# Godkendelsesopgave 5 i "Styresystemer og multiprogrammering"

## Generelt

Denne ugeopgave stilles fredag den 11. marts 2011 og skal afleveres senest mandag den 21. marts 2011 klokken 23:59:59. Det er *ikke* muligt at gen-aflevere denne godkendelsesopgave (ligesom det hellere ikke vil være det med godkendelsesopgave G6). 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 lagerstyring

Der er i den nuværende version af BUENOS ikke nogen mulighed for at brugerprocesser eller tråde dynamisk allokerer plads fra arbejdeslageret. For at dette skal kunne lade sig gøre skal BUENOS kernen udvides til at foretage (arbejds)lagerstyring. I denne opgave, er det målet at tillade at brugerprocesser og tråde dynamisk skal kunne allokere lager og frigøre det igen. Implementationen af lagerstyringen falder i tre dele:

1. Implementering af funktioner til håndtering af TLB undtagelser.
2. Implementering af allokerings og frigørelsesfunktioner i kernen.
3. Implementering af systemkaldet memlimit.

Del 3 kan først løses når del 1 og 2 er løst, mens del 1 og 2 principielt er uafhængige af hinanden. Del 2 tillader kernetråde at indsætte nye indgange i sidetablen for kernetråde, mens del 1 tillader at kernetråde rent faktisk tilgår dele af lageret, som ikke er placeret i TLB'en. Det kan derfor være fordelagtigt at arbejde på del 1 og 2 samtidigt.

I denne afleveringsopgave er det relevant at læse (dele af) dokumentationen for YAMS (YAMS guiden fremover). Specielt kapitel 6: "Simulated machine". Ligeledes er kapitel 7 (Virtual memory) af BUENOS guiden relevant.

## G5.1: Håndtering af TLB undtagelser

**Opgave 1** Implementer funktionerne til håndtering af TLB undtagelserne. Når en undtagelse "opstår" i BUENOS kaldes en foruddefineret funktion (se "_cswitch_vector_code"; sættes ved opstart af BUENOS i `kernel/interrupt.c` linje 91).

Undtagelser håndteres i BUENOS af funktionerne "kernel_exception_handle" i tilfælde af at undtagelsen opstod fra kerne-tilstand (defineret i `kernel/exception.c`) og "user_exception_handle" i tilfælde af at undtagelsen opstod fra bruger-tilstand (defineret i `proc/exception.c`). De undtagelser, som skal håndteres er (undtagelserne er også beskrevet i YAMS guiden 6.2.1):

`EXCEPTION_TLB` Svarer til "TLB modification" undtagelsen. Denne undtagelse opstår når en tråd har forsøgt at skrive til en "mapped" lageradresse, som er markeret som ikke-skrivbar i TLB'en. Se afsnit 6.3.1 af YAMS guiden samt BUENOS guiden 7.1 for en beskrivelse af hvordan arbejdslageret er struktureret

`EXCEPTION_TLBL` Svarer til "TLB load" undtagelsen. Denne undtagelse opstår når enten den ønskede indgang i TLB'en ikke kunne findes eller indgangen er markeret som ikke-gyldig. Hvis den tråd/process som var årsag til at undtagelsen opståd har en tilknyttet sidetabel (i feltet "pagetable" i strukturen "tread_table_t") skal der søges efter en gyldig indgang i sidetabellen og hvis den findes, skal den gemmes i TLB'en.

`EXCEPTION_TLBS` Svarer til "TLB store" undtagelsen. Denne undtagelse opstår når en tråd/process har forsøgt at skrive til en adresse uden at der kunne findes en indgang i TLB'en, som svarede til den adresse, der skulle skrives til. Den er analog til "TLB load" ovenfor.

Man kan starte med at implementere funktionerne "tlb_modified_exception", "tlb_load_exception" og "tlb_store_exception" i filen `vm/tlb.c`. Ingen af disse tre funktioner tager nogle parametre. For at få oplysninger om den aktuelle undtagelse kan man benytte "_tlb_get_exception_state" der tager adressen på en "tlb_exception_state_t". Resultatet af funktionskaldet er at strukturen udfyldes med værdier, som er relevante for at håndtere den pågældende undtagelse.

### G5.2: Dynamisk allokering af "mapped" lager

I den nuværende BUENOS implementation kan kernetråde dynamisk allokere og frigøre lager ved at benytte "pagepool_get_phys_page" samt "pagepool_free_phys_page".

**Opgave 2** Implementer funktioner sådan at kernetråde kan allokere "mapped" lager. Det skal være muligt for en kernetråd at allokere (og frigøre) lager af vilkårlig størrelse. Det lager en kernetråd allokerer, skal tages fra *kernel mapped* segmentet (0xE0000000 til 0xFFFFFFFF) som har en størrelse på 512 MB. En fordelagtig tilgang er at lade kernetråde vedligholde en kernehob og tilfredsstille forespørgsler på lager fra kernehoben. Eksempler på hob-baseret lageradministration kan findes i [K&R] samt i det udleverede test-bibliotek `lib.c` under "Undervisningsmateriale/Buenos testkode".

Generelt er effekten af at en kernetråd allokerer mapped lager at nye indgange oprettes i trådens sidetabel ("pagetable" feltet i "thread_table_t" strukturen) sådan at den virtulle (eller logiske) adresse afbildes over i en fysisk adresse. Funktionen "vm_map" (defineret i `vm/vm.c`) kan bruges til at indsætte en indgang i en "pagetable_t" struktur. Ligeledes skal indgangen fjernes igen når det tilsvarende lager frigøres. Det kan være fordelagtigt at forsøge at minimere fragmentering af lageret som resultat af mange allokeringer og frigørelser.

Den nuværende implementation af BUENOS kun tillader "PAGETABLE_ENTRIES" (defineret til 340 i `vm/pagetable.h`) indgange. Det er *tilladeligt* at forbedre denne begrænsning, men en sådan forbedring er ikke formålet med denne godkendelsesopgave. Det maksimale antal bytes en kernetråd derfor kan allokere er derfor $340 * 4096$ bytes.

### G5.3: Bibliotek til brugerprocesser

**Opgave 3** Implementer et lille bibliotek som tillader at brugerprocesser kan allokere og frigive hukommelse. Biblieteket skal indeholde følgende to funktioner:

- "**void** *malloc(**int** size)"
- "**void** free(**void** *ptr)"

Et kald til "malloc(s)" allokerer "s" bytes til brugerprocessen/tråden som fortager kaldet og returnerer en peger, som peger på startadressen for det allokerede lager. Brugerprocesser/tråde skal allokere lager fra *user mapped* segmentet. Et kald til free(ptr) skal frigøre det lager, som tidligere er blevet allokeret til den kaldende process. Hvis man forsøge at frigøre lager, som allerede er frigjort er opførslen af "free(ptr)" udefineret. Hvis "ptr" er "NULL" har "free(ptr)" ingen effekt. For at implementere de to funktioner er det meningen at man skal benytte "memlimit" systemkaldet (som således også skal implementeres). En beskrivelse af "memlimit" findes i BUENOS guiden afsnit 6.4.2. Bemærk at der er mange lighedspunkter mellem opgave 2 og opgave 3.