LINKÖPINGS UNIVERSITET

TDIU16: Process- och operativsystemprogrammering

Argument till `main` i Pintos

Klas Arvidsson, Daniel Thorén, Filip Strömbäck

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2018

2020-03-06

1 Mål

Målet med denna uppgiften är att använda den kod som gjordes i labben ”Stack till `main`” för att se till så att kommandoradsparametrar kan skickas till program i Pintos. Detta gör att vi dels kan skicka parametrar till den första processen som startas från kommandot som startar Pintos, och dels så att program kan skicka parametrar till barnprocesser som startas med systemanropet `exec` (som vi implementerar senare).

2 Uppgift

Exempelprogrammet `sumargv` från föregående uppgifter kan inte exekvera korrekt uan en stack som har `argc` (antalet ord på kommandoraden) och `argv` (en array med pekare till orden) korrekt uppsatta. Du skall nu implementera möjlighet att ge argument till main. Du skall inte behöva göra mer än att klistra in relevant kod och anropa funktionen från uppgift 8 med rätt parametrar och sedan testa din lösning. Utför följande steg:

1. Radera raden **HACK if_.esp -= 12;** som finns i `userprog/process.c`. `if_.esp` är processens initiala stackpekare. I denna uppgift löser vi problemet på rätt sätt, och nämnda hack behövs inte mer. Nu behöver du inte låtsas att det finns data på stacken längre, din funktion från ”Stack till `main`” kommer förbereda stacken med korrekt data.
2. Kör ett användarprogram och studera vad som händer? Varför blir det så? Varför subtraheras just 12 och inte t.ex. 8? Studera `_start`-funktionen i `src/lib/user/entry.c` och fundera på vilka adresser på stacken som läses.
3. Kopiera över relevant kod från ”Stack till `main`” till `userprog/process.c`. Lägg in ett anrop till den nya funktionen på lämplig plats, med lämpliga parametrar. Studera vad du gjorde i den tidigare uppgiften och vilka funktioner som exekveras i vilken ordning när `sumargv` startar (titta på debugutskrifterna och koden) för att komma fram till var stacken skall initieras.
4. Funktionen `strtok_r` finns i Pintos. För att få tillgång till den behöver du inkludera `lib/string.h` från `userprog/process.c`. Den fungerar identiskt med C-bibliotekets version.
5. Testa din lösning noga med programmet `sumargv`. Några exempel:

```
pintos -p ../examples/sumargv -a sumargv   \
       -v -k --fs-disk=2 -- -f -q run 'sumargv 1 2 3 4'

pintos -p ../examples/sumargv -a sumargv   \
       -v -k --fs-disk=2 -- -f -q run 'sumargv 1000 200 30 4'

pintos -p ../examples/sumargv -a sumargv   \
       -v -k --fs-disk=2 -- -f -q run 'sumargv 36 64 100'

pintos -p ../examples/sumargv -a sumargv   \
       -v -k --fs-disk=2 -- -f -q run 'sumargv 2 3 5 7 11'

pintos -p ../examples/sumargv -a sumargv   \
       -v -k --fs-disk=2 -- -f -q run 'sumargv'

pintos -p ../examples/sumargv -a sumargv   \
       -v -k --fs-disk=2 -- -f -q run 'sumargv 0'
```

Du skall nu kunna mata in en rad tal som argument på kommandoraden, och `sumargv` skall beräkna rätt summa. Vilken utskrift visar summan? Vad kan det andra värdet på stacken vara nu igen?

6. Städa sedan upp i de spårutskrifter din implementation gör, så att de inte gör det svårare att hitta relevanta spårutskrifter för framtida laborationer.