Styresystemer og multiprogrammering (OSM) - G2

Dennis Bøgelund Olesen - 060593 - cwb759
Emil Lagoni - 051290 - frs575
Erik David Allin - 171292 - smt504

24. Februar 2013

Task 1

I denne opgave er der redigeret i filerne: `proc/process.h`, `proc/process.c` og `init/main.c`

Da det kun er spawn der genererer nye processorer, og vi via vores debugging i `proc/process.c` tester dette (via de mange udkommenterede kprintf) skulle der meget gerne kun være to processer kørende, men alligevel gives der fejlen *"TLB - load not handled yet"*.
Vi har af denne grund ikke kunnet teste særlig grundigt, men af debuggingen se at Spawn er istand til at starte programmet.

Nedenfor beskrives funktionerne Init, spawn, join og finish:

Init

Kører alle processer igennem hvor antal = `PROCESS_MAX_PROCESSES` og sætter deres state til FREE.
Herudover gemmes processernes ID (pid) samt deres ID i den thread/forældre de kører i.

Spawn

Først findes en ledig plads blandt processerne via funktionen `findFreeBlock`, der enten returnerer en ledig process i process tabellen eller meddeler, at der ikke er nogle ledige.
Herefter gemmes exec i variablen executable, så start senere kan tilgå den.
Der sættes forældrens ID og herefter ændres processens state til RUNNING.
Herefter køres processen via process start i en ny tråd, der laves i newThread.
Til sidst returneres process ID'et på den nye process.

Join

I join valgte vi at følge proceduren fra Buenos Roadmap delkapitel 5.2.1 for hvordan processer ligges i SleepQ'en.

Dette har vi dog ikke kunne teste ordentligt pga. fejlen, der også tidligere er beskrevet.

Finish

Dræber en konkret tråd og sætter den process.state til ZOMBIE.
Gemmer returværdien (argumentet retval) i process tablen/kontrolblokken.

process.h

For at have tilstande har vi lavet en `enum` til at repræsentere de tilstande en proces kan have. Dertil selve datastrukturen for kontrolblokken, som indeholder et procesid, PID, forældrens PID for at kunne tjekke om processen har en forældre, og tilstanden for processen.

Task 2

I denne opgave er der redigeret i filerne: `proc/syscall.h` og `proc/syscall.c`

Her er systemkaldende EXEC, EXIT og JOIN implementeret.

EXEC: Kører filen med det pågældende process id, som findes via at finde pid'et i process.tabellen hvorefter der køres spawn på denne.
EXIT: Terminerer en pågældende process. Herudover tjekker den om returværdien er positiv og hvis den er dette køres finish på denne. Til sidst returneres retval.
Hvis retval er negativ returneres -1, da der så er tale om en fejl.
JOIN: Kører funktionen join på en 'børne-process' når den har ventet på at den er færdig med at køre.