930185

Herausgegeben von/Published by
Siemens Nixdorf Informationssysteme AG
Postfach 2160, W-4790 Paderborn
Postfach 830951, W-8000 München 83

Bestell-Nr./Order No. ***U6301-J-Z145-1***
Printed in the Federal Republic of Germany
410 AG 2925. (520)

SIEMENS NIXDORF

SINIX

CES V5.41

cc-Kommando für MX300 und WX200

Beschreibung

Sie haben

uns zu diesem Handbuch etwas mitzuteilen?
Schicken Sie uns bitte Ihre Anregungen unter
Angabe der Bestellnummer dieses Handbuches.

Siemens Nixdorf Informationssysteme AG
Manualredaktion STM QM 2
Otto-Hahn-Ring 6
W-8000 München 83

Fax: (089) 636-40443

email im EUnet:
man@sieqm2.uucp

Dieses Handbuch ist auf chlorfrei gebleichtem Papier gedruckt.

CES (SINIX) V5.41

cc-Kommando für MX300
und WX200

Beschreibung

Ausgabe April 1992 (CES V5.41)

Das cc-Kommando

In dieser Kurzbeschreibung wird das `cc`-Kommando von CES V5.41 für das Betriebssystem SINIX V5.41 auf den Maschinen MX300 und WX200 beschrieben.

Folgende Optionen sind gegenüber der Beschreibung des `cc`-Kommandos in dem Handbuch "CES V5.41 Leitfaden und Werkzeuge für die Programmierung mit C" zusätzlich dargestellt:

`-X`	Übersetzung in Assemblercode (siehe Seite 5)
`-F`	Optimierungsoption (siehe Seite 11)
`-KPIC`	Steuerung der Generierung von positionsunabhängigem Code (siehe Seite 10)
`-KFCW`	Steuerung des Rundungsverhaltens bei Floating-Point-Operationen (siehe Seite 13)

Für alle übrigen Optionen entspricht die folgende Beschreibung dem Abschnitt "Das cc-Kommando" im Handbuch "CES V5.41 Leitfaden und Werkzeuge für die Programmierung mit C".

In dieser Kurzbeschreibung finden Sie im Anschluß an die Beschreibung des cc-Kommandos eine Liste von Dateien und Dateiverzeichnissen, die für CES von Bedeutung sind.

Voraussetzung für den korrekten Ablauf des CES V5.41-Übersetzers ist, daß das Standard C-Entwicklungssystem mit den entsprechenden Headerfiles, Bibliotheken und Werkzeugen aus dem Paket `<scde>` installiert ist.

Vor dem Aufruf beachten

`cc` erzeugt Dateien für Zwischen- und Endergebnisse. Daher müssen Sie für das Dateiverzeichnis Schreiberlaubnis haben, in dem `cc` die Dateien erzeugt. Standardmäßig erzeugt `cc` die Dateien in dem Dateiverzeichnis, in dem `cc` aufgerufen wird.

Die Umgebungsvariable `$LANG` muß auf eine Sprache gesetzt sein, wenn sie nicht standardmäßig vorbelegt ist. Wenn Sie englische Übersetzermeldungen erhalten wollen, ist eine gültige Vorbesetzung `En_US.ASCII`.

Bei Ablage eines Message-Katalogs unter einem anderen als dem default-Pfad `/opt/lib/nls/msg/$LANG/ces.cat` muß dieser über die Umgebungsvariable `$NLSPATH` bekannt gemacht werden.

Vorsicht

Der CES-C-Übersetzer ist der Übersetzer des integriereten C-Entwicklungssystems für die SINIX-Rechnerfamilie der Siemens Nixdorf Informationssysteme AG. Alternativ kann auch das C-Entwicklungssystem von AT&T mit dem C-Übersetzer PCC genutzt werden. Daher sind unter SINIX bei installiertem CES zwei C-Übersetzer und damit zwei `cc`-Kommandos vorhanden:

- Den CES-C-Übersetzer rufen Sie wie folgt auf:

```
$ cc
```

- Den PCC-Übersetzer rufen Sie wie folgt auf:

```
$ /usr/ccs/bin/cc
```

Im folgenden ist die Syntax des CES-`cc`-Kommandos beschrieben.

