Godkendelsesopgave 4 i "Styresystemer og multiprogrammering"

Generelt

Denne ugeopgave stilles fredag den 25. februar 2011 og skal afleveres senest mandag den 14. marts 2011 klokken 23:59:59. Den kan løses af grupper på op til 2 personer (3 kan tillades, men frarådes). Besvarelsen af opgaven vil resultere i enten 0, 1/2 eller 1 point. Pointene uddeles efter følgende retningslinjer:

- 0 point: besvarelsen har flere store mangler.
- 1/2 point: besvarelsen opfylder i store træk kravene, men har flere mindre mangler.
- 1 point: besvarelsen opfylder kravene til opgaven med kun få eller ingen mangler.

Det er en betingelse for at gå til eksamen på kurset at man har opnået mindst 4 point i alt.

Besvarelsen skal indleveres elektronisk via kursushjemmesiden (Absalon). 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 bør efternavne på alle gruppemedlemmer indgå i filnavnet - desuden skal opgavenummer fremgå af navnet:

```
efternavn1-efternavn2-efternavn3-G1.<endelse>
```

Jeres aflevering skal være i et format der kan læses på DIKUs systemer uden problemer (således bør formatet f.eks. ikke være MS Word dokumenter). Se i øvrigt "Krav til G-opgaver" siden på kursushjemmesiden under "G-opgaver" menupunktet.

I skal kun aflevere én fuld besvarelse per gruppe. Personen, som afleverer skal markere sine gruppemedlemmer via Absalon.

Om dokumentation af jeres løsning

Afleveringen skal indeholde en kortfattet rapport der dokumenterer hvilke vigtige observationer I har gjort jer, antagelser jeres løsninger afhænger af (som I enten har valgt, eller som er valgt af designerne af BUENOS) og desuden skal I redegøre kortfattet for designbeslutninger i den kode I har implementeret. I skal også huske at kommentere jeres kildetekst så den er let at forstå. Især er det vigtigt at dokumentere hvis I har foretaget ændringer i allerede implementerede dele af BUENOS.

1 Denne uges tema: Implementering af filsystem

Denne (G4) afleveringsopgave handler om implementering af et nyt filsystem. Overordnet er opgaven at implementere en simpel udgave af et filsystem, som benytter en filallokeringstabel (FAT) til at beskrive filer på filsystemet. Helt konkret drejer det sig om at implemetere et filsystem i BUENOS som understøtter en forenklet udgave af Microsoft's FAT32 filsystem. Man kan med fordel læse kapitel 11 i [SGG] for at få en overordnet beskrivelse af hvordan FAT-filsystemer fungerer. Desuden beskriver kapitel 8 i BUENOS guiden hvordan det filsystemer i BUENOS fungerer. For første del af G4, er det relevant at kende til det "virtuelle" filsystem. Desuden er Microsofts egen beskrivelse af filsystemet også værdifuld: `http://msdn.microsoft.com/en-us/windows/hardware/gg463080.aspx`. Microsofts egen beskrivelse indeholder dog også en del oplysninger om forgængerne til FAT32-filsystemet. Derfor kan følgende guide være en hjælp til at fokusere på hvilke dele, der er relevante for FAT32: `http://www.pjrc.com/tech/8051/ide/fat32.html`.

G4.1: Implementering af systemkald til filhåndtering

I en tidligere afleveringsopgave har I implementeret systemkaldene for læsning og skrivning "syscall_read" og "syscall_write" med den begrænsning at det kun skulle være muligt at læse fra "stdin" og skrive til "stdout". I første opgave skal i *udvide* jeres implementation af "syscall_read" og "syscall_write" sådan at det skal være muligt at læse og skrive til vilkårlige filer.

Opgave 1 Implementer alle systemkald som er relateret til håndtering af filer i BUENOS.

Beskrivelse af systemkald Nedenfor er en beskrivelse af de systemkald, som skal implementeres. Beskrivelserne er baseret på afsnit 6.4 samt kapitel 8 af BUENOS guiden.

