Kom-igång-hjälp till debuggern gdb

Kompilera för debugging

Om man ska använda debuggern `gdb` för att följa programexekveringen och för att felsöka behövs ett särskilt kompileringsdirektiv, `-g`, för att generera symbolinformation till debuggern.

Anta att de programdelar som ska länkas ihop inför körning heter `Mprov.s` och `bibliotek.s`. Då genereras en körbar fil (kompilerar och länkar) för debugging som heter `test` med:

```
gcc -g -o test Mprov.s bibliotek.s
```

Starta och stoppa exekvering

När programmet är kompilerat och länkat kan debuggern aktiveras med:

```
gdb test
```

För att köra programmet skriver man `run` (`r` kort), men att göra bara det är inte mycket mening eftersom programmet nu uppför sig exakt som det gör direkt vid Linux-prompten. Vitsen med att använda debuggern är att man nu kan sätta *brytpunkter*, så att programexekveringen *stannar* på väl valda ställen. Det är lätt att sätta brytpunkter med kommandot `break` (`b` kort) vid etiketter (eng. labels) i koden. Om det finns en etikett i koden där funktionen `inimage` börjar, kan en brytpunkt sättas där med kommandot:

```
b inimage
```

Om en sådan brytpunkt skapats kan programmet köras med kommandot `run`. Nu stannar programmet vid den angivna etiketten, och man kan titta på vad som finns i register och minne om man vill.

Om programmet kört till en brytpunkt och man vill titta närmare på vad som händer steg för steg i programmet kan stegvis exekvering användas. Kommandot `step` (`s` kort) kör en instruktion och om en siffra anges efter kommandot körs så många instruktioner som siffran anger innan programmet avbryts igen. Det finns en variant på kommandot som heter `next` (`n` kort). Skillnaden är att `next` inte går in i funktioner instruktionsvis, utan funktionen ses som en instruktion när programmet stegas igenom.

Titta på registerinnehåll

Om man vill titta på innehållet i cpu-registren kan kommandot `info` användas. Registerinnehållen (utom stackpekare m fl. som bara innehåller adresser) presenteras då både på hexadecimal och decimal form:

```
info registers
```

Vill man bara titta på innehållet i ett specifikt register kan man använda kommandot `print`. Där kan man också ange en formatkod för vilket format man vill se innehållet på (decimal är default). Koden är `d` för decimal form, `x` för hexadecimal, `t` för binär form och `c` för ASCII-tecken. Exempel för att visa innehållet i register `rax` på hexadecimal form:

```
print/x $rax
```

Om man frekvent är intresserad av innehållet i ett specifikt register kan man använda kommandot display. Det register man angivit kommer då att visa sitt innehåll varje gång programmet avbryts. Exempel:

```
display $rax
```

Man kan upprepa detta för flera register, vilka då hamnar i en numrerad lista. Vill man sluta visa registret skriver man bara `undisplay`. Om man vill ta bort ett specifikt register anger man dess listnummer. Utan argument kan man ta bort alla samtidigt.

Titta på minnesinnehåll

Minnesinnehåll kan undersökas med kommandot `x`. Här kan man också ange hur många byte man vill se från och med en viss adress och på vilket format. Anta att en etikett `inbuff` definierats för en buffert i minnet. Om man vill titta på 12 byte framåt i minnet därifrån som ASCII-tecken bytevis kan ett kommando för det skrivas som:

```
x/12cb &inbuff
```

`12` är hur många enheter i minnet som ska granskas c betyder att innehållet ska tolkas som ASCII-tecken (se formatkoderna under rubriken ”Titta på registerinnehåll”) och `b` betyder att de `12` enheterna är av typen byte. Andra valmöjligheter där är `h` (16-bits ord) och `w` (32-bits ord).

`&inbuff` betyder helt enkelt adressen till etiketten `inbuff`.

Mera hjälp

Kommandot `help` ger en lista över kommandotyper. Man kan få veta alla kommandon av en viss typ genom att ange:

`help` *kommandotyp*

eller få specifik hjälp för ett visst kommando med:

`help` *kommando*