Enkel felsökning i program

Fel kan ge sig tillkänna på olika sätt under programkörning. En del fel ser du genom att programmet ger felaktiga utdata. Ibland händer det att ett program somnar in, vilket kan bero på att programmet hamnat i en oändlig slinga (och definitivt inte sover!) eller att det står och väntar på en inmatning som inte kommer därför att t.ex. utskrifter av ledtexter och inläsning kommit ur fas. Sedan kan också allvarliga inträffa fel som leder till att programmet avbryts med något felmeddelande, t.ex. ”segmentation fault” eller ”buss error”.

Beroende på vilket slags fel det är frågan kan du använda olika sätt för att hitta ett fel. Här får du några tips om några enkla metoder för att leta efter och upptäcka fel.

Programmerare har en fantastisk förmåga att missbedöma var fel måste finnas och absolut inte kan finnas – *ta inget för givet* – felsök förutsättningslöst och systematiskt!

Spårutskrifter

En enkel teknik men ofta bra teknik för att hitta fel är lägga in spårutskrifter i programmet. Låt programmet göra utskrifter som visar att programmet passerat en vissa punkter i koden. Värden på variabler är också ofta intressant att skriva ut för kontroll.

```
cerr << ”Spårning: i = ” << i << ”, c = ” << c << endl;
```

Tänk dock på att utmatning kan vara buffrad, annars kan du lätt bli lurad av spårutskrifter. Om du skriver på cerr är det inga problem, den utströmmen är inte buffrad. Om du däremot skriver på cout, som är buffrad, måste du se till att varje spårutskrift avslutas med endl, vilket förutom radframmatning också ser till att hela utmatningsbuffertens innehåll skrivs ut.

Avlusare

Exekveringsfel, som t.ex. ”segmentation fault”, är ofta enkla att hitta med hjälp av en avlusare (”debugger”). Se till att all programkod är kompilerad med flaggan -g. Starta sedan programmet i avlusaren (t.ex. DDD) och kör (utan brytpunkter, stegning eller annat som stoppar upp exekveringen) till dess programmet kraschar. Normalt står du då på den rad i programmet där orsaken till problemet finns. Du kan nu undersöka värdena på de variabler som används på raden, inte sällan är det en pekare eller annan form av adress som har ett illegalt värde.

För att använda avlusaren för att söka andra slags fel krävs lite mer strategi i sökandet.

assert-makrot

När du skriver program kan du lägga in anrop till makrot assert, för att försäkra dig om att något viktigt villkor är uppfyllt då efterföljande kod ska utföras. Det kan t.ex. vara att en variabel ska vara större än noll eller att en pekare inte får vara tomma pekaren. Exempel:

```
assert(n > 0);
```

Om inte ett villkoret är uppfyllt kommer programkörningen att avbrytas med ett felmeddelande:

```
Assertion failed: uttryck, file xyz, line nnn
```

Makrot assert är främst ett hjälpmedel under programutveckling. För att förhindra att assert-anropen kompileras in i programmen kompilerar du med flaggan -DNDEBUG. Preprocessorsatsen #define NDEBUG före #include <cassert> är också ett sätt att förhindra att assert-anropen kommer med.

Observera, assert är ett makro och uttrycket får därför inte innehålla strängliteraler.