LINKÖPINGS UNIVERSITET

TDIU16: Process- och operativsystemprogrammering

Grunder i C

Klas Arvidsson, Daniel Thorén, Filip Strömbäck

INSTITUTIONEN FÖR DATAVETENSKAP

Vårterminen 2023

2020-03-06

1 Mål

Målet med denna frivilliga laboration är att öva på C i en miljö som är lite vänligare än inuti Pintos. Gör man fel inuti Pintos kan mycket konstigt hända, gör man fel här är det lite större sannolikhet att felet syns tidigt, exempelvis med hjälp av `valgrind`.

2 Saker som kan vara bra att känna till

För att kompilera använder du `gcc` med följande flaggor:

```
gcc -m32 -Wall -Wextra -std=c99 -pedantic -g assignment1.c
```

Programmet kommer heta *a.out* om du inte döper om det med flaggan `-o`

3 Uppgift

I denna uppgift kommer du att använda flera av de vanligaste satserna, typerna och operatorerna. Du skall skriva ett program som utför följande:

1. Deklarerar ett heltal, ett flyttal, ett tecken, en text-sträng och en array med 5 heltal. Initiera alla variabler direkt vid deklaration. En text-sträng, eller C-sträng, är en pekare till en sekvens av tecken. Det är inte givet på förhand hur lång sekvensen är. Istället är det givet att sekvensen är minst ett tecken, där det sista tecknet alltid måste vara det så kallade noll-tecknet (`'\0'`). En tom sträng innehåller alltså endast ett nolltecken. Nolltecknet brukar inte räknas med när längden på en sträng beräknas. En tom sträng har alltså längden 0, även fast sekvensen för att lagra den är ett tecken lång.

2. Använd funktionen printf från standardbiblioteket (`man -s3 printf`) för att skriva ut namn, adress, och värde på varje variabel. Adresserna skall skrivas ut hexadecimalt, alltid med åtta siffror (exempelvis `0x00010a08`). Skriv ut innehållet i array- och pekarvariablerna med formatet `\%p` (dvs. skriv ut adressen till första värdet i arrayen, pekaren som anger var sekvensen värden startar, inte hela arrayen). Strängen skriver du ut med formatet `%s` som hanterar textsträngar speciellt. *Detta har direkt relevans för debugutskrifter i Pintos.*

   Fundera på vad följande printf-satser kommer att skriva ut:

   ```
   int i = 2;
   int array[5] = { 1, 2, 3, 4, 5 };
   const char *string = "Hello";
   printf("%p\n", array);
   printf("%p\n", &array);
   printf("%d\n", array[i]);
   printf("%p\n", &array[i]);
   printf("%p\n", string);
   printf("%s\n", string);
   ```

3. Använd en while-loop och pekarstegring för att skriva ut varje tecken i strängen. Först skall adressen där tecknet är lagrat skrivas ut, sedan själva tecknet, ett tecken per rad. Med pekarstegring avses att du varje varv i loopen ökar en startpekare att peka till nästa position i en array. Innehållet på adressen hämtas som vanligt med `*` operatorn.

4. Använd en for-loop och indexering för att skriva ut varje tecken i heltals-arrayen på samma format som tecken arrayen. Med indexering avses att du använder indexoperatorn `[]`.

5. Skapa en funktion, du kan kalla den getline, som tar emot en array av tecken samt ett heltal som anger arrayens storlek. Funktionen skall läsa in en rad med tecken från tangentbordet med funktionen getchar

från standardbiblioteket (`man -s3 getchar`). Rader avslutas med tecknet för nyrad (`\n`). Tecknen skall lagras i arrayen. Om arrayen är för liten avbryter du inläsningen av raden i förtid. Avsluta i alla lägen arrayen med ett nolltecken.

6. Använd funktionen från föregående steg till att läsa in en sträng, och använd funktionen atoi från standardbiblioteket (`man -s3 atoi`) för att konvertera den inlästa strängen till ett heltal. Avrunda sedan heltalet till närmaste högre heltal som är jämnt delbart med fyra. Modulusoperatorn (`%`) för att få resten vid division kan vara användbar, men tänk till så du inte avrundar i onödan eller avrundar nedåt. *Avrundningen har direkt relevans för senare stackhantering.*

Du kan lägga ditt program i exempelvis `standalone/labx1`.