Alte Software für AmigaOS4 neu kompilieren

Ein altes C-Program kann durchaus für den AmigaOne und AmigaOS 4 angepaßt werden. Meine Erfahrungen basieren auf SAS/C 6.5x und StormC (GCC-basiert). Um Software zu portieren sollte man sich an die folgenden Tips halten:

1. Header-Dateien

a) Clibs, Pragmas und Proto Header-Dateien

Diese Header-Dateien enthalten hauptsächlich Header für die AmigaOS Libraries usw. In einigen Programmen werden Clib und Pragma Header direkt verwendet:

```
#include <clib/exec_protos.h>
#include <pragmas/exec_pragmas.h>
```

GadtoolsBox schreibt manchmal Header in diesem Format. Diese Methode funktioniert nicht vollständig mit GCC auf AmigaOS 4. Man sollte diese Zeilen durch Proto Header ersetzen:

```
#include <proto/exec.h>
```

Diese Methode holt selbständig alle nötigen Clib und Pragma Header.

b) Inline Header-Dateien

GCC verwendet die Inline Header-Dateien, um auf die AmigaOS Libraries zuzugreifen, welche automatisch durch die Proto Header in ein Programm eingebunden werden. SAS/C verwendet die Inline Header nicht.

2. C-Functionen

a) Formatierung von Funktionen

In den frühen Tagen wurden Funktionen in C in einem anderen Format geschrieben, welches die Datentypen der Argumente des Funktionsaufrufes in separaten Zeilen definierte:

```
myFunction(num1, num2, string1)
int num1, num2;
char *string1;
{

}
```

Dies entspricht nicht dem modernen ANSI-Format. Bei myFunction wird hier angenommen, daß es entweder int oder void zurückgibt und die Definitionen der Argumente sind separat. Solche Funktions-Köpfe sollten einzeilig umgeschrieben werden:

```
int myFunction(int num1, int num2, char *string1)
{

}
```

b) Prototypen

Gewöhnlich werden entweder am Anfang der C-Dateien oder in einer Header-Datei für sämtliche Funktionen Prototypen deklariert. Wenn man mehrere C-Dateien hat, kann man alle Prototypen in eine Header-Datei setzen und diese dann mittels #include in allen C-Dateien einbinden, um die Konsistenz innerhalb des ganzen Projektes sicherzustellen. Man sollte sich hier wie oben beschrieben an den ANSI-C Standard halten. z.B.

```
/* Prototypes */

int myFunction(int num1, int num2, char *string1);
```

c) Void

Wenn keinerlei Parameter übergeben oder zurückgegeben werden, sollte man void benutzen, wo immer es möglich ist:

```
void otherFunction((void)
{
}
```

d) Parameter und Argumente

Wenn man Daten als Argumente übergibt, sollte man sich davon überzeugen, daß der korrekte Datentyp verwendet wird. Zum Beispiel sollte man nicht "long" übergeben, wenn die Funktion "int" erwartet, da dies viele Warnungen beim kompilieren ergeben kann. Entweder setzt man den korrekten Datentyp am Anfang der betroffenen Funktion oder man wandelt den Datentyp bei allen Aufrufen um:

```
void otherFunction(void)
{
    long numa, numb;
    char name[30];
    myFunction(numa, numb, name); /* this will generate warnings as numa,numb are longs not ints */
}
```

sollte umgeschrieben werden in:

```
void otherFunction(void){
    int numa, numb;
    char name[30];
    myFunction(numa, numb, name); /* this is now correct */
}
```

Die Funktion main() sollte immer ein "int" zurückgeben, dazu schreibt man am Ende den Befehl "return 0;" um den fehlerfreien Abschluß des Programms anzuzeigen. Falls benötigt, kann man sich die Kommandozeilen-Argumente argc und argv übergeben lassen; ansonsten kann man auch "void" verwenden.

