LINKÖPINGS UNIVERSITET

TDIU16: Process- och operativsystemprogrammering

Systemanropen `halt` och `exit`

Klas Arvidsson, Daniel Thorén, Filip Strömbäck

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2018

2020-03-06

1 Mål

Systemanrop är en säkerhetsmekanism som låter OS kontrollera och hantera tillgång till hårdvaran i en dator. Detta är mycket viktigt eftersom processer (användarprogram) annars kan förstöra för både OS och andra processer. För mer information om systemanrop i Pintos se Pintos-Wiki under rubriken ”Systemanrop i Pintos”.

Målet med laborationen är att implementera två grundläggande systemanrop, `halt` och `exit`, för att se hur systemanropsmekanismen fungerar i Pintos.

2 Testprogram

Ett testprogram som använder systemanropet `halt` finns i `examples/halt.c`. Se ”Att köra Pintos” i Pintos-Wiki för detaljer om hur du kör program i Pintos.

För att testa `exit` modifieras lämpligtvis testprogrammet för `halt` så att det i stället anropar `exit` med lämplig parameter.

3 Uppgift

Implementera systemanropen `halt` och `exit` genom att lägga till nödvändig kod i interrupt-hanteraren för systemanrop (`userprog/syscall.c`).

Systemanropet `halt` ska stänga av datorn (den virtuella dator som Pintos körs i). Pintos har en funktion `power_off` implementerad i `threads/init.c` som gör detta. Inkludera `threads/init.h` för att få tillgång till denna.

Systemanropet `exit` ska avsluta det program som anropade `exit`. Notera att `exit` tar en parameter. I framtida laborationer kommer vi använda denna, men för närvarande räcker det att parametern skrivs ut, så att man kan se att allt gick som det skulle. Se filen `threads/thread.h` för lämpliga funktioner för detta.

Testa din implementation. Det kan vara bra att lägga till några debug-utskrifter så du ser mer av vad som händer.

4 Beskrivning av uppgiftens systemanrop

Systemanropen `halt` och `exit` är deklarerade i `lib/user/syscall.h` och kan sedan användas av användarprogrammen:

- `void halt(void) NO_RETURN;`

  Stänger av *datorn* (emulatorn) med omedelbar effekt. Processen kommer inte fortsätta sin exekvering eftersom datorn stängs av.

- `void exit(int status) NO_RETURN;`

  Stänger av den process som anropade den. Processen kommer inte fortsätta sin exekevering efter detta eftersom den stängdes av.

  Om processen lyckades med det den programmerades för (processpecifikt) anropas `exit` traditionellt med `status` satt till 0. Misslyckas något väljer programmeraren traditionellt något annat värde på

`status` (processspecifikt) för att kunna särskilja olika fel. Värdet på `status` är intressant främst för den som startade processen, och en mekanism för att ta reda på `status` kommer implementeras senare. Normalt skickas returvärdet från `main` som `status`, se ”Skapa stack till `main`”.