StormC-FAQ-deutsch

COLLABORATORS			
	TITLE : StormC-FAQ-deutsch		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 8, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

StormC-FAQ-deutsch

1.1 StormC-FAQ

```
                StormC-FAQ
Stand: 28 May 1999


            Wie bekomme ich Support?

            Warum kommt es zu der Linker-Fehlermeldung 'Symbol _exit nicht

            definiert', wenn man als Shared-Library linkt?

            Warum ist die Bibliothek 'storm.lib' so gross und warum gibt es im

            Gegensatz zu SAS/C nur eine Bibliothek?

            Warum kompilieren andere Compiler schneller als StormC?

            Wie kann man ein 'Hello World' Programm wirklich klein kompilieren ↩
               ?

            Warum ist selbst ein kleines Programm wie 'Hello World' gleich  ↩
               mehrere

            KBytes lang?

            Warum hat StormC keinen 'Global Optimizer'?
                Dieses FAQ ist Copyright by Haage&Partner.
Neuigkeiten und aktuelle FAQs bei: http://www.haage-partner.com

Erzeugt am 28 May 1999 von Fiasco von Nils Bandener
und einem magischen ARexx-Skript von Martin Steigerwald.
```

1.2 StormC-FAQ

```
Wie bekomme ich Support?
```

Am schnellsten und einfachsten geht der Support über das Internet. Wichtige Voraussetzung dafür ist, daß sie bei uns registriert sind. Bitte schicken sie uns auf keinen Fall ihre Seriennummer über das Internet, es sei denn sie verwenden PGP.

Info: http://www.haage-partner.com/sc_d.htm
Support: http://www.haage-partner.com/sc_sup_d.htm

Mailingliste: http://www.haage-partner.com/sc_mlist.htm

Email-Support: <stormc-support@haage-partner.com>

1.3 StormC-FAQ

Warum kommt es zu der Linker-Fehlermeldung 'Symbol _exit nicht definiert', wenn man als Shared-Library linkt?

Die Shared-Library ruft die ANSI-Funktion exit() auf. Das kann sie zu einen explizit, weil Sie diese Funktion verwenden, oder zum anderen implitzit durch Linker-Bibliotheken, die diese Funktion verwenden. Die 'Storm.Lib' nutzt diese Funktion beim automatischen Oeffnen der benutzten Shared-Libraries, z.B. der 'utility.library'.

Grundsaetzlich darf aber eine Shared-Library die Funktion exit() nicht verwenden, da sie nicht einfach so beendet werden kann. Wie vermeidet man den Aufruf?

Man darf das automatische Oeffnen von benutzten Shared-Libraries nicht verwenden, sondern muss die Bibliotheken wie im Handbuch auf Seite 140 beschrieben, oeffnen und schliessen.

Um herauszufinden, welche Bibliotheken alle benutzt werden, sollte man zuerst eine Funktion: 'void exit() {}' in die Shared-Library aufnehmen. Jetzt laesst sich die Library linken.

Verwenden Sie die Linker-Option 'Map-Datei schreiben'. Der Linker erzeugt eine Datei mit der Endung '.map'. Suchen Sie alle INIT-Funktionen, die den Basisnamen einer Shared-Library enthalten, z.B. INIT_1_UtilityBase.

Öffnen Sie diese Bibliotheken nun alle selbst. Denken Sie auch daran, die entsprechende Basisvariable (z.B. UtilityBase) selbst zu deklarieren. Vergessen Sie nicht, die eigene exit()-Funktion wieder aus Ihrem Source zu entfernen.

1.4 StormC-FAQ

Warum ist die Bibliothek 'storm.lib' so gross und warum gibt es im Gegensatz zu SAS/C nur eine Bibliothek?

StormC unterstuetzt ein weiterentwickeltes Objektformat, das auch bei Linkerbibliotheken zum Einsatz kommt. Dieses Format ist 100% kompatibel zum alten (sowohl aufwaerts, wie auch abwaerts), allerdings koennen der

StormLinker und der StormC Compiler mit diesem Format mehrere Datenmodelle in einer Objektdatei aufnehmen.

