

# F Den Compiler kompilieren

Das Team um Free Pascal veröffentlicht in bestimmten Abständen vorbereitete Pakete, die sogenannten Releases, die den Compiler und die geläufigsten Units enthalten, direkt lauffähig sind und in sich geschlossene Verbesserungen enthalten. Nach einem Release geht die Arbeit am Compiler natürlich weiter, Fehler werden beseitigt und neue Verbesserungen und Optionen hinzugefügt. Das Free-Pascal-Team bringt jedoch nicht ein neues Release heraus, sobald irgendetwas am Compiler geändert wird, stattdessen stehen die Quelltextdateien für jeden zum Gebrauch und zur eigenen Kompilierung zur Verfügung. Kompilierte Versionen werden ebenfalls täglich erzeugt und ins Web gestellt.

Es mag trotzdem Situationen geben, in denen Sie den Compiler selbst erzeugen wollen. Beispielsweise, wenn Sie selbst Änderungen am Compilercode vorgenommen haben oder den Quelltext über CVS runtergeladen haben. Es gibt zwei verschiedene Wege um den Compiler zu rekompilieren, entweder per Hand oder über die Makefiles. Beide Methoden werden hier besprochen.

Um den Compiler ohne Probleme zu kompilieren, ist es am besten, wenn Sie die folgende Verzeichnisstruktur beibehalten (das Ausgangsverzeichnis »/pp/src« ist empfehlenswert, kann aber auch anders sein):

```
/pp/src/Makefile
      /makefile.fpc
      /rtl/linux
            /inc
            /i386
            /...
      /compiler
```

Wenn Sie die Makefiles verwenden wollen, müssen Sie diesen Verzeichnisbaum verwenden.

Der Quelltext des Compilers und der Laufzeitbibliothek sind so gepackt, daß wenn Sie beide Archive im gleichen Verzeichnis (»/pp/src« in obigen Beispiel) entpacken, der obige Verzeichnisbaum entsteht. »makefile.fpc« und das Makefile finden Sie in der Datei »base.zip« auf der FTP-Site. Wenn Sie manuell kompilieren, benötigen Sie diese Dateien nicht.

Je nach Situation gibt es zwei Wege, wie sie mit der Kompilierung beginnen können. Normalerweise muß die Laufzeitbibliothek zuerst kompiliert werden, und erst dann der Compiler unter Verwendung des alten Compilers. In einigen speziellen Fällen muß der Compiler aber mit einer älteren Version der RTL als erstes kompiliert werden.

Wie bestimmt man, was zuerst kompiliert werden muß? Im allgemeinen lautet die Antwort: zuerst die RTL. Es gibt zwei Ausnahmen zu dieser Regel:

1. Die erste Ausnahme tritt ein, wenn einige interne Routinen der RTL sich verändert haben. Da der alte Compiler über diese internen Veränderungen nicht Bescheid weiß, gibt er Funktionen aus, die auf der alten RTL basieren und deshalb nicht korrekt sind. Entweder wird das Linken fehlschlagen oder die Programmdatei wird Fehlermeldungen ausgeben.
2. Die zweite Ausnahme tritt ein, wenn etwas der RTL hinzugefügt wurde, was der Compiler wissen muß (beispielsweise ein neuer Assemblermechanismus).

Woher weiß man aber nun, ob diese Dinge stattgefunden haben? Es gibt hierauf keine Antwort, außer dem Free Pascal Team zu e-mailen. Wenn Sie den Compiler nicht auf dem einen Weg kompilieren können, versuchen Sie einfach den anderen.

## ● **Compilierung mittels Make**

Wenn Sie mittels *Make* kompilieren, ist es notwendig, daß die oben beschriebene Verzeichnisstruktur vorhanden ist. Der Compiler wird kompiliert, indem Sie *cycle* als Ziel für *Make* angeben.

Unter normalen Umständen beschränkt sich die Compilierung des Compilers auf die folgenden Schritte (unter der Voraussetzung, daß Sie sich im Verzeichnis »/pp/src« befinden):

```
cd compiler
make cycle
```

Dies funktioniert nur dann, wenn »makefile.fpc« korrekt installiert wurde und alle notwendigen Hilfsprogramme in den jeweiligen Verzeichnissen (*PATH*) gefunden werden können. Die obigen Befehle führen zu folgendem:

- ◆ Unter Verwendung des aktuellen Compilers wird die Laufzeitbibliothek ins richtige Verzeichnis kompiliert, das durch das Betriebssystem bestimmt wird, unter dem Sie arbeiten. Beispielsweise befindet sich unter Linux die kompilierte RTL im Verzeichnis »rtl/linux«.
- ◆ Der Compiler wird unter Verwendung der neu kompilierten RTL kompiliert. Falls dies erfolgreich verlief, wird der neu kompilierte Compiler in eine temporäre Programmdatei kopiert.
- ◆ Unter Verwendung des temporären Compilers wird die RTL neu kompiliert.
- ◆ Mit dem temporären Compiler und der neuen RTL wird dann der Compiler erneut kompiliert.

Die letzts zwei Schritte werden so lange wiederholt, bis drei Durchgänge gemacht wurden oder die Programmdatei des erzeugten Compilers mit der des Compilers, mit der sie erzeugt wurde, übereinstimmt. Dies stellt sicher, daß die Programmdatei des Compilers korrekt ist.

## ● **Compilierung für eine andere Plattform**

Wenn Sie für eine andere Plattform kompilieren wollen, müssen Sie die Makefile-Variable *OS\_TARGET* setzen. Mögliche Werte für die Variable sind *win32*, *go32v2*, *os2* und *linux*. Als Beispiel soll die Cross-Compilierung einer *go32v2*-Version von einer Win32-Plattform aus dienen:

```
cd compiler
make cycle OS_TARGET = go32v2
```

Dies erzeugt eine *go32v2*-Laufzeitbibliothek und einen *go32v2*-Compiler.

Wenn Sie einen neuen Compiler erzeugen wollen, aber hierbei der Compiler zuerst mit einer vorhandenen RTL kompiliert werden soll, müssen Sie *all* als Ziel und ein anderes Verzeichnis als das Standardverzeichnis für die RTL (das das Verzeichnis *./rtl/\${OS\_TARGET}* ist) angeben. Wenn zum Beispiel die kompilierte RTL sich in »/pp/rtl« befindet, können Sie eingeben:

```
cd compiler
make clean
make all UNITDIR=/pp/rtl
```

Dies kompiliert dann den Compiler unter Verwendung der RTL im Verzeichnis »/pp/rtl«. Anschließend können Sie dann *make cycle* aufrufen:

```
make cycle PP=./ppc386
```

Dies führt dazu, daß die gleichen Schritte wie oben ausgeführt werden, der Compiler aber verwendet wird, der durch die *make all*-Anweisung erzeugt wurde.



In allen Fällen können viele Optionen hinzugefügt werden, um den Kompilierungsprozeß zu beeinflussen. Im allgemeinen kann man jedoch sagen, daß die Makefiles alle benötigten Compileroptionen zur Kommandozeile hinzufügen, so daß die RTL korrekt kompiliert werden kann. Sie können zusätzliche Optionen (beispielsweise Optionen zur Optimierung) über die Variable *OPT* übergeben.

## ● Manuelles Kompilieren

Manuelles Kompilieren ist schwierig und anstrengend, aber dennoch möglich. Die Kompilierung von RTL und Compiler werden separat besprochen.

*Kompilierung der RTL:*

Um die RTL zu kompilieren, damit ein neuer Compiler erzeugt werden kann, müssen zumindest die folgenden Units in der gegebenen Reihenfolge kompiliert werden:

*loaders:* Die Programmfragmente, die den Start-Code für jedes Programm bilden. Diese Dateien haben die Dateierweiterung ».as«, weil sie in Assemblersprache geschrieben wurden. Sie müssen mittels des GNU Assemblers »as« kompiliert werden. Diese Fragmente finden sich im Verzeichnis der vom Betriebssystem abhängigen Dateien, mit Ausnahme von Linux, bei dem sie sich im vom Prozessor abhängigen Unterverzeichnissen (»i386« oder »m68k«) befinden.

*system:* Die System-Unit. Die Unit heißt auf den verschiedenen Systemen unterschiedlich: nur unter GO32v2 heißt sie auch »system«, unter Linux wird sie »syslinux« genannt. »Syswin32« heißt sie unter Windows NT, unter OS/2 heißt sie dann »sysos2«. Die Unit kann in den Verzeichnissen gefunden werden, die vom Betriebssystem abhängig sind.

*strings:* Die String-Unit. Diese Unit befindet sich im Unterverzeichnis »inc« der RTL.

*dos:* Die Dos-Unit. Sie befindet sich im vom Betriebssystem abhängigen Verzeichnis der RTL. Es kann sein, daß beim Kompilieren dieser Unit andere Units ebenfalls mitkompiliert werden (beispielsweise wird unter Linux die Unit *linux* und unter go32v2 die Unit *go32* mit kompiliert).

*objects:* Die Objects-Unit. Sie befindet sich im Unterverzeichnis »inc« der RTL.

Um diese Units auf einem i386 zu kompilieren, genügen die folgenden Befehle:

```
ppc386 -Tlinux -b- -Fi../inc -Fi../i386 -FE. -di386 -Us -Sg syslinux.pp
ppc386 -Tlinux -b- -Fi../inc -Fi../i386 -FE. -di386 ../inc/strings.pp
ppc386 -Tlinux -b- -Fi../inc -Fi../i386 -FE. -di386 dos.pp
ppc386 -Tlinux -b- -Fi../inc -Fi../i386 -FE. -di386 ../inc/objects.pp
```

Dies sind die minimalen Kommandozeilenbefehle, die zur Kompilierung der RTL nötig sind.

Für andere Prozessoren sollten Sie i386 in den entsprechenden Prozessor ändern, unter anderen Betriebssystemen müssen sie *syslinux* in die entsprechende System-Unit ändern und das Zielbetriebssystem (-T) will ebenfalls angepaßt werden.

In Abhängigkeit des Betriebssystems für das kompiliert werden soll, gibt es andere Units, die Sie vielleicht kompilieren wollen, die aber nicht unbedingt für die Kompilierung des Compilers benötigt werden. Die folgenden Dateien stehen für alle Plattformen zur Verfügung:

*objpas:* Wird für den Delphi-Modus benötigt. Verlangt -S2 als Kommandozeilenoption und befindet sich im Unterverzeichnis »objpas«.

*sysutils:* Enthält viele Hilfsfunktionen, wie aus Delphi bekannt. Befindet sich im »objpas«-Verzeichnis und benötigt zum Kompilieren die Kommandozeilenoption -S2.

*typinfo:* Funktion um, wie in Delphi, auf die Laufzeittypinformationen (RTTI) zuzugreifen. Befindet sich im Verzeichnis »objpas«.

*math:* Die mathematischen Funktionen wie unter Delphi. Befindet sich ebenfalls im »objpas«-Verzeichnis.

*mmx:* Die Erweiterungen für die Intel-Prozessoren mit MMX-Technologie. Befindet sich im Verzeichnis »i386«.



*getopts*: Eine zu GNU-kompatible »getopts«-Unit. Befindet sich im »inc«-Verzeichnis.

*heaptrc*: Eine Unit um Fehlersuche auf dem Heap durchzuführen. Befindet sich im »inc«-Verzeichnis.

## ● Kompilierung des Compilers

Die Kompilierung des Compilers kann eigentlich über einen einzigen Befehl geschehen. Es ist allerdings immer am besten, zuerst alle Units zu entfernen, die sich im Compiler-Verzeichnis befinden:

```
rm *.ppu *.o
```

unter Linux. Unter Dos ergibt sich:

```
del *.ppu
```

```
del *.o
```

Anschließend kann der Compiler durch folgenden Befehl kompiliert werden:

```
ppc386 -Tlinux -Fu../rtl/linux -di386 -dGDB pp.pas
```

Somit sind die minimalen Optionen:

- ◆ Das Zielbetriebssystem. Dies kann weggelassen werden, wenn für die gleiche Plattform kompiliert wird, die momentan verwendet wird.
- ◆ Ein Verweis auf das Verzeichnis der RTL. Dies kann weggelassen werden, wenn sich eine korrekte »ppc386.cfg«-Datei auf Ihrem System befindet. Wenn Sie die RTL zuerst kompilieren wollen, sollten Sie hier »../rtl/OS« angeben (ersetzen Sie *OS* mit dem entsprechenden Unterverzeichnis der RTL).
- ◆ Geben Sie den Prozessor an, für den Sie kompilieren. Diese Angabe ist zwingend erforderlich.
- ◆ *-dGDB* wird nicht unbedingt benötigt, aber es ist besser, denn ansonsten können Sie nicht mit Informationen zur Fehlersuche kompilieren.
- ◆ *-Sg* wird benötigt, da einige Teil des Compilers *goto*-Anweisungen verwenden (um genauer zu sein: der Scanner).

Jetzt also absolut minimal:

```
ppc386 -di386 -Sg pp.pas
```

Sie können andere Kommandozeilenoptionen angeben, aber die obigen sind das Minimum.

Eine Liste aller erkannten Symbole finden Sie in folgender Tabelle.

Symbol	Beschreibung
USE_RHIDE	Generiert Fehler und Warnung in einem Format, das von <i>RHIDE</i> erkannt wird.
TP	Wird benötigt, um den Compiler mit Turbo oder Borland Pascal zu kompilieren.
Delphi	Wird benötigt, um den Compiler mit Delphi von Borland zu kompilieren.
GDB	Schaltet die Unterstützung für den GNU Debugger an.
I386	Erzeugt einen Compiler für Intel-Prozessoren (386+).
M68K	Erzeugt einen Compiler für M68000-Prozessoren.
USEOVERLAY	Kompiliert eine TP-Version, die Overlays verwendet.
EXTDEBUG	Es wird zusätzlicher Code zur Fehlersuche ausgeführt.
SUPPORT_MMX	Aktiviert den Compilerschalter MMX, der es ermöglicht, MMX-Befehle zu verwenden (nur i386).
EXTERN_MSG	Kompiliert die msg-Dateien des Compilers nicht, sondern benutzt immer externe Message-Dateien (Standard für TP).
NOAG386INT	Der entstehende Compiler kann keinen Intel-Assembler-Code ausgegeben.
NOAG386NSM	Der entstehende Compiler kann keinen NASM-Code ausgegeben.
NOAG386BIN	Der binäre Schreiber wird nicht eingebunden.

**Tab. F.1:** Mögliche Symbole bei der Kompilierung von FPC