



GNU C++

Das freie C

Unter Linux ist der GNU C++-Kompiler Standard. Der Einsatz auf der Kommandozeile ist nicht schwer, jedoch gewöhnungsbedürftig.

OLIVER MÜLLER

Als UNIX erfunden wurde, gab es noch keine grafischen Oberflächen für dieses System. Die Kommandozeile, die Shell, war das einzige Kommunikationsmittel zwischen Benutzer und Betriebssystem. Die Programme wurden über Befehle auf diesem Kommandoprompt aufgerufen und ausgeführt. Die Entwicklungstools, allen voran der C-Kompiler, waren da keine Ausnahme.

Aus dieser Zeit stammt das heute verwendete Prinzip von Kommandozeilentools, wie dem GNU C++-Kompiler. Statt alles in einer Oberfläche verfügbar zu haben, müssen Sie sich mit dem (auf den ersten Blick) zweifelhaften Komfort der Shell zufrieden geben.



GNU C++ ist für Windows erhältlich. Schauen Sie mal bei <http://www.cygwin.com> vorbei.

Von Os und As

Das Duo Kompiler und Linker benötigt und erzeugt verschiedene Dateien. Ein Überblick über alles, was beim Build eines Programms an Dateitypen durch das Dateisystem geistert, ist sicherlich einen Blick wert.

Als Eingabe für den Kompiler dient immer der C++-Quelltext, der in Dateien mit den Endungen *.cc*, *.C* oder *.cpp* gespeichert wird. Eine solche C++-Quelltextdatei wird vom Kompiler in eine Objektcode-datei übersetzt. Diese

Datei trägt die Endung *.o* (unter Windows *.obj*). Dies ist noch kein lauffähiges Programm, sondern nur ein Teilstück einer Software.

Zum Übersetzen der Sources in Objektcode verwendet der Kompiler zwei weitere Tools aus der GNU-Gemeinde. Das erste ist der GNU Präprozessor. Das ist das Programm, das die #-Direktiven, wie *#include*, *#pragma* und *#define*, auflöst. Das zweite ist der GNU Assembler. Dieser generiert den Objektcode.

Starten müssen Sie nur den Kompiler. Wenn Sie den Kompiler als Blackbox betrachten, nimmt dieser den C++-Quelltext als Eingabe und liefert Objektcode als Ausgabe.

Die Objektcodes werden vom Linker zusammen mit dynamischen (*.so**) und/oder statischen Libraries (*.a*) zu einem lauffähigen Programm gebunden. Das lauffähige Programm trägt unter Linux keine besondere Dateiendung, wie etwa das unter Windows übliche *.exe*, da der Status "ausführbar" als Dateiattribut gespeichert wird.



Bei dynamischen Libraries wird Code zum Laden der Libraries eingebunden. Die Library selbst wird erst zur Laufzeit dynamisch hinzu gebunden.

Diese letzte Ausgabedatei, das ausführbare Programm, wird lediglich unter Linux

mit *a.out* benannt, sofern Sie nicht beim Linken explizit einen anderen Dateinamen angeben.

...und Action!

Der C++-Kompiler von GNU heißt unter Linux *g++* bzw. *c++* und wird von der Shell (=Kommandoprompt) aus gestartet. Der Aufruf des C++-Kompilers erfolgt immer nach dieser Form:

```
c++ [Optionen] Quelltext
```

bzw.

```
g++ [Optionen] Quelltext
```

In der Regel sind bei GNU C++ beide Programmnamen - *g++* und *c++* - gültig.

Wenn Sie diesem Kompiler einen C++-Quelltext wie *foo.cc* übergeben und dabei keine weiteren Optionen angeben, erzeugt der Kompiler gleich ein lauffähiges Programm (*a.out*):

```
c++ foo.cc
```

Er ruft sofort nach dem Erzeugen des Objektcodes den Linker auf, der das Executable erstellt. Sie können Ihr brandneues Programm dann durch

```
./a.out
```

starten.

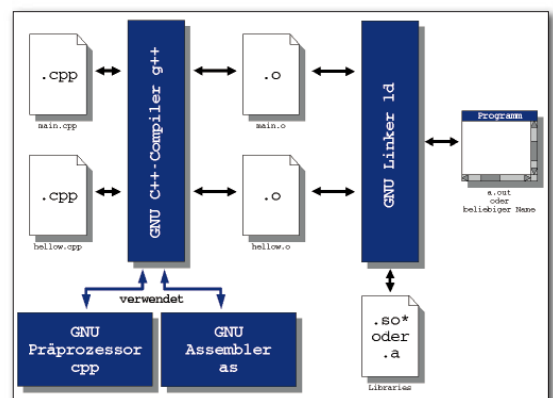
a.out ist nicht gerade der schönste aller Namen, den ein Programm haben kann. Über die Option *-o* gefolgt von einem Dateinamen, können Sie dem Kompiler gleich mitteilen, welcher Namen vergeben werden soll. Der Befehl

```
c++ -o foo foo.cc
```

erzeugt aus dem Quelltext *foo.cc* das ausführbare Programm *foo*.

Professionals

Wenn Sie Software für die Praxis programmieren, werden Sie schnell aufgeben alles in einen Quelltext zu packen.



KOMPILIEREN UND LINKEN, ein komplexer Vorgang



Ein C++-Quelltext für eine große Anwendung, welche Zehntausende von Zeilen umfasst, werden Sie kaum mehr in einer einzigen Datei speichern.

