

Operativsystemer og Multiprogrammering

G-opgave 1

Ronni Elken Lindsgaard - 0911831791
Hans-Kristian Bjerregaard - 0612862087

16. februar, 2010

1 Prioriteret kø (pqueue)

1.1 Datastruktur

Som grundlæggende datastruktur for prioritetsskøen har vi valgt at anvende en enkelthægtet liste. Listen er sorteret efter prioritet således at det næste element der skal returneres altid er hovedet af listen.

For at sikre *FIFO* bliver elementer indsat således at nye elementer bliver indsat sidst i delmængden af elementer med samme prioritet.

Optimalt vil man vælge at implementere en min-hob med prioriteter og så for alle elementer med samme prioritet kan man så bruge en linket liste for at bibeholde *FIFO*.

1.2 Implementering

1.2.1 pqueue_insert

Ved indsætning allokeres først plads til det nye element. Lykkes dette ikke returneres *-1* for at indikere fejl. Ellers gemmes data for det nye element i den allokerede hukommelse.

Elementet indsættes i prioritetsskøen ved at scanne igennem køen til der enten nås et element med en højere prioritetsværdi eller der ikke er flere elementer i listen. Under denne scan vedligeholdes der to variabler der peger på hhv. det nuværende og det sidst tjekkede element. Scanningen fortsætter så længe der er flere elementer i listen og det næste element har en prioritetsværdi der er mindre end eller lig med prioritetsværdien for det element der skal indsættes. Det er vigtigt at man scanner videre hvis prioritetsværdierne er ens for at sikre *FIFO*. Når scanbetingelsen ikke længere er opfyldt vil de to vedligeholdte værdier pege på de elementer i køen, som elementet skal indsættes mellem.

For at indsætte elementet sættes det forrige element til at pege på det element, der skal indsættes, og det indsatte element, peger på det element hvor betingelsen blev opfyldt. Er det forrige element *NULL*, skal elementet indsættes i hovedet af køen og derved skal selve pqueue opdateres til at pege på elementet.

Udover opgavens krav har vi implementeret starvation check, hver 20ende gang pqueue_insert køres bliver en funktion kørt der tjekker at et job ikke er for gammelt, hvis det har eksisteret for lang tid bliver prioriteten forhøjet så jobbet køres inde for en ordentlig tidsramme.

1.2.2 pqueue_remove

Da prioritetsskøen er implementeret som en linket liste, sorteret efter elementernes prioritet, (incl. *FIFO*) er det altså listens hoved der skal returneres. Derved skal der bare tjekkes om listen er tom og returneres *NULL* hvis det er tilfældet. Ellers

returneres hovedet og prioritetskøen sættes til at pege på det næste element, som nu er listens hovede.

1.2.3 Afprøvning

Testene beror på om elementer bliver lagt til og trukket fra i den korrekte orden. Vi tester først om et element med en prioritet bliver indsat det korrekte sted. Derefter tester vi om `remove` virkelig fjerner listens hoved og opdaterer queueen korrekt.

2 Prioriteret arbejdskø (wqueue)

2.1 Implementering

2.1.1 `wqueue.insert`

For at indsætte et job i en arbejdskø der er implementeret via `pqueue`, skal der bruges en struktur for at samle funktionen og dens inddata i ét element der kan bruges i køen. Dette gøres via strukturen `job_t`. Indsætning er så ganske simpel: Allokér hukommelse til `job_t` (og returner -1 hvis det fejler), indsæt funktion og data i `job_t`, og indsæt `job_t` i prioritetskøen med den angivne prioritet.

2.1.2 `wqueue.run`

For at køre et job hentes det først ud af prioritetskøen. Er denne tom returneres 0 ellers hentes funktionen og dataene ud af jobbet og funktionen køres med de hentede data som input.

2.1.3 `wqueue.add`

To tal skal lægges sammen, konverteres til en streng og selve opgaven at printe resultatet skal lægges i køen. For at sikre det er præcis den data som vi tilføjer der bliver lagret er det nødvendigt at allokere plads. Vi har valgt at allokere 5 chars da det bør være rigeligt til at repræsentere resultatet som en streng. Herefter kaldes `wqueue.insert` som sørger for at lægge det i køen.

2.1.4 Afprøvning

I stedet for simpel data skal der nu udføres `wqueue.add` i stedet som sørger for at lægge de 2 heltalt sammen. Vi har valgt at teste at jobsne bliver udført i den korrekte rækkefølge i form af hvilken prioritet de har. Vi kunne have udført mere grundige tests, men de tests vil handle om `pqueue` som allerede er testet.