- "**int** syscall_open(**const char** *filename)" Et kald til "syscall_open" skal klargøre filen identificeret ved "filename" til læsning og skrivning. Der er modsat i Linux ikke mulighed for at åbne en fil med henblik på kun at læse fra den eller kun skrive til den. Resultatet af et kald til "syscall_open" er enten et negativt tal eller et positivt tal større end 2. Et negativt tal angiver at der skete en fejl og et positivt tal skal fortolkes som en repræsentation af den åbnede fil. Det er muligt at åbne den samme fil to gange og resultatet skal da være *forskellige* tal.
- "**int** syscall_close(**int** filehandle)" Et kald til denne funktion skal have den effekt at man ikke efterfølgende kan benytte "filehandle" til at læse eller skrive til filen, som er identificeret derved.
- "**int** syscall_create(**const char*** filename, **int** size)" Opret filen "filename" med en størrelse på "size". Filen er ikke i udgangspunktet at betragte som åben.
- "**int** syscall_delete(**const char** *filename)" Slet filen "filename" fra det filsystem den findes på.
- "syscall_seek(**int** filehandle, **int** offset)" Sæt den nuværende position for den åbne fil repræsenteret af "filehandle" til "offset". Værdien af "offset" skal fortolkes som antallet af bytes fra starten af filen.
- "syscall_read(**int** filehandle, **void** *buffer, **int** length)" Indlæs højest "length" bytes fra den allerede åbnede fil identificeret ved "filehandle". Det indlæste data placeres i "buffer",

som derfor skal pege på allerede allokeret lager. Indlæsning starter fra den nuværende position og efter indlæsning sættes den nye position af "filehandle" til den gamle plus antallet af bytes, som blev indlæst. Hvis "filehandle == STDIN" skal der (som i G2) indlæses fra konsollen.

- "syscall_write(**int** filehandle, **const void*** buffer, **int** length)" Skriv "length" bytes fra "buffer" til den allerede åbnede fil "filehandle". Skrivningen starter fra den nuværende position af filen repræsenteret ved "filehandle" og efter skrivning er positionen flyttet svarende til antallet af bytes, som blev skrevet.

Hovedformålet med opgave 1 er at gøre jer bekendte med hvordan det virtuelle filsystemslag fungerer i BUENOS. I jeres implementation af systemkaldene, skal I derfor "bare" finde de rette funktioner fra `fs/vfs.h` (som alle allerede er implementeret i `fs/vfs.c`) og kalde dem.

G4.2: Implementation af FAT32

FAT32 er et filsystem fra Microsoft som benytter en filallokeringstabel til at vedligeholde oplysninger om filerne på filsystemet. Se afsnit 11.4.2 (side 475) i [SGG] for nærmere beskrivelse af filallokeringstabel-metoden. FAT32 er en udbyggelse af tidligere versioner (FAT12, FAT16) og benytter 32 bits ord til at adressere disk-klynger. En klynge består af et antal disk-sektorer. Kun 28 bit af en klyngeadresse kan dog bruges, så der kan principielt maksimalt være 2^{28} klynger. Hvis klyngestørrelsen er 32 KiB (svarende til 64 disk-sektorer af 512 bytes størrelse), kan der understøttes diske af op til $2^{28} \times 32768$ hvilket svarer til 8 TiB. Den faktiske implementation af FAT32 har dog nogle yderligere begrænsninger i forhold til dette så den reelt understøttede maksimale diskstørrelse en betydeligt mindre end dette.

Opgave 2 Implementér et FAT32 filsystem til BUENOS.

Om at tilføje nye filsystemer til BUENOS Det eneste filsystem, som allerede er implementeret i BUENOS kernen er `tfs`, *trivial filesystem*. Man kan med fordel kigge på koden for dette, for at se hvordan man tilgår diske i BUENOS.

For at tilføje et nyt filsystem til BUENOS, skal man tilføje til listen "filesystems" som tildeles i filen `fs/filesystems.c`.

```
static filesystems_t filesystems[] = {
    {"TFS", &tfs_init},
    { NULL, NULL} /* Last entry must be a NULL pair. */
};
```

Hvert element af "filesystems" er af typen "filesystems_t" som er en struct der indeholder to felter: "**const char*** name" og "fs_t *(*init)(gbd_t *disk)". Feltet "name" sættes til navnet for jeres nye filsystem (f.eks. "SLIM32") og "init" sættes til adressen for en funktion som initialiserer jeres filsystem. Ved opstart af BUENOS, vil hvert filsystem som er defineret i "filesystems" bliver initialiseret ved et kald til filsystemets "init"-funktion (sker i funktionen "filesystems_try_all" defineret i `fs/filsystems.c`.

Funktionen til at initialisere et filsystem skal returnere en værdi af typen "fs_t" som er defineret i `fs/vfs.h`. En "fs_t" struktur består af:

- en peger ("**void*** internal") til noget data, som er specifikt for det pågældende filsystem. For `tfs` sættes "internal" feltet til at pege på en "tfs_t" struktur, som indeholder oplysninger specifikke for `tfs` filsystemet.