Besteht Ihr Programm aus den Quelltexten `foo1.cc` und `foo2.cc`, dann haben Sie Schwierigkeiten, wenn Sie es mit dem obigen Programm übersetzen. Geben Sie das Kommando

```
c++ foo1.cc
```

ein, wird der Linker die passende Fehlermeldung parat halten. Programmteile, die erst in `foo2.cc` definiert werden, kann er nicht finden, da er durch den obigen Aufruf nur `foo1.cc` kennt.

Hier kommt das oben beschriebene Verfahren zum Einsatz. Zuerst müssen Sie aus den C++-Quelltexten Objektcodes erstellen und diese dem Linker zuführen. Sie müssen dem Compiler mitteilen, dass er nicht sofort den Linker starten soll. Das erreichen Sie mit der Option `-c`.

```
c++ -c foo1.cc
```

Wenn Sie `foo1.cc` durch dieses Kommando kompilieren, wird der Linker

nicht aufgerufen. Sie erhalten damit lediglich eine Datei `foo1.o`.

Wenn Sie hier die Option `-o` verwenden, geben Sie damit den Namen der Objektcode-Datei an.

Nachdem Sie `foo2.cc` auf diese Weise in eine `foo2.o` übersetzt haben, können Sie zum Linken schreiten. Der Aufruf des GNU Linkers `ld` ist den meisten Programmieren zu umständlich, da in seiner Kommandozeile der Startup-Code und die grundlegenden Libraries explizit angegeben werden müssen.

Statt sich dieser Tortur zu unterziehen, verwenden findige C++-Programmierer unter Linux einfach wieder den C++-Compiler, um den Linker zu starten. In der Kommandozeile geben Sie jetzt nur die Objektcodes statt der C++-Quelltexte an:

```
c++ -o foo foo1.o foo2.o
```

Das erzeugt das Programm `foo` aus den Objektcode-Dateien `foo1.o` und `foo2.o`.

Bibliothekar

Wenn Ihr Programm weitere Libraries, wie MICO, verwenden soll, müssen Sie

diese ebenfalls angeben. Um MICO einzubinden, müssen Sie zunächst wissen, wie das Library-File heißt: `libmico2.3.3.so`. Eine solche Library hat einen fest aufgebauten Namen. Sie besteht aus den Teilen "lib" "Name" "Version" "Erweiterung". Für MICO sind dies: "lib", "mico", "2.3.3", und ".so". Um eine solche Library einzubinden, geben Sie diese durch die Option `-l` direkt gefolgt vom Namen der Bibliothek an:

```
c++ -o foo -lmico foo1.o foo2.o
```

Damit Compiler und Linker die C++-Header-Files bzw. Libraries finden, können Sie über die Optionen `-I` respektive `-L` Suchpfade dafür angeben:

```
c++ -c -I/usr/local/include/mico  
foo1.cc
```

Wenn die MICO-Header in `/usr/local/include/mico` liegen. Liegt die MICO-Library im Pfad `/usr/local/lib`, nennen Sie dieses Verzeichnis beim Linken:

```
c++ -o foo -L/usr/local/lib foo1.o  
foo2.o
```



DAS GNU KOMPIERER

Das GNU Compiler-System umfasst eine Reihe von Paketen, die installiert sein müssen, damit Sie Sources übersetzen können. Im Wesentlichen müssen Sie die Compiler und Linker, die Entwicklungs-Libraries und entsprechende Software-Werkzeuge installieren.

Je nachdem wie das Installationsprogramm Ihrer Distribution arbeitet, müssen Sie entweder separate Pakete auswählen, oder können komplette Bereiche wie "Entwicklung" angeben.

Wenn Sie einzelne Pakete, wie RPMs installieren müssen, benötigen Sie für C++ folgende Pakete

```
- cpp  
- dev86  
- binutils  
- patch  
- cdecl
```

und den Compiler und sein C++-Frontend

```
selbst:  
- gcc  
- gcc-c++  
oder  
- egcs  
- egcs-c++
```

Damit Sie Software entwickeln können, müssen Sie diese Entwicklungsbibliotheken installieren:

```
- glibc-devel  
- libstdc++-devel
```

Diese Basis können Sie bereits zum Programmieren von einfachen C++-Programmen verwenden. Es ist jedoch sinnvoll andere häufig benötigte Libraries zu installieren:

```
- zlib-devel  
- readline-devel  
- gpm-devel  
- libtermcap-devel  
- ncurses-devel  
- svgalib-devel
```

Für viele grafische Programme, auch X11-Software benötigen Sie:

```
- libgr-devel  
- libpng-devel  
- libtiff-devel  
- libjpeg-devel  
- libungif-devel
```

Wenn Sie X11-Programme kompilieren wollen, benötigen Sie zusätzlich folgende Pakete mit Entwicklungs-Libraries:

```
- XFree86-devel  
- Xaw3d-devel  
- glib-devel  
- gtk+-devel  
- xpm-devel
```

Wollen Sie GNOME-Software übersetzen, sind neben den zuvor genannten X11-Paketen noch folgende Packages fällig:

```
- ORBit-devel  
- audiofile-devel  
- control-center-devel  
- esound-devel  
- fnlib-devel  
- gnome-core-devel  
- gnome-games-devel  
- gnome-libs-devel  
- gnome-pim-devel  
- imlib-devel  
- libhttp-devel  
- libgtop-devel  
- libxml-devel
```

Für KDE-Software müssen Sie neben den X11-Paketen das Paket

```
- qt-devel
```

installieren.

Nur Compiler und Linker sind etwas sparsam zu verwenden. Sie sollten daher diese zusätzlichen Software-Werkzeuge installieren:

```
- libtool  
- m4  
- make  
- autoconf  
- automake
```