Deluppgift 13 Systemanropet exit

Förberedelse

Repetera informationen given i föregående uppgifter. Alla systemanrop går till på samma sätt som i uppgift 12. Deluppgift 8 visar på vad somhänder när `main` returnerar och skickar returvärdet vidare till systemanropet exit. Även om vi inte löser uppgift 10 ännu så beskrivs där viktiga funktioner för tråd- och processhantering. Läs alltså uppgift 10 i förväg utan att lösa uppgiften. (Det är inte riktigt aktuellt att fundera på ännu, men deluppgift 3 exemplifierar vad som händer med exit-status när en process i UNIX avslutar, och hur (och när) det går att komma åt efteråt med hjälp av wait.) Fundera igenom följande frågor innan du startar:

1) Vilka funktioner är involverade i att avsluta en tråd? En process?
2) Vilken uppgift har respektive funktion?
3) I vilken ordning exekveras de?
4) I vilken fil är parametrarna till varje systemanrop specificerad?
5) Var i Pintos minne kan du hitta parametrarna?
6) Vad skall operativsystemet göra med parametern till exit (exit status)?
7) Vad händer när main returnerar?

Beskrivning av uppgiftens systemanrop

Systemanropet `exit` är deklarerat i `lib/user/syscall.h` och används av användar-programmen därefter:

```
void exit(int status) NO_RETURN;
```

Avsluta processen. Processen skall inte fortsätta sin exekvering. Om processen lyckades med det den programmerades för (processspecifikt) anropas `exit` traditionellt med `status` satt till 0. Misslyckas något väljer programmeraren av processen traditionellt något annat värde på `status` (processspecifikt) för att kunna särskilja olika fel. Värdet på `status` är intressant främst för den som startade processen (förälder), och en mekanism för föräldern att ta reda på `status` skall finnas (senare deluppgift). Normalt skickas returvärdet från `main` in som `status`, se uppgift 8.

Uppgift

Komplettera systemanropshanteraren med systemanropet för `exit`, `SYS_EXIT`. Systemanropet `exit` avslutar en process. I denna version kan du ignorera värdet som ges som argument (processens exit-`status`) eftersom nödvändig infrastruktur för att lagra detta inte är implementerad. Det räcker att göra en debug-utskrift av `status` så du kan se värdet i terminalen.

Testa din implementation med hjälp av `sumargv`-programmet. Modifiera även `halt`-programmet så det anropar `exit` innan `halt`. Har du gjort rätt skall `halt` inte längre anropas, eftersom programmet avslutar med `exit` innan dess. Om PintOS ändå stänger av har du gjort något fel.

För att komma åt den tråd som exekverar finns en funktion i `thread/thread.h` som returnerar en pekare till den tråd som kör. Funktionen heter `thread_current` (se uppgift 10).