e) Kompilieren
Wenn das zu portierende Programm für SAS/C geschrieben wurde, startet man SCOPTS und ändert die Stufe der Warnhinweise ("message options") in "ANSI, STRICT". Ebenso sollte man alle Warnungen ("warnings") einschalten (falls welche ausgeschaltet sind). Nun kompiliert man neu. Das Programm verwendet evtl. alte Funktoinen, die nur auf dem SAS/C existieren. Diese existeren nicht in anderen Kompilern wie StromC oder GCC. Funktionen, die nicht mit dem GCC verwendet werden können, sind in der SAS/C-Dokumentation im "User's Guide" unter "C Library Reference" entsprechend gekennzeichnet. Alle Funktionsaufrufe, die mit "AmigaOS", "OLD", "UNIX" oder "SAS/C" gekennzeichnet sind müssen in ihre ANSI-Gegenstücke umgewandelt oder neu geschrieben werden.

Ist das Programm einmal erfolgreich neu kompiliert, kopiert man den Quellcode in ein anderes Verzeichnis und kompiliert es mit GCC/StormC. Dies sollte alle weiteren Fehler oder Warnungen aufzeigen, die weitere Aufmerksamkeit benötigen.

Nutzt man Assembler, kann man 68k-Code mit PhxAss neu kompilieren, was kompatibel mit SAS/C Assembler-Code ist, wenn man das "asm-include" Verzeichnis im Projekt includet, um die Assembler Header-Dateien einzubinden. Wenn möglich, sollte man Assembler-Code in C neu schreiben um die Portabilität auf den PowerPC zu verbessern.

Weiterhin sollte man den Code so anpassen, daß so viele Fehler und Warnungen wie möglich beim kompilieren ausgemerzt werden. So wird es einfacher später den Code zu aktualisieren und man entfernt so potentielle Bugs aus dem Code. Ebenfalls sehr wichtig ist das testen; man sollte sich gute Test-Bedingungen überlegen um alle logischen Fehler zu beseitigen, denn nicht alle diese werden vom Kompiler gefunden.

f) Makefiles
Makefiles sind nicht einfach zwischen zwei verschiedenen Kompilern übertragbar. In SAS/C kann man Makefiles mit "smake" erstellen und mit "SCOPTS" Optionen für den Prozessor-Typ, Optimierung usw erstellen. Bei StormC enthält die Projekt-Datei diese Informationen (lesbar mit einem Text-Editor). Für GCC muß man ein eigenes Makefile schreiben oder mit dem bevorzugten Tool erstellen.

g) Mit Updates experimentieren
Wenn man einfach mal loslegen möchte, kann man ein altes Amiga-Programm aus dem Aminet laden, für das der Quellcode vorliegt. Bevor man jedoch seine Änderungen veröffentlicht sollte man das Readme und die beiliegende Dokumentation lesen. Der Urheber könnte etwas dagegen haben, wenn die eigene Arbeit ohne sein Einverständnis aktualisiert oder weiterverbreitet wird. Falls man sich nicht sicher ist, sollte man besser nichts unternehmen.

h) Upgrade auf neuere GUI APIs
Der Amiga hat den Vorteil einer ganzen Auswahl von GUI-Systemen: Inituition, GadTools, MUI, ClassAct, Reaction und viele andere. Falls man es sich zutraut, kann man auch gerne ein Programm upgraden, sodaß es die neueren GUIs wie Reaction oder MUI nutzt. Beide werden von AmigaOS 4 unterstüztt. Für Reaction steht das Programm Reactor bereit, welches man auf der Amiga Developer CD 2.1 im NDK 3.5 Verzeichnis findet. Für MUI gibt es den MUIBuilder.

Geschrieben von:
Peter Hutchison
pjhutch@pcguru.plus.com
http://www.pcguru.plus.com/

Redigiert vom Intuitionbase-Team
Übersetzt von:
Philippe Bourdin

Published 1st July 2004