ point.

Besvarelsen skal indleveres elektronisk via kursushjemmesiden på ISIS. I modsætning til første godkendelsesopgave, skal I bruge arbejdsgruppe funktionaliteten i ISIS når I afleverer for en gruppe. I skal oprette en arbejdsgruppe og tilføje medlemmerne til gruppen. Herefter kan I aflevere som en samlet gruppe under 'Aflevering af opgaver'. Systemet sørger ikke for et unikt gruppenavne, så brug følgende skabelon til at navngive Jeres gruppe:

```
efternavn1-efternavn2-efternavn3
```

Skulle dette ikke være nok til at sikre at jeres gruppenavn er unikt, kan I anvende fornavne, fødselsdage, eller tilfældige tal til at sikre unikhed.

Besvarelsen skal ske ved aflevering af en enkelt fil. Brug 'zip' eller 'tar.gz' til at samle flere filer. Filnavnet skal have følgende format:

```
efternavn1-efternavn2-efternavn3-Hold<h>-G<n>.<endelse>
```

hvor `<n>` er opgavenummeret og `<h>` er holdnummeret for den instruktor som sidst rettede Jeres opgave.

Jeres rapport skal være i et format der kan læses på DIKUs systemer uden problemer (og bør f.eks. ikke være MS Word dokumenter). Opgavenummer og navne på gruppemedlemmer skal fremgå tydelig af første side i rapporten.

2 Denne uges tema: Multiprogrammering i C med pthreads

Denne uges godkendelsesopgave handler om hvordan man anvender mutex'er til at beskytte en datastruktur, der deles mellem flere tråde, hvordan man opretter og nedlægger tråde, samt hvordan man anvender betingelsesvariable til at synkronisere afviklingen af flere tråde.

Aflevereingen skal indeholde én rapport på 1-3 sider der dokumenterer hver delopgave. Kravene til dokumentation er specificeret i hver opgave. I skal også huske at kommentere Jeres kildetekst så den let at forstå.

2.1 G2.1 En ring af tråde

Lav en ring af N tråde, hvor tråd 2 venter på tråd 1, tråd 3 venter på tråd 2, ... , tråd N venter på tråd N-1 og tråd 1 venter på tråd N. Lad tråd 1 starte med at videregive en stafet til tråd 2, og lad derefter stafetten cirkulere i ringen 5 gange inden tråd 1 erstatter den med en stopstafet, der får alle trådende til at stoppe efter de har videresendt stopstafetten.

Opgavebesvarelsen for dette delspørgsmål skal indeholde:

- Programmet i en fil med navnet "ring.c".
- Rapport: Kort beskrivelse af hvordan programmet virker.

2.2 G2.2. Trådsikret arbejdskø med trådpulje

I denne opgave skal arbejdskøen fra sidste ugeopgave udvides så flere tråde samtidigt kan tilføje og udføre arbejde. Derfor skal de kritiske regioner sikres, så trådene ikke ændrer køen samtidigt. Til gengæld skal arbejdsopgaverne kunne udføres sideløbende af flere tråde, så det bliver muligt at udnytte flere CPU'er eller multi-core CPU'er.

Opgaven skal løses ved at implementere to nye funktioner som er defineret i en ny udgave af header filen 'wqueue.h'. Funktionen 'wqueue_ts_insert' skal være en trådsikker udgave af 'wqueue_insert'.

Funktionen 'wqueue_thread_pool' skal starte en pulje af tråde som alle udtager et stykke arbejde fra arbejdskøen og udfører det. Hvis køen er tom skal tråde uden arbejde blokkere indtil der igen er nyt arbejde. Derfor skal funktionen 'wqueue_ts_insert' signalere når køen går fra tom til ikke tom. Når køen er tom, og der ikke er flere tråde som arbejder, skal alle tråde nedlægges, hvorefter 'wqueue_thread_pool' skal returnere.

For at afprøve jeres kø, skal I lave testprogrammer, som tester følgende:

- At flere tråde samtidigt kan udføre arbejde. Lav evt. flere arbejdsopgaver som kontinuerligt udskriver et unikt ID.
- At tråde blokerer når køen er tom og bliver vækket når der kommer mere arbejde. Lav evt. en arbejdsopgave som laver en kort udregning (1+ sekunder), udskriver et tråd ID ('pthread_self'), og herefter lægger et antal arbejdsopgaver på køen. Disse opgaver skal bare udskrive et tråd ID. På denne måde skulle I kunne se at de tråde som ikke først fik en opgave bliver vækket og udfører de nye opgaver.
- At 'wqueue_thread_pool' returnerer når køen er tom og der ikke flere tråde som arbejder.

Opgavebesvarelsen for dette delspørgsmål skal indeholde:

- Den udvidede implementation i en fil med navnet "wqueue.c".
- Headeren "wqueue.h", hvis der er ændringer i denne.
- Kildetekst som Jeres implementation bruger - f.eks. prioritetskøen fra forrige opgave.
- Testprogram kildetekster.
- Rapport: Beskrivelse af implementation og de overvejelser I har gjort i forbindelse med testprogrammer.

Hvis I ikke er sikre på at jeres arbejdskø fra forrige opgave fungerer, kan I i stedet hente et forslag til en sådan på kursushjemmesiden (vil blive tilgængelig efter afleveringsfristen for G1).