LINKÖPINGS UNIVERSITET

TDIU16: Process- och operativsystemprogrammering

Den dubbellänkade listan i Pintos

Klas Arvidsson, Daniel Thorén, Filip Strömbäck

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2018

2020-03-06

1 Mål

Målet med denna uppgiften är att introducera den länkade listan som finns tillgänglig i Pintos. Den är i princip som en `std::list` i C++, men i och med att C saknar templates fungerar den ganska annorlunda, vilket kan vara förvirrande till en början.

OBS! Var försiktig med att klippa och klistra kod. En del tecken konverteras fel. Oftast olika former av citattecken (" och " och ' och '). De kan se rätt ut i emacs men kompilatorn förstår ej.

I Pintos finns en länkad lista (src/lib/kernel/list.h). Det är bra att veta hur den kan användas.

I uppgiften nedan skall du använda denna lista till att implementera Erathostenes primtalssåll. Kod för att implementera sållet finns given i filen `src/standalone/labx3/` i Pintos-repositoryt, så du kan koncentrera dig på att skapa den struct som behövs, hur du använder `malloc` och `free` (`man -s3 malloc`), samt hur listan fungerar. Det är bättre att göra fel i ett litet program utanför Pintos än att göra fel i Pintos. I och med att detta program implementeras utanför Pintos kan `valgrind` användas för att enkelt hitta minnesfel.

För mer information på hur listan i pintos fungerar se Pintos-Wiki.

2 Kompilering av flera filer

När ditt program är uppdelat i flera moduler (flera filer) så måste alla implementationsfiler kompileras för att hela programmet skall kunna skapas. Observera att headerfiler `aldrig` skall kompileras. (Headerfiler kommer med vid kompileringen av implementationsfilen eftersom de inkluderas. Om du kompilerar en headerfil av misstag kommer kommer du så småningom att få problem om du inte är noga med att ta bort alla filer av typen `*.gch` som skapas.)

I denna uppgift behöver du kompilera två filer, listan och huvudprogrammet. Antag att filerna heter list.c och use-list.c, då behövs följande kommando för att förkompilera listan (eftersom du inte skall ändra listan räcker det att kompilera den en gång):

```
gcc -Wall -Wextra -std=c99 -pedantic -g -c ../../lib/kernel/list.c
```

Sedan kompileras hela programmet. Här nyttjas den förkompilerade objektfilen `list.o`:

```
gcc -Wall -Wextra -std=c99 -pedantic -g list.o use-list.c
```

Det går även att kompilera allt på en gång (ett kommando), eller att förkompilera även huvudprogrammet. Tänk till själv så kommer du fram till hur. Kompilatorn använder per default "standard" *gnu99* vilket är ISO C99 standarden med några tillägg. I kompileringsraderna ovan instruerar vi kompilatorn att använda strikt ISO C99 standard.

3 Uppgift

Algoritmen du skall implementera (för att göra något med listan) är given nedan och finns även som fil via kurshemsidan. Fokus i denna uppgift ligger på att öva listhantering, pekarhantering och minneshantering. Ersätt kommentarerna i filen med korrekt listkod.

```
/* TODO: skapa och initera en lista */
for (i = 2; i < N; ++i)
{
  /* TODO: sätt in talet 'i' sorterat listan */
}

for (i = 2; i < N; ++i)
{
  for (j = i*2; j < N; j += i)
  {
    struct list_elem *e;

    /* TODO: Stega igenom listan och ta bort varje element
       som är jämt delbart med j. Glöm inte avallokera minne.
     */
  }
}

/* TODO: skriv ut alla tal i listan */

/* TODO: töm listan och avallokera alla element */
```