Objektkonformität

CES V5.41 erzeugt ABI386 (**A**pplication **B**inary **I**nterface) -konforme Objekte, insbesondere im Hinblick auf:

- das erzeugte Objektformat (ELF)
- die unterstützten Datentypen (Integral, Pointer, Floatingpoint, Bitfeld)
- die Ausrichtung an "half-word"(16 Bit)-, "word"(32 Bit)- und "double-word"(64 Bit)-Grenzen
- die Objektgrößen
- die Organisation der Datenobjekte in Datenstrukturen

Durch diese Objektkonformität ist es möglich, die mit dem CES-Übersetzer erzeugten Objekte mit anderen vom PCC-Übersetzer erzeugten Objekten zu einem ablauffähigen Programm zu verbinden.

Das Objektformat wird mit ELF-Symbolinformation erzeugt, so daß CES-Programme mit dem symbolischen Debugger DBX getestet werden können.

cc-Syntax

`cc`[␣*option*]...␣*datei*␣... [␣`-l`*archivkuerzel*]...

option

Durch Angabe von Optionen im `cc`-Aufruf können Sie den Übersetzungsablauf steuern und beeinflussen, mit welchen Argumenten die Programme für die einzelnen Übersetzungsphasen versorgt werden. Die Optionen in alphabetischer Reihenfolge sind:

`-B, -c, -C, -d, -D, -E, -F, -g, -G, -h, -I, -k, -KPIC, -KFCW, -l, -L, -LD, -o, -O, -p, -P, -qp, -S, -U, -V, -W, -X.`

Keine *option* angegeben

Sind die Dateiinhalte der angegebenen Dateien syntaktisch korrekt und können alle offenen Referenzen aufgelöst werden, erstellt `cc` eine ausführbare Datei `a.out`, die das ablauffähige Programm enthält. Sind mehrere Dateien angegeben, dann speichert `cc` den Objektcode zu den einzelnen Dateien zusätzlich in gleichnamigen `.o`-Dateien ab.

datei

Name der Eingabedatei. Es muß mindestens eine Nicht-Bibliotheksdatei angegeben werden.
`cc` schließt aus der Endung des Dateinamens auf den Dateiinhalt und führt die jeweils noch erforderlichen Übersetzungsschritte aus. Der Dateiname muß daher das Suffix haben, das gemäß den SINIX-Konventionen zu dem Dateiinhalt paßt. Im einzelnen gibt es folgende Möglichkeiten:

Suffix	Dateiinhalt
.c	C-Quellcode vor Präprozessorlauf
.i	C-Quellcode nach Präprozessorlauf
.s	Assembler-Quellcode
.o	Objektcode
.a	Bibliothek mit Objektmodulen
.so	mehrfach benutzbare Bibliothek mit Objektmodulen

Ein Dateiname darf nicht mit einer Ziffer und nicht mit einem Punkt . beginnen.

`-l`*archivkuerzel*

Die Option `-l` hat eine Sonderstellung, da die `-l`-Angaben als letzte im `cc`-Aufruf stehen sollten.

Allgemeine cc-Optionen

`-V`

(`V` - Verbose) Während der Ausführung von `cc` und den von `cc` aufgerufenen Übersetzungsphasen werden Meldungen über den Ablauf auf die Standardfehlerausgabe geschrieben.

`-W[`*n*`]`

(`W` - Warning) Mit dieser Option können Sie festlegen, in welchem Umfang Warnungen ausgegeben werden.

n kann sein: `0`, `1` oder `2`.

`0`	Alle Warnungen werden unterdrückt.
`1`	Nur bestimmte, wichtige Warnungen werden ausgegeben.
`2`	Alle Warnungen werden ausgegeben.

Die bei *n* = `2` zusätzlich ausgegebenen Warnungen betreffen i.a. interne Abläufe, die für ein Anwendungsprogramm ohne Bedeutung sind.