Damit bleibt fuer Sie nicht mehr die fehleranfaellige Auswahl der richtigen Bibliothek zu ihren gewaehlten Compileroptionen, sondern der Linker waehlt aus der grossen Bibliothek 'storm.lib' die Teile aus, die fuer das gewaehlte Datenmodell (grossesDatenmodell oder eines der beiden kleinen Datenmodelle) gerade passt. Deshalb ist die 'storm.lib' etwa so gross, wie drei einzelne Bibliotheken fuer jedes Datenmodell zusammen.

In Zukunft wird StormC auch noch verschiedene Codemodelle und CPU- bzw. FPU-Modelle in der Bibliothek unterstuetzen, sodass die 'storm.lib' und alle weiteren Bibliotheken jeweils optimale Programmeerzeugung automatisch erlauben.

1.5 StormC-FAQ

Warum kompilieren andere Compiler schneller als StormC?

StormC erzeugt sauberen optimierten Code mit optimaler Registerbenutzungund vielfaeltigen Optimierungen. Ausserdem ist StormC konsequent auf den PowerPC vorbereitet und verzichtet deshalb auf den Einsatz schneller aber fehleranfaelliger Assemblerroutinen. Leider ist darum die derzeitige Version des Compilers nicht so schnell wie beispielsweise MaxonC++, allerdings arbeiten wir an einer speziellen Optimiererstufe fuer die Entwicklungsphase eines Projektes mit noch kuerzeren Uebersetzungszeiten.

1.6 StormC-FAQ

Wie kann man ein 'Hello World' Programm wirklich klein kompilieren?

Kann man auf Fliesskommaausgabe verzichten und reicht einem die Pufferung des AmigaDOS aus, kann man jederzeit zur Ausgabe das AmigaDOS direkt verwenden. Die Funktionen 'VPrintf' und 'VFPrintf' ermoeglichen nämlich direkt die Ausgabe aehnlich wie 'printf' auf AmigaDOS Dateien. Allerdings sind diese Funktionen nicht 100% ANSI kompatibel.Weitere Moeglichkeiten bietet natuerlich der Verzicht auf das automatische Öffnen und Schliessen der Bibliotheken, denn die dazu benutzten Funktionen bieten eine komfortable Fehlermeldungsausgabe mit Unterscheidung zwischen Workbench und CLI Start eines Programms und Beachtung alter Betriebssystemversionen 1.3 und aelter.

Dieser Komfort ist nicht fuer jedes Programm notwendig. Man kann einen Minimalstartupcode, den man in Assembler schreibt als eigenen Startupcode benutzen und darin nur die notwendigsten Arbeiten erledigen, z.B. das kleine Datenmodell unterstuetzen ohne gleichzeitig residente Programme zu erlauben.

1.7 StormC-FAQ

Warum ist selbst ein kleines Programm wie 'Hello World' gleich mehrere KBytes lang?

Die mitgelieferte StormC Bibliothek 'storm.lib' ist eine ANSI C Bibliothek. Fuer ein 'Hello World' Programm muss aus dieser Bibliothek die 'printf' Funktion gelinkt werden. Dadurch kommen aber auch unbenutzte Funktionen z.B. fuer die Ausgabe von Integer und Fliesskommazahlen ins Programm, da aus dem 'printf' Befehl nicht ersichtlich ist, welche der Umwandlungen noetig sind. Ausserdem werden alle Ausgaben in Dateien gepuffert durchgefuehrt. Dadurch werden auch kurze Programm relativ gross.

1.8 StormC-FAQ

Warum hat StormC keinen 'Global Optimizer'?

Ein 'Global Optimizer' optimiert ein Programm unter Beachtung der ganzen Funktion und nicht nur einzelner Anweisungen oder Ausdruecke. Dadurch kann ein Global Optimizer z.B. Ausdruecke, die in einer Schleife ausgewertet werden, aber in jedem Scheifendurchlauf immer das gleiche Ergebnis liefern muessen, aus der Schleife herausziehen und schon vor der Schleife bearbeiten.

StormC kann diese globale Optimierung bislang nicht, ist allerdings schon in Version 1 dazu vorbereitet, in einer der naechsten Versionen auch diese Optimierung zu beherrschen. Allerdings kann StormC schon jetzt einige Optimierungen, die auch Aufgaben des Global Optimizers sind. Dazu gehoert insbesondere die globale CPU und FPU Registerverteilung in den hoeheren Optimierungsstufen. Die Register werden in der ganzen Funktion unter Beachtung aller Variablenzuweisungen und Funktionsaufrufe in der Funktion optimal verteilt.