- et navn på den "volume" hvor filsystemet er placeret ("volume_name"),
- et antal pegere på funktioner, som benyttes af det virtuelle filsystem til at tilgå filsystemet. For at have en fuldstændig implementation af et filsystem, skal man definere funktionerne
  - "**int** unmount(**struct** fs_struct *fs)"
  - "**int** open(**struct** fs_struct *fs, **char** *filename)"
  - "**int** close(**struct** fs_struct *fs, **int** fileid)"
  - "**int** read(**struct** fs_struct *fs, **int** fileid , **void** *buffer, **int** size, **int** offset)"
  - "**int** write(**struct** fs_struct *fs, **int** fileid , **void** *buffer, **int** size, **int** offset)"
  - "**int** create(**struct** fs_struct *fs, **char** *filename, **int** size)"
  - "**int** remove(**struct** fs_struct *fs, **char** *filename)"
  - "**int** getfree(**struct** fs_struct *fs)"

  Funktionaliteten som hver funktion skal have er beskrevet sammen med "fs_t" typen i `fs/vfs.h`.

Om at oprette et nyt filsystem på en disk I BUENOS er der inkluderet et lille værktøj, `tfstool` som kan benyttes til at oprette et nyt `tfs` filsystem i en fil på systemet, hvor MIPS-simulatoren startes fra (værtssystemet). Filen kan benyttes til simulering af en disk set fra BUENOS-kernens side (gæstsystemet). Der findes et lignende værktøj til at oprette et FAT32 filsystem i en fil (på værtssystemet) til Linux. Det er en del af `dosfstools` pakken. Kommandoen, til at oprette et nyt FAT32 filsystem i en fil `myfat` med en størrelse på omkring 128 MiB:

```
mkdosfs -F 32 -n myfatlabel -s 4 -S 512 -C myfat 131072
```

I kan (i Linux) nu "mount'e" filsystemet med kommandoen:

```
sudo mount -o rw myfat myfat-dir
```

Hvor `myfat` er navnet på filen, som indeholder FAT32 filsystemet og `myfat-dir` er det katalog på værtssystemet, hvor I ønsker at mounte FAT32 filsystemet. Filerne i `myfat` kan nu tilgåes på vanlig vis i Linux (det kan tænkes I får brug for at være '`root`' når I skriver til filsystemet).

Beskrivelse af FAT32

De to primære kilder til beskrivelsen af FAT32 filsystemet (også nævnt i introduktionen) kan findes på følgende adresser:

1. Microsofts egen dokumentation: `http://msdn.microsoft.com/en-us/windows/hardware/gg463080.aspx` og
2. En ingeniørs fortolkning af Microsofts beskrivelse: `http://www.pjrc.com/tech/8051/ide/fat32.html`.

Det er fordelagtigt at læse begge til at afklare uklarheder omkring FAT32 filsystemet. Der er desuden en del implementationer af filsystemet at finde via internettet. Det betragtes ikke som snyd at få inspiration af disse, men det vil formentlig vise sig at være for tidskrævende at tilpasse de tilgængelige FAT32 implementationers typer og funktioner i jeres egen implementation. Det er iøvrigt vigtigt at bemærke at data gemmes i FAT32 filsystemet i *little-endian* format og ikke *big-endian* som på MIPS arkitekturen–se wikipedia artikel om *little endian*: `http://en.wikipedia.org/wiki/Little-endian`.

Begrænsning af opgaven For at gøre opgaven mere overskuelig kan I antage at:

- De fleste filattributter kan ignoreres. De eneste to vigtige er `ATTR_DIRECTORY` som angiver at den pågældende "directory entry" er et katalog og `ATTR_VOLUME_ID` som angiver at der er tale om en speciel fil hvis navn angiver `volume-ID` (se nedenfor).
- Værdien af feltet "volume_name" som skal sættes i "fs_t" strukturen i jeres filsystems-initialiserings funktion skal bestemmes af den særlige `volume-ID` fil og I må antage at denne altid eksisterer.
- I behøver ikke foretage nogen form for check af konsistens af filsystemet. Antag i stedet at filsystemet altid er konsistent.
- I kan antage at alle (korte og lange) navne er i ASCII. Lange navne gemmes i UCS-2 og i kan således nøjes med at benytte de mindst betydende bytes af det lange navn.
- I bebøver ikke benytte (læse/skrive) datofelterne.
- I kan antage at der altid kun er ét filsystem på disk (ingen partitioner).
- I behøver ikke generere korte navne fra lange navne på samme måde som der beskrives i Microsofts dokumentation.