Datalogisk Institut – Københavns Universitet
Foråret 2010

Godkendelsesopgave 1 i “Styresystemer og multiprogrammering”

1. Generelt

Denne ugeopgave stilles fredag den 5. februar 2010 og skal afleveres tirsdag den 16. februar 2010 klokken 06:00. Den kan løses af grupper på 2 personer. Besvarelsen af opgaven vil resultere i enten 0, ½ eller 1 point. Pointene uddeles efter følgende retningslinjer:

- 0 point: besvarelsen har flere store mangler.
- ½ 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 ½ point.

Besvarelsen skal indleveres elektronisk via kursushjemmesiden på SIS. Besvarelsen bør ske ved aflevering af en enkelt fil. Brug ’zip’ eller ’tar.gz’ til at samle flere filer. Opgavenummer og navne på gruppemedlemmer skal fremgå tydelig af første side i besvarelsen. Når I indleverer elektronisk bør efternavne på de to gruppemedlemmer indgå i filnavnet - desuden skal opgavenummer fremgå af navnet:

efternavn1-efternavn2-G1.<endelse>

Skulle dette ikke være nok til at sikre at jeres filnavn er unikt, kan I anvende fornavne, fødselsdage, eller tilfældige tal til at sikre unikhed. Jeres aflevering skal være i et format der kan læses på DIKUs systemer uden problemer (og bør f.eks. ikke være MS Word dokumenter).

I behøver kun aflevere én fuld besvarelse, men for at I selv nemt kan se evalueringen af de enkelte opgaver, skal de gruppemedlemmer, der ikke afleverer den fulde besvarelse blot aflevere forsideinformationen: dvs. opgavenummer og navne på gruppemedlemmer.

2. Denne uges tema: Dynamisk lagerallokering

Formålet med denne uges opgaver er at se nærmere på dynamisk lagerallokering i C, samt brugen af pegere (pointer) til både data og funktioner. Løsningerne må ikke anvende en statisk øvre grænse for antallet af elementer med mindre det er angivet i opgaven. I stedet kan der anvendes enkelt eller dobbelthægtede lister eller lignende.

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 er let at forstå.

2.1 G1.1. Prioritetskø

I denne opgave skal I designe og implementere en prioritetskø. Brugerne af køen skal kunne indsætte og udtage elementer, i form af pegere til vilkårlige data (void *). Ved indsættelse angives prioriteten som et positivt heltal. Bemærk at det skal være tilladt at indsætte flere elementer med samme prioritet. Ved udtagelse skal det element som har højst prioritet (største heltal) udtages først. Hvis flere elementer har samme prioritet skal elementerne udtages i den rækkefølge de blev indsat (FIFO=first-in-first-out).

I skal implementere køen ved at lave en implementation af grænsefladen som er defineret i den vedlagte headerfil ”pqueue.h”. Hver funktion er detaljeret beskrevet i headerfilen. Design af datastrukturer og oprettelse og nedlæggelse af køen, er op til Jer. I skal også sørge for at fejl rapporteres til brugeren (returværdier). Læg Jeres implementation i filen ”pqueue.c”

For at afprøve jeres kø, skal I designe og implementere et testprogram, der tester følgende:

- At alle indsatte elementer også kommer ud igen.
- At udtagelse bliver gjort i henhold til prioritet.
- At FIFO virker for elementer med samme prioritet.

Opgavebesvarelsen for dette delspørgsmål skal

indeholde:

- Prioritetskøimplementationen i en fil med navnet ”pqueue.c”.
- Headeren ”pqueue.h”, hvis der er ændringer i denne.
- Testprogram kildeteksten i en fil med navnet ”pqueue-test.c”.
- Rapport: Beskrivelse af implementation og de overvejelser I har gjort i forbindelse med testprogrammet.

2.2 G1.2. Prioriteret arbejdskø

I denne opgave skal I designe og implementere en arbejdskø som er baseret på prioritetskøen fra opgave G1.1. Arbejdskøen skal kunne bruges til at gemme eller fordele arbejdsopgaver med. Brugerne af køen skal kunne indsætte og udtage arbejdsopgaver, hvor en arbejdsopgave er beskrevet af en C funktion og noget data. C funktionen skal tage netop et argument - en peger til typen void. Dvs. at alt data skal kunne medgives ved hjælp af en enkelt peger (drejer

det sig om mere komplekse data, kan det derfor være nødvendigt først at lave en enkelt datastruktur, der kan indeholde/referere al nødvendig data).

I skal implementere arbejdskøen ved at lave en implementation af grænsefladen som er defineret i den vedlagte headerfil ”wqueue.h”. Hver funktion er detaljeret beskrevet i headerfilen. Design af datastrukturer og oprettelse og nedlæggelse af køen, er op til Jer. I skal også sørge for at evt. fejl rapporteres til brugeren (returværdier). Læg Jeres implementation i filen ”wqueue.c”

For at afprøve jeres kø, skal I implementere et testprogram, der indsætter to forskellige typer af arbejdsopgaver i køen:

- En funktion der udskriver det medleverede data som en streng.
- En funktion der udregner summen af to tal, konverterer resultatet til en streng og udskriver det ved at indsætte en ny arbejdsopgave, der anvender den ovenstående funktion.

Testprogrammet skal efter indsættelse af et antal beregningsarbejdsopgaver, udtage et element af gangen fra arbejdskøen og udføre arbejdsopgaven, indtil køen er tom.

Opgavebesvarelsen for dette delspørgsmål skal indeholde:

- Arbejdskøimplementationen i en fil med navnet ”wqueue.c”.
- Headeren ”wqueue.h”, hvis der er ændringer i denne.
- Testprogram kildeteksten i en fil med navnet ”wqueue-test.c”.
- Rapport: Beskrivelse af implementation og de overvejelser I har gjort i forbindelse med testprogrammet.