Linux som utvecklingsmiljö
Övning 6 - Bibliotek

Axel Vidmark 8710171417 axel.vidmark@gmail.com

11 maj 2017

Del 1

Uppgift:

Redovisa en beskrivning av det egna biblioteket och applikationen vad gäller:

1. Funktion och användning.
2. En algoritmbeskrivning.
3. Hur du kompilerat och testat det i ett eget program. Beskriv vilka kommandon du använder för att kompilera biblioteket och hur du sedan använder biblioteket i ditt huvudprogram.
4. Beskriv de växlar du använt till kommandona ovan och vad var och en av dessa har för funktion.

Biliotek 2, libpower.so
Ström som passerar ett motstånd värmer upp motståndet med en viss effekt (P). Effekten kan beräknas med hjälp av spänningen och ström eller spänning och motståndsvärdet enligt dessa formler:

- $P = U * I$ (Spänning gånger strömmen)
- $P = U^2/R$ (Spänning i kvadrat delat i resistansen)

Skriv ett bibliotek, libpower.so, med funktioner för att beräkna den totala effektutvecklingen i en krets med en spänningskälla kopplad i serie med en en resistans:

```
float calc_power_r(float volt, float resistance);

float calc_power_i(float volt, float current);
```

Svar:

Biblioteket libpower innehåller två funktioner för att räkna ut värmeutvecklingen i ett motstånd med hjälp att spänning och ström eller spänning och motsånd. Algoritmen för de två funktionerna är beskriven med pseudokod nedan:

```
float calc_power_r(float volt, float resistance):
    p = powf(volt, 2.)/resistance
float calc_power_i(float volt, float current):
    p = volt * current;
```

Testprogrammet `testpower` skriver ut

```
Effekten när spänningen är 50 V och resistansen är 300 ohm är 8.333333
Den borde vara 8.3
Effekten när spänningen är 50 V och strömmen är 0.2 A är 10.000000
Den borde vara 10
```

Där outputten är generad med hjälp av biblioteksfunktionerna.

För att testa mitt eget program så skapade jag en enkel makefile. För testning lokalt statiskt länkade bibliotek använder jag först

```
gcc -c -fPIC libpower.c -lm
```

för att generera biblioteket. -c sätter ihop filerna utan någn länkning, -fPIC generar en fil som är "oberoende av position" dvs den kan exekveras utan ändras för att ta hänsyn till var den laddades in i minnet, medan -lm säger åt den att hämta mattebiblioteket som behövdes för powf.

Sedan länkar jag det till min testfil genom:

```
ar rcs libpower.a libpower.o
gcc -static test_libpower.c -o testpower -L. -lpower -lm
```

ar skapar ett arkiv libpower.a för att länkas in i programmet. rcs innebär sätt in, skapa, skriv till arkivet. Sedan länkar vi arkivet statiskt (-static), med bibliotek i nuvarande dir (-L.) och inkluderar mattepaketet.

Del 2

Uppgift:

Skriv en Makefile med dessa regler:

- lib, för att bygga enbart biblioteket.
- all, För att bygga både programmet och biblioteken där biblioteken läggs i en egen katalog, lib, under den man är jus nu, tex `/home/bl/electro/lib/`. Här ska programmet länkas för att använda de lokala biblioteken. OBS! Ni får inte temporärt ändra libsökvägarna i `LD_LIBRARY_PATH`! install. Här kopierar du både programmet och biblioteken till lämpliga kataloger (tex `/usr/bin/` och `/usr/lib/`) och länkar så att programmet använder de publika biblioteken.
- Redovisa en beskrivning av hur du länkat in alla 3 bibliotek och använt det i ett huvudprogram enligt ovan. Beskriv också vilka du jobbat med.

Svar:

Först så skapar vi alla bibliotek och lägger dem i en undermapp.

```
lib: lib1 lib2 lib3

lib1: src/lib1/libresistance.c src/lib1/libresistance.h
    gcc -c -fPIC src/lib1/libresistance.c -o lib/libresistance.o
    gcc -shared -o lib/libresistance.so lib/libresistance.o

lib2: src/lib2/libpower.c src/lib2/libpower.h
    gcc -c -fPIC src/lib2/libpower.c -o lib/libpower.o -lm
    gcc -shared -o lib/libpower.so lib/libpower.o -lm

lib3: src/lib3/libcomponent.c src/lib3/libcomponent.h
    gcc -c -fPIC src/lib3/libcomponent.c -o lib/libcomponent.o -lm -std=c99
    gcc -shared -o lib/libcomponent.so lib/libcomponent.o -lm -std=c99
```

