

Godkendelsesopgave 3 i “Styresystemer og multiprogrammering”

1 Generelt

Denne ugeopgave skal afleveres senest torsdag den 28. 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 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 instruktør som sidst rettede Jeres opgave.

Jeres rapport skal være i PDF eller ASCII format, for at lette instruktørernes rettearbejde. Opgavenummer og navne på gruppemedlemmer skal fremgå tydelig af første side i rapporten.

Afleveringen 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 tydelig af første side i rapporten.**
- Brug arbejdsgruppe 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.

2 Denne uges tema: Systemkald

Denne uges godkendelsesopgave handler om brug og implementation af systemkald. I lærebogens afsnit 2.3 kan I læse generelt om systemkald og sidst i kapitlet gennemgås implementation af et simpelt systemkald i Linux kernen.

Da der er for mange praktiske problemer med at implementere et systemkald i et rigtigt operativ system som Linux, skal I gøre det i en simpel eksempel kerne. Denne kører i en almindelig UNIX proces ligesom de programmer I tidligere har skrevet.

2.1 Første skridt

Start med at udpakke den udleverede kildetekst. Gå ind i 'src-g3' biblioteket og kør følgende:

```
make
./getpid-benchmark
./user-getpid-benchmark
./user-sum-example
./user-gettimeofday-test
```

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 kompilatoren bliver kaldt. I behøver ikke at tilføje nye filer i denne opgave, så I kan bruge denne kommando hver gang i skal compilere.

Den anden kommando kører programmet 'getpid-benchmark.c', som er et eksempel på hvordan man bruger UNIX funktionen 'gettimeofday' og UNIX systemkaldet 'getpid'. Funktionen 'gettimeofday' giver antallet af sekunder plus mikrosekunder, som er gået siden 00:00:00 UTC, januar 1, 1970. For mere information kør:

```
man gettimeofday
```

Systemkaldet 'getpid' returnerer proces ID'et for den proces det bliver kaldt fra. For mere information kør:

```
man getpid
```

Den tredje kommando kører programmet 'user-getpid-benchmark.c', som er et eksempel på hvordan man bruger 'user_getpid' systemkaldet i eksempel kernen.

Den fjerde kommando kører programmet 'user-sum-example.c', som er et eksempel på hvordan man bruger 'user_sum' systemkaldet i eksempel kernen.

Den sidste kommando kører programmet 'user-gettimeofday-test.c', som er en skabelon til testprogrammet til G3.3.

2.2 G3.1 Benchmark af systemkaldet 'getpid'

I denne opgave skal I lave et program som måler hvor lang tid det tager at lave et systemkald. Systemkaldet 'getpid' skal måles ved at ændre i programmet 'getpid-benchmark.c'.

Målingen skal foretages vha. funktionen 'gettimeofday'. Da 'gettimeofday' ikke med sikkerhed kan måle korte tidsrum præcist, skal I i stedet måle hvor lang tid det tager at udføre mange systemkald (en måling skal mindst tage ét sekund). Herefter kan I udregne et gennemsnit for hvor lang tid ét kald tager. Programmet skal udskrive følgende:

- Antallet n af systemkald som er blevet målt.
- Den samlede måletid t_s for alle systemkald.
- Den gennemsnitlige tid $t = t_s/n$ for ét systemkald.

Opgavebesvarelsen for dette delspørgsmål skal indeholde:

- Programmet i filen 'getpid-benchmark.c'.
- Rapport: Kort beskrivelse af hvad I har gjort og hvordan målingen er lavt. Resultat af måling.

2.3 G3.2. Benchmark af systemkaldet 'user_getpid'

Eksempel kernen bruger et user-level kontekstskifte til at simulere skiftet fra user-mode til kernel-mode. Dette foregår på samme måde som skift mellem user-level threads (lærebog afsnit 4.2 og note). Selve skiftet er implementeret med UNIX funktionen 'swapcontext' i filen 'kernel.c'. For mere information:

```
man swapcontext
```

I denne opgave skal I lave samme måling som i opgave G3.1 på systemkaldet 'user_getpid'. Dette er et systemkald i eksempel kernen. Målingen vil derfor indirekte være en måling af to user-level kontekstskift (user-mode til kernel-mode og tilbage til user-mode).

På samme måde som i opgave G3.1 skal I her tilføje tidsmåling i filen 'user-getpid-benchmark.c'. I kan kopiere størstedelen af koden fra G3.1.

Opgavebesvarelsen for dette delspørgsmål skal indeholde:

- Programmet i filen 'user-getpid-benchmark.c'.
- Rapport: Kort beskrivelse af hvad I har gjort og eventuelle forskelle fra opgave G3.1. Resultat af måling.

2.4 G3.3. Implementation af systemkald

Eksempel kernen bruger systemkaldsmodellen vist i figur 2.3 i lærebogen, hvilket også er den model Linux bruger.