`-W[`*n*`]` nicht angegeben:
nur bestimmte, wichtige Warnungen werden ausgegeben (wie `-W1`)
`-W` ohne *n* angegeben:
Es gilt *n* = `2`.

Optionen, die den Übersetzungsablauf steuern

`-P` (`P` - Preprocessor) Nur der Präprozessor wird aufgerufen. `cc` wird nach dem Präprozessor-Lauf angehalten. Das Ergebnis zu einer Datei *datei*`.c` wird in eine Datei *datei*`.i` geschrieben. Die Datei *datei*`.i` kann mit `cc` weiter übersetzt werden.

`-E` Wie bei der Option `-P` wird nur der Präprozessor aufgerufen. Das Ergebnis wird aber auf die Standardausgabe statt in eine Datei geschrieben.

`-S` Die angegebenen Dateien werden in Assembler-Quellcode übersetzt. Danach stoppt `cc`. Der Assembler-Quellcode zu einer Datei *datei*`.c` wird in die Datei *datei*`.s` geschrieben.

`-X` Wie bei der Option `-S` stoppt `cc` nach der Erzeugung des Assemblercodes. Zusätzlich wird der C-Quellcode in Form von Kommentarzeilen in die Datei *datei*`.s` geschrieben.

`-c` Die angegebenen Dateien werden in Objektcode übersetzt. Danach wird `cc` gestoppt. Der Objektcode zu einer Datei *datei*`.c` wird in die Datei *datei*`.o` geschrieben.

Präprozessor-Optionen

`-C` C-Kommentare `/* ... */` werden nicht entfernt.

`-D`*name*`[=`*wert*`]`
Die symbolische Konstante *name* wird definiert wie durch die Präprozessor-Anweisung `#define`. Wird auch *wert* angegeben, so entspricht dies `#define` *name wert*.

`-I`*dv*
dv wird in die Liste der Dateiverzeichnisse aufgenommen, in denen der Präprozessor nach Include-Dateien sucht.
Bei Include-Dateien, deren relativer Dateiname (beginnt nicht mit Schrägstrich /) in Anführungszeichen "..." eingeschlossen ist, durchsucht der Präprozessor die Dateiverzeichnisse in folgender Reihenfolge:

1. das Dateiverzeichnis der Quelldatei, die die `#include`-Anweisung enthält
2. die Dateiverzeichnisse, die mit der Präprozessor-Option `-I` angegeben wurden
3. das Dateiverzeichnis `/usr/include`.

Bei Include-Dateien, deren relativer Dateiname in spitzen Klammern <...> eingeschlossen ist, durchsucht der Präprozessor die Dateiverzeichnisse in folgender Reihenfolge:

1. die Dateiverzeichnisse, die mit der Präprozessor-Option `-I` angegeben wurden
2. das Dateiverzeichnis `/usr/include`.

Ausnahme

Enthält eine Include-Datei selbst auch eine Include-Anweisung ohne spitze Klammern <...>, durchsucht der Präprozessor die Dateiverzeichnisse in folgender Reihenfolge:

1. das Dateiverzeichnis der Include-Datei
2. die Dateiverzeichnisse, die mit der Präprozessor-Option `-I` angegeben wurden.

`-U`*name*
Die Definition für ein Makro bzw. eine symbolische Konstante *name* wird gelöscht wie durch die Präprozessor-Anweisung `#undef`.

Kommt *name* in der Eingabedatei nicht vor, wird die Option `-U` ignoriert.

Enthält die Eingabedatei `#define`-Anweisungen für *name*, hat die Option `-U` keine Wirkung.

name
Vordefinierter Präprozessor-Name oder ein Name, der **vor** dem Präprozessor-Aufruf definiert wurde durch:

- Angabe einer `-D`-Option vor der `-U`-Option in demselben `cc`-Aufruf
- eine `#define`-Anweisung von `cc`.
  Die von `cc` definierten Namen erfahren Sie, wenn Sie `cc` mit der Option `-V` aufrufen.

`-U`*name* nicht angegeben:
Es gelten nur die vom System vordefinierten Namen.