Här är alla växlar detsamma som i del 1, med skillnaden att vi har ett extra steg för att skapa delade bibliotek, -shared skapar ett bibliotek för delad länkning. Här var vi även tvungna att specifiera vilken c-standard kompilatorn skulle jobba med då en av inte kunde kompilera pga initering av variabel i en for-loop (c99)

För att testa programmet och bygga det så att det använder de lokala biblioteken så använder vi sen följande del av makefilen.

```
all: lib src/electrotest.c
    ar rcs lib/libresistance.a lib/libresistance.o
    ar rcs lib/libpower.a lib/libpower.o
    ar rcs lib/libcomponent.a lib/libcomponent.o

    electrotest:
$(CC)  -o electrotest src/electrotest.c -Wl,-rpath,$(CURDIR)/lib -L$(CURDIR)/lib -lpower -lresistance -
```

`-Wl,-rpath,$(CURDIR)/lib` lägger till den lokala mappen med biblitek till körsökvägen. `-L$(CURDIR)/lib` justerar sökvägen för bibliotek

Make `electrotest_static` testar statisk länkning:

```
electrotest_static: all
$(CC) -static src/electrotest.c -Llib -lpower -lresistance -lcomponent -o electrotest_static
-lm -std=c99
```

Här skapar vi på samma sätt som i uppgift 1 lokala arkiv som vi sedan länkar in statiskt i programmet.

Slutligen för att installera biblioteken och programmet så det använder de delade biblioteken använder vi följande.

```
install: installlib electrotest_global
install electrotest_global /usr/local/bin/electrotest
installlib: lib1 lib2 lib3
install lib/libpower.so /usr/lib
```

```
install lib/libresistance.so /usr/lib
install lib/libcomponent.so /usr/lib

electrotest_global:
$(CC) $(CFLAGS) -o electrotest_global src/electrotest.c ...
... -lresistance -lpower -lcomponent -rpath -lm -std=c99
```

(Obs: ingen sökväg till lokal bibliotek här.)

Här installera vi först de tre biblioteken och länkar sen in dem från /usr/lib. Kör vi nu ldd på det program vi installerade och exekverar det installerade electrotest får vi följande output.

```
$ ldd /usr/local/bin/electrotest
linux-vdso.so.1 => (0x00007ffc9b1da000)
libresistance.so => /usr/lib/libresistance.so (0x00007fc9e3a45000)
libpower.so => /usr/lib/libpower.so (0x00007fc9e3843000)
libcomponent.so => /usr/lib/libcomponent.so (0x00007fc9e3641000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fc9e3279000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007fc9e2f73000)
/lib64/ld-linux-x86-64.so.2 (0x00007fc9e3c47000)


axvi@axvi-VirtualBox:~/linum6$ electrotest
Ange spänningskälla i V: 50
Ange koppling[S | P]: S
Antal komponenter: 3
Komponent 1 i ohm: 300
Komponent 2 i ohm: 500
Komponent 3 i ohm: 598
Ersättningsresistans: 1398.000000
Effekt: 1.788269
Ersättningsresistanser i E12-serien kopplade i serie:
1200
180
18
```

De olika biblioteken länkades in i huvudprogrammet med hjälp av deras headerfiler. I den här uppgiften jobbade jag med:

- Antero Metso antero.metso@gmail.com
- Samuel Backteman samuel.backteman@gmail.com

Del 3

Uppgift:

Skriv en kort beskrivning av hur samarbetet gått samt en reflektion över vad som krävs för att ett sådant här arbete ska fungera även i större skala med betydligt större skara utvecklare och större kodbas.

Svar:

Samarbetet gick bra, mycket tack vare att vi alla verkade ha liknande förväntningar. Mail fungerar hjälpligt som kommunikationsväg medan det var hjälpsamt att vi innan vi började pratades vid över voip för att stämma av förväntningar och hur vi uppfattat uppgiften.

Tack vare att vi tidigt satte upp en gemensam git-repository som vi alla jobbade mot kunde vi lätt följa vad de andra i gruppen gjorde. Sedan kunde vi mailledes hålla varandra uppdaterade med längre förklaringar av commits vid behov.

Ska man jobba i ännu större grupper eller med större projekt så är det några grunder jag tycker är viktiga:

- Tydliga förväntningar (deadlines, kvalitet etc.)
- Tydlig ansvarsuppdelning
- Tydlig arbetsprocess (i vårt fall: gör ändringar och commita så att alla kan bedöma/kika allt eftersom arbetet fortskrider)
- Tydlig kodstruktur (i vårt fall: iom att den första delen var individuell så ser lösningarna lite olika ut, men för den gemensamma delen så bestämdes det efter den som började, inte helt optimalt men fungerade i det här lilla projektet)