Når brugeren kalder et systemkald, er det i virkligheden en biblioteksfunktion som bliver kaldt (se 'user.c'). Denne funktion sørger for at systemkaldsnummeret og funktionsparametre bliver lagt de rigtige steder, så kernen kan finde dem efter der er skiftet til kernel-mode. Selve kontekstskiftet laves ved at kalde 'kernel_trap' funktionen (se 'kernel.c'), som laver user-level kontekstskiftet ind i kernel-mode. Dette betyder at kernen fortsætter hvor den var sidst den skiftede tilbage til user-mode. Dette er inde funktionen 'syscall_handler' i 'kernel.c'.

'syscall_handler' dekode informationen fra brugeren og kalder det pågældende systemkald. Systemkaldet gemmer returværdi og eventuelle fejlkode så de kan findes på user-level når kernen forlades igen. Det er så op til biblioteksfunktionen at flytte disse værdier rundt så det giver mening for brugerens billede af et systemkald.

2.4.1 Kaldekonvention

User-mode og kernel-mode har ikke samme stak, da de kører i hver sin user-level tråd. Dette gør at et systemkald ikke kan implementeres som en almindelig funktion, da denne bruger stakken til at gemme parametre og returværdi. En virkelig kerne har også sin egen stak da man ønsker en adskillelse af hukommelsen i user-mode og i kernel-mode. Derfor skal der være en kaldekonvention som definerer hvordan data overføres mellem kernen og brugerprocesser (lærebog afsnit 2.3).

Ofte bruges CPU'ens registre til at overføre data, da disse ikke slettes under skiftet. Dette er dog ikke tilfældet i eksempel kernen, da der laves et user-level kontekstskift som jo genskaber tidligere registerværdier.

Derfor bruges nogle virtuelle registre (r0-r7) som bare er globale variable af typen (fra 'kernel.h'):

```
typedef union {
    long i;
    unsigned long ui;
    void *p;
} reg_t;
```

Denne definition betyder at et register kan indeholde et heltal, et positivt heltal eller en pointer, men ikke alle værdier samtidigt.

Disse registre kan så bruges til at overføre data mellem kerne og brugerproces. Der mangler bare en konvention for hvilke registre der bruges til hvad:

register	kald	retur
r0	systemkaldsnummer	returværdi
r1	parameter0	fejlkode
r2-r7	parameter1-6	

Brugen af denne konvention kan studeres i de allerede implementerede systemkald i eksempel kernen (se 'user.c' og 'kernel.c'). Systemkaldsnummeret er et heltal som kernen bruger til at identificere hvilket systemkald der skal kaldes (se syscall.h).

Ved fejl skal et systemkald returnere -1 i r0 og r1 skal indeholde en gyldig fejlkode. Et normalt funktionskald i C kan ikke have to returværdier. Derfor skal fejlkoden kommunikeres op til brugen der jo ikke ved noget om registre. Dette gøres ved at gemme fejlkoden i den globale variable 'errno' (se 'user.c'). Fejlkodeerne er defineret i 'errno.h'.

2.4.2 Opgaven

I skal implementere systemkaldet 'user_gettimeofday' i eksempel kernen. Denne skal stort set være ækvivalent med UNIX funktionen 'gettimeofday' fra de forrige opgaver. I 'user.h' kan i se hvordan funktionen skal se ud for brugeren.

I kernen skal i bruge den rigtige 'gettimeofday' til at få fat i tiden. I programmet 'user-gettimeofday-test.c' kan i se hvordan fejlkode for 'gettimeofday' håndteres. Disse fejlkode skal også håndteres af Jeres systemkald.

Følgende er en liste over hvad i skal gøre ved de forskellige filer for at implementere systemkaldet:

- **syscall.h:** Her skal tilføjes et systemkaldsnummer.
- **errnum.h:** Her skal I tilføje eventuelle ekstra fejlkode.
- **kernel.c:** Selve systemkaldsfunktionen (se de eksisterende systemkald). Derudover skal systemkaldet tilføjes i 'syscall_handler'.
- **user.c:** Implementation af funktionen 'user_gettimeofday' som sørger for at udføre kaldet.

Kig grundigt på de eksisterende systemkald 'user_getpid' og 'user_sum' som hjælp til Jeres egen implementation.

I 'user-gettimeofday-test.c' skal I lave en test af Jeres systemkald ved at sammenligne med tiden fra den rigtige 'gettimeofday'. I skal også teste om kaldet fejler rigtigt. I skal ikke teste for EPERM fejlen, da den er svær at fremprovokere.

Opgavebesvarelsen for dette delspørgsmål skal indeholde:

- All kildeteksten med Jeres ændringer.
- Rapport: Beskrivelse af hvad I har gjort for at implementere systemkaldet og testen. Testresultater.