Folgende Namen sind vordefiniert:

- vordefinierte Makros des Präprozessors

`__DATE__`	aktuelles Datum, Anpassung an ANSI-C-Konvention: "Mmm dd yyyy"
`__FILE__`	aktuelle Eingabedatei
`__LINE__`	aktuelle Zeilennummer
`__TIME__`	aktuelle Uhrzeit
`__STDC__`	=0 für K&R-Modus =1 für ANSI-C-Modus

- von `cc` definierte symbolische Konstanten, die ausgegeben werden, wenn Sie `cc` mit der Option `-V` aufrufen:

im K&R-Modus	im ANSI-C-Modus
`#define i386`	`#define i80386`
`#define i80386`	`#define sinix`
`#define unix`	`#define SNI`
`#define sinix`	`#define M_I386`
`#define SNI`	`#define __HOST__="i80386.sinix"`
`#define M_I386`	`#define __TARG__="i80386.sinix"`
`#define _XOPEN_SOURCE`	
`#define __HOST__="i80386.sinix"`	
`#define __TARG__="i80386.sinix"`	

Übersetzer-Optionen

-k[*modus*]

Angabe des Sprachmodus.
Wird die Option -k nicht angegeben, verwendet der Übersetzer den K&R-Modus, so wie er in der ersten Ausgabe "The C Programming Language" von Kernighan und Ritchie beschrieben ist (__STDC__=0).

modus kann sein: ansi oder cc

ansi — alle ANSI-C Syntax- und Semantik-Konstrukte werden akzeptiert und __STDC__ wird auf 1 gesetzt.

cc — K&R-Modus.
Die C-Definition gemäß Kernighan&Richie wird verwendet.

modus nicht angegeben:
es gilt der K&R-Modus (__STDC__=0)

Vorsicht

Programme können unter Umständen verschiedene Ergebnisse liefern, wenn sie in unterschiedlichen Sprachmodi übersetzt wurden. Bei ANSI-C gelten beispielsweise andere Regeln für die implizite Typumwandlung als im K&R-Modus. Ganzzahlige Werte in Ausdrücken werden so umgewandelt, daß Vorzeichen und Wert erhalten bleiben ("sign preserving"- und "value preserving"-Regel). Bei der C-Definition nach Kernighan&Ritchie wird ein Ausdruck insgesamt in unsigned umgewandelt, sobald ein unsigned-Operand in dem Ausdruck vorkommt ("unsigned preserving" -Regel).

Weitere Implementierungsspezifika im ANSI-C-Sprachmodus

Der Datentyp long double wird syntaktisch vom Übersetzer akzeptiert, wird aber auf den Datentyp double abgebildet.

In Zeichenkonstanten und Kommentaren werden keine Multibyte-Zeichensätze unterstützt. Der CES-Übersetzer erlaubt die Nutzung der 7 bzw. 8 Bit Zeichensätze ISO646 bzw. ISO8859-1.

Bitfelder können als Datenobjekte der Typen char, short, int oder long gebildet werden. Diese werden - auch im K&R-Modus - als unsigned Datenobjekte vom Übersetzer verarbeitet.

-p (p - profile) Der Übersetzer erzeugt zusätzlich für jede Funktion Code, der mitzählt, wie oft die Funktion aufgerufen wird. Führen Sie das übersetzte Programm aus, wird bei erfolgreicher Beendigung des Programms eine Profildatei mon.out erzeugt, die die Zählerwerte enthält. Mit dem Kommando prof können Sie aus der Profildatei ein Laufzeitprofil des Programms erstellen. prof liefert CPU-Zeit und Aufrufstatistik für alle Funktionen des Programms, die mit der Übersetzeroption -p übersetzt wurden.

-qp Diese Option hat die gleiche Wirkung wie -p.

-g Der Übersetzer speichert Informationen für symbolische Testhilfen. Folgende Informationen werden zusätzlich in der Objektdatei abgelegt:

- Für jede Variable:
  - Name, Datentyp und Adresse
- Für jede Funktion:
  - Name, Datentyp und Adresse
  - Name, Datentyp und Adresse im Stackrahmen von allen Parametern und lokalen Variablen.
- Für jede Quellcodezeile:
  - Anfangsadresse der Maschinenbefehle, die die Übersetzung dieser Zeile darstellen.

Zusammenwirken der Optionen:

Geben Sie in einem Aufruf gleichzeitig die Optionen -g und -O an, dann setzt die Option -g die Option -O außer Kraft.

Geben Sie gleichzeitig die Binder-Option -s an, ist die Option -g wirkungslos. Dadurch entstehen Einschränkungen beim Testen mit symbolischen Testhilfen. Da die Symboltabellen mit der Binder-Option -s entfernt werden, kann nur noch auf Maschinencode-Ebene getestet werden.

Optionen zur Optimierung und Codegenerierung

`-O` (`O` - Optimizer) `cc` ruft zusätzlich ein Programm auf, das die Optimierungen durchführt. Üblicherweise können Sie damit die Größe der Objektdatei reduzieren und die Ausführungszeit verkürzen. Allerdings wird die Übersetzungszeit erhöht.
Mit `-OO` schalten Sie jegliche Optimierung aus.

Zusammenwirken der Optionen:
Wenn Sie in einem Aufruf gleichzeitig die Option `-O` und die Option `-F[`*unteroption*`]` angeben, schalten Sie damit die Standardoptimierungen und die durch *unteroption* ausgewählten speziellen Optimierungen ein. Die Standardoptimierungen umfassen die Optionen `-ULPCc`.

`-KPIC` oder `-Kpic`
Diese Option steuert dieGenerierung von positionsunabhängigem Code (PIC).
Standardmäßig wird kein PIC-Code erzeugt.

Ist der PIC-Code mit der option `-KPIC` eingeschaltet, dann werden globale Daten indirekt adressiert. Der generierte Code kann mit mehrfach benutzbaren Objekten gebunden werden. Voraussetzung ist dynamisches Binden (`-dy`), dies ist die Standardeinstellung.

Nur die Bibliotheken `libc.so` und `libdl.so` stehen in einer PIC-Version zur Verfügung.

`-F[`*unteroption*`]...`
Mit der Option `-F` werden spezielle Optimierungsmaßnahmen eingeschaltet und die Codegenerierung gesteuert.
Es können mehrere Unteroptionen gleichzeitig angegeben werden.
Die Reihenfolge der Unteroptionen entspricht der Reihenfolge, in der die Optimierungen durchgeführt werden.
Wird keine *unteroption* angegeben, dann werden keine Optimierungen aktiviert.

Für *unteroption* kann folgendes angegeben werden:

`C, c, e, L, o, P, s, U.`

`C` (`C`- Commoning) Gemeinsame Teilausdrücke werden entfernt. Zusätzlich werden nicht erreichbare Codeteile eliminiert und aufeinanderfolgende Sprünge optimiert.

`c` (`c` - commoning) Wirkt wie `C`, jedoch werden die gemeinsamen Teilausdrücke auf Assembler-Ebene entfernt.

`e` (`e` - execution) Retten und Laden des Registerkontexts werden auf ein Minimum beschränkt. Vor Funktionsaufruf werden nur die Register gerettet, die für den Funktionsablauf benötigt werden. Nur diese Register werden bei Rückkehr aus der Funktion wieder geladen.

`L` (`L` - Loop) Die Indexrechnung in Schleifen wird optimiert. Bei der Berechnung der Adresse eines Array-Elements werden Multiplikationen auf Additionen zurückgeführt.
Zusätzlich werden nicht erreichbare Codeteile entfernt und aufeinanderfolgende Sprünge optimiert.

`o` (`o` - overflow) Der Codegenerator erzeugt Testcode, der bei Operationen mit Variablen des Datentyps `signed int` einen Überlauf erkennt und den Programmlauf mit einer Fehlermeldung beendet.

`P` (`P` - Propagation) Konstante Ausdrücke werden ersetzt. Kommen in Ausdrücken Variablen vor, deren Wert zum Zeitpunkt der Übersetzung bekannt ist, werden sie durch diesen Wert ersetzt.
Zusätzlich werden nicht erreichbare Codeteile entfernt und aufeinanderfolgende Sprünge optimiert.

s (s - stack) Auf die dynamische Verkettung der Stackrahmen wird verzichtet. Diese Code-Optimierung verschlechtert die Möglichkeit, Objektmodule mit Testhilfen zu testen, da keine Information für die Rückverfolgung der dynamischen Aufrufhierarchie mehr zur Verfügung steht.
Diese Option darf bei Sprachgemischen mit Pascal-XT und Fortran77 nicht verwendet werden.

U (U - Unused Code) Nicht benötigte Codeteile werden entfernt. Dies können Zuweisungen an Variablen sein, die im weiteren Programmlauf nicht mehr verwendet werden.
Zusätzlich werden aufeinanderfolgende Sprünge optimiert.

`-FI[`*name*`]`

(I - Inline) Die Funktion *name* wird in den Programmtext eingebunden. Dies erspart den Aufwand für den Funktionsaufruf und die Funktionsrückkehr. Diese Optimierung ist bei häufig verwendeten Laufzeitfunktionen sinnvoll.

name ist der Name einer der folgenden Funktionen, die bei CES eingebunden werden können:

```
abs(), acos(), asin(), atan(), atan2(), cos(), exp(), fabs(),
hypot(), log(), log10(), memcmp(), memcpy(), memset(), pow(),
sin(), sqrt(), strcat(), strcmp(), strcpy(), strlen(), tan().
```

name nicht angegeben:
Alle bei CES verfügbaren Inline-Funktionen werden eingebunden.

`-KFCW[␣]`*compiler*

Mit der Option `-KFCW` wird das Floating Point Control Word des arithmetischen Coprozessors beeinflußt.

Wird die Option `-KFCW` nicht angegeben, ist die Standardeinstellung das CES-kompatible Rundungsverhalten (`-KFCWCES`).

compiler kann sein: `CES`, `ces`, `ATT`, `att`, `CESN` oder `cesn`.

`CES` oder `ces`
Das Floating Point Control Word wird in CES-kompatibler Weise gesetzt: Das Rundungsverhalten wird auf "round to zero" eingestellt, und die Ereignisbehandlung wird für "overflow" und "zero-divide" aktiviert.

`ATT` oder `att`
Das Floating Point Control Word wird in PCC-kompatibler Weise gesetzt: Das Rundungsverhalten wird auf "round to nearest" eingestellt, und es wird keine Ereignisbehandlung durchgeführt.

`CESN` oder `cesn`
Das Floating Point Control Word wird in CES-kompatibler Weise gesetzt: Das Rundungsverhalten wird auf "round to nearest" eingestellt, und die Ereignisbehandlung wird für "overflow" und "zero-divide" aktiviert.

Binder-Optionen

Die für das `ld`-Kommando definierten Optionen können Sie auch wie folgt im `cc`-Aufruf angeben:

- Die `ld`-Optionen `-B`, `-d`, `-G`, `-h`, und `-L` können Sie direkt angeben.
- Die Optionen `-o` und `-V` können Sie auch direkt angeben. Sie haben aber auch für andere Übersetzungsphasen eine Bedeutung. Sind sie im `cc`-Aufruf direkt angegeben, gelten sie für alle durchlaufenen Phasen, für die sie definiert sind.
- Über die Option `-LD` können Sie jede beliebige `ld`-Option an `ld` weiterreichen. Eine bei `-LD` aufgeführte Option gilt nur für `ld`.

`-Bdynamic`

Die Option `-Bdynamic` veranlaßt den Binder `ld`, ab der Angabe dieser Option auf der Aufrufzeile **nur** nach dynamischen Bibliotheken (`libx.so`) zu suchen. Dieses wird bis zum Ende der Aufrufzeile oder bis zum Auftreten einer `-Bstatic`-Option fortgesetzt.

Zusammenwirken der Optionen:
Sie dürfen nicht gleichzeitig die Optionen `-Bdynamic` und `-dn` angeben.

`-Bstatic`

Die Option `-Bstatic` veranlaßt den Binder `ld`, ab der Angabe dieser Option auf der Aufrufzeile **nur** nach statischen Bibliotheken (`libx.a`) zu suchen. Dieses wird bis zum Ende der Aufrufzeile oder bis zum Auftreten einer `-Bdynamic`-Option fortgesetzt.

Zusammenwirken der Optionen:
Sie dürfen nicht gleichzeitig die Optionen `-Bstatic` und `-G` angeben. Objekte, die mit `-G` in eine mehrfach benutzbare Bibliothek eingebracht werden sollen, müssen mit `-KPIC` bzw. `-Kpic` erzeugt werden.

`-Bsymbolic`

Angabe, wie bei dynamischem Binden offene Referenzen aufgelöst werden.
Enthält ein Objektmodul Definitionen für globale Symbole, löst `ld` die offenen Referenzen auf diese Symbole schon beim Erzeugen des mehrfach benutzbaren Objektmoduls auf. Auf diese Weise wird, falls vorhanden, immer die objekteigene Definition für ein Symbol verwendet.
Normalerweise werden Referenzen zu globalen Symbolen innerhalb mehrfach benutzbarer Objektmodule erst zur Ausführungszeit aufgelöst, auch wenn die Definitionen vorhanden sind. Dadurch kann es vorkommen, daß statt der objekteigenen Definition für ein Symbol eine Definition verwendet wird, die in einem zuvor angegebenen ausführbaren Programm oder Objektmodul steht. Mit `-Bsymbolic` kann dies verhindert werden.

Zusammenwirken der Optionen:
Sie dürfen nicht gleichzeitig die Optionen `-Bsymbolic` und `-dn` angeben.
Die Option `-Bsymbolic` wirkt nur bei gleichzeitiger Angabe von `-KPIC` bzw. `-Kpic`.

`-d`*o*

Globale Einstellung von `ld` für dynamisches oder statisches Binden.

o kann sein: `y` oder `n`.

`y` Dynamisch Binden.
Die Objektmodule werden erst zur Ausführungszeit gebunden.

`n` Statisch Binden.
Die Objektmodule werden schon zur Übersetzungszeit gebunden.

Zusammenwirken der Optionen:
Sie dürfen nicht gleichzeitig die Optionen `-Bdynamic` und `-dn` angeben.

`-d` nicht angegeben: Es gilt *o* = `y`.

`-G` Mehrfach benutzbares Objekt erzeugen.

Zusammenwirken der Optionen:
Sie dürfen nicht gleichzeitig die Optionen `-G` und `-dn` bzw. `-Bstatic` angeben.

-h*name*

Für das erzeugte mehrfach benutzbare Objektmodul wird der Name *name* verwendet. *name* wird in den Abschnitt für dynamisches Binden (`.dynamic section`) der Objektdatei und in alle Programme, die mit dem Objektmodul gebunden werden, eingetragen. Der dynamische Binder sucht nach *name*, wenn er externe Referenzen auflösen will.

Zusammenwirken der Optionen:
Die Option -h*name* ist nur zusammen mit der Option `-G` wirksam.

-LD["]*ld_option*["]

Binder-Option angeben.
Für *ld_option* können Sie jede beliebige `ld`-Option angeben.
Sie müssen die `-LD`-Angabe in Anführungszeichen setzen, wenn *ld_option* Leerzeichen enthält.
Die `ld`-Optionen finden Sie bei der `ld`-Beschreibung im "Referenzhandbuch für Programmierer".

-L*dv*

Weiteres Suchverzeichnis angeben.
Diese Option wird vor allem zusammen mit der `-l`-Option verwendet.
Mit *dv* wird ein zusätzliches Dateiverzeichnis angegeben, in dem der Binder nach der mit der Option `-l` angegebenen Bibliothek suchen soll.
Die so angegebenen Verzeichnisse werden noch vor den Standard-Bibliotheksverzeichnissen nach den angegebenen Bibliotheken durchsucht.

-o *ausgabedatei*

(`o` - output) