Softwaretechnik Übung: Compiler & Binutils

Klaus Kusche

1. Compiliere ein paar eigene Programme mit möglichst vielen *Warnings* und / oder typischen *Fehlern*, provoziere Warnings (z.B. für uninitialisierte Variablen oder für **=** statt **==** in **`if`**'s).

   Vergleiche die Qualität der Warn- und Fehlermeldungen von **`gcc`** und **`clang`**.

2. Wie rufst du **`gcc`** auf, damit der interne *Preprozessor-Output-File* und der *Assembler-Sourcecode-File* erhalten bleiben?

3. Wie erzeugst du mit **`gcc`** eine Liste aller definierten und vordefinierten *Makros*?

4. Erzeuge ein Programm mit voller Debug-Information und wende **`strip`** darauf an. Wie groß ist der *Größen-Unterschied*? Wieviel mehr Ersparnis bringt **`sstrip`** ?

5. Mit welchem Befehl kann man bei **`gcc`** und den Binutils *separate Debug-Info-Files* erzeugen (d.h. einen Exe-File ohne Debug-Info und dazu einen Debug-Info-File).

6. Compiliere unser Grafik-Programm so, dass du die **`.o`**-Files zu jedem Sourcefile erhältst (*mit* Debug-Info).

   Wende das Tool **`nm`** darauf an.

   - Was bedeutet **`T`**, **`U`** usw. im Output?
   - Fällt dir ein Unterschied zwischen den *Funktionsnamen* im **`nm`**-Output auf, wenn du dasselbe Programm einmal als C und einmal als C++ compilierst?

     Wie nennt man das? Von welchem Konstrukt im C++-Source ist das abhängig (d.h. wie bringt man C++ dazu, C-Namen für Funktionen zu erzeugen)?

     Hilft dir das Tool **`c++filt`** weiter?

     Findest du eine Option für **`nm`**, die dasselbe leistet?
   - Kannst du mit **`nm`** auch herausfinden, *wie groß* die einzelnen Variablen und Funktionen sind?
   - Kannst du mit **`nm`** auch herausfinden, an welcher Stelle im *Source* die Symbole bzw. Referenzen stehen? (du brauchst dafür **`.o`**-Files mit Debug-Info!)
   - Wie siehst du *nur die fehlenden Symbole* in einem File?
   - Wie siehst du mit **`nm`**, *welche Funktionen* eine Shared Library bereitstellt?

7. Kannst du den Output des Tools **`size`** deuten?

8. Zeige dir von einem ganz einfachen Programm und einem größeren GUI-Programm an, *welche Shared Libraries* es beim Start wirklich lädt (Tipp: **`ldd`**).

9. Probiere das Tool **`strings`** aus.

10. Experimentiere mit **`objdump`** (auf einem Objekt-File und einem Executable).

    Zu testende Optionen: **`-f -p -h -d -s -t -T`**

    Ein alternatives Tool ist **`readelf`** (mit **`-a`**) und **`elfls`** .

    Welche Informationen siehst du?