Godkendelsesopgave 4 i "Styresystemer og multiprogrammering"

Generelt

Denne ugeopgave skal afleveres senest torsdag den 6. marts 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 skal 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 gruppenavn, 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 PDF eller ASCII format, for at lette instruktorernes rettearbejde. Opgavenummer og navne på gruppemedlemmer skal fremgå tydeligt af første side i rapporten.

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.

Tjekliste for aflevering:

- **Opgavenummer og navne på gruppemedlemmer skal fremgå tydeligt af første side i rapporten.**
- Brug arbejdgruppe funktionaliteten i ISIS.
- Følg navngivningsskabelonen for arbejdsgrupper og afleveringsfil (se ovenfor).

- Rapport i PDF eller ASCII format.
- Brug 'zip' eller 'tar.gz' til at samle alle filer i én fil.
- Kontroller at I har opfyldt kravene til opgaveaflevering for hvert delspørgsmål.

Introduktion

I denne opgave skal I dels arbejde praktisk med at dele hukommelse (shared memory) mellem processer, dels skal I løse en teoretisk opgave om sidetabeller.

Første skridt

Start med at udpakke den udleverede kildetekst. Gå ind i 'src-g4' biblioteket og kør følgende:

```
make
./pipe 4
./shm
```

Den første kommando kompilerer alt kildeteksten ud fra en beskrivelse i filen 'Makefile'. Det er ikke vigtigt at I forstår denne del og I kan fra output'et se hvordan kompileren bliver kaldt. I behøver ikke at tilføje nye filer i denne opgave, så I kan bruge denne kommando hver gang i skal kompilere.

Den anden kommando kører programmet 'pipe.c', som er det program der skal bruges som udgangspunkt i opgave G4.1.

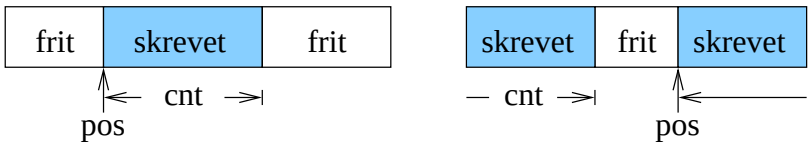
Den sidste kommando kører programmet 'shm.c', som er et eksempel på opstart af flere processer, delt hukommelse og POSIX semaforer.

G4.1: Pipe i delt hukommelse

I denne opgave skal I lave en pipe som er baseret på delt hukommelse. En pipe er en cirkulær begrænset buffer som processer kan skrive og læse data fra. På den måde kan de udveksle data på tværs af adresserum. En pipe har en fast bufferstørrelse *size*, hvilket betyder at hvis en proces prøver at skrive mere end der kan være i bufferen blokerer den og venter på at der er en anden proces som tømmer pipe'en ved at læse. I en ikke-blokerende (non-blocking) version vil processen ikke få alt data skrevet og må derfor prøve igen på et senere tidspunkt. Det er den ikke-blokerende version som bruges i denne opgave.

Unix har også en pipe funktionalitet, hvor bufferen lever i kernen (man pipe).

I programmet 'pipe.c' er der implementeret en pipe som ikke bruger delt hukommelse og hvis læse og skrive operationer er ikke-blokerende. Pipe'en er baseret på en cirkulær buffer som vist i Figur 1.



Figur 1: Pipe buffer.

Variablen 'pos' er det indeks fra hvilket den næste byte skal læses og 'cnt' er antallet af bytes som er skrevet men endnu ikke læst. På denne måde kan man se hvor der skal læses og skrives fra, samt om bufferen er fuld ($cnt = size$) eller tom ($cnt = 0$).

Jeres opgave er at ændre 'pipe.c' så bufferen ligger i delt hukommelse og på den måde kan bruges til at kommunikere mellem processer. Da flere processer kan tilgå pipe'en samtidigt skal I også sørge for gensidig udelukkelse vha. semaforer.

Programmet 'shm.c' er et eksempel på hvordan man anvender delt hukommelse og semaforer, samt hvordan man starter en ny proces. Opgaven er derfor at sætte sig ind i hvordan 'shm.c' virker og bruge dette til at ændre 'pipe.c'. Til dette kan I anvende vedlagte link som beskriver funktionerne og ellers bruge 'man funktion' kommandoen til at få dokumentation.

Følgende er en liste over hvad i skal ændre ved 'pipe.c':

- **pipe_t:** Her skal semafor strukturen tilføjes så den kommer til at ligge i den delte hukommelse.
- **pipe_create:** Her skal den delte hukommelse oprettes og semaforen skal initialiseres. Det antages at 'pipe_create' bliver kaldt før andre processer, som skal bruge pipe'en, bliver startet. Delingen af hukommelse sker altså automatisk når 'fork' efterfølgende kaldes (se 'shm.c').
- **pipe_free:** Her skal den delte hukommelse nedlægges (afhæktes).
- **pipe_write:** Adgang til pipe data skal gøres udelelig (gensidig udelukkelse) vha. af semaforen.
- **pipe_read:** Adgang til pipe data skal gøres udelelig (gensidig udelukkelse) vha. af semaforen.
- **main:** Her skal pipe'en testes som kommunikations medium mellem to processer. Start med at kalde 'pipe_create' før I starter en ny proces med 'fork'. I må selv afgøre hvad de to processer skal gøre for at teste pipe'en. Husk at pipe'en har en fast bufferstørrelse som kunne være én byte. Slut af med at begge processer kalder 'pipe_free'.

Opgavebesvarelsen for dette delspørgsmål skal indeholde:

- Kildeteksten i filen 'pipe.c'.
- Rapport: Beskrivelse af hvordan I har implementere pipe'en samt testen. Testresultater!

G4.2: Segmenterede sidetabeller

I denne opgave skal I arbejde med segmenterede sidetabeller som anvendt af Intel Pentium processoren (se kapitel 8.7 i SGG'05)[1]. I skal forestille jer at en proces på et tidspunkt har de nedenfor angivne "global descriptor table" (GDT), "local descriptor table" (LDT) og sidetabeller (der anvendes 4KB sidepladser). Bemærk at hexadecimale tal angives ved præfikset `0x` hvorimod decimale tal angives uden præfiks.

GDT (g=0)

	base	limit
0	0x801000	0x1000
...		
78	0x803000	0x100
79	0xC01000	0x80
...		
8191	0xA0B000	0x2F00

LDT (g=1)

	base	limit
0	0x802000	0x100
...		
33	0xC00000	0x1000
34	0xC02000	0x2000
...		
8191	0xA0F000	0xFFF

Sidekatalog

	sidetabel
0	...
1	...
2	0x03000
3	0x03400
...	
...	
1023	...

Sidetabeller

Sidetabel på 0x03000

	sideplads
0	0xB2
1	0xCA
2	0X04
3	0x07
...	
1023	0xC1

Sidetabel på 0x03400

	sideplads
0	0x43
1	0x1A
2	0x01
3	0x05
...	
1023	0xC0

Givet ovenstående situation for processen, angiv de linære og de fysiske adresser samt hvordan de beregnes ud fra for følgende logiske adresser (I selectoren refererer *g=0* til GDT og *g=1* til LDT. Beskyttelsesdelen p anvendes ikke, men er tilstede i selectoren):

1. selector = `0x270`, offset = `0x11`
2. selector = `0x278`, offset = `0xF0`
3. selector = `0x10C`, offset = `0xF0F`

I skal vise alle trin i beregningen; afkodning af selector, beregning af linær adresse; beregning af sidekatalog (*eng.:page directory*) og sidetabel (*eng.: page tabel*) samt beregning af den fysiske adresse.

Besvarelsen skal indgå i Jeres rapport.

[1]Bemærk følgende rettelser: s. 305, nederst: 16 KB → 16 K; s. 306, øverst & 3: 8 KB → 8 K; side 308, Fig. 8.23, toppen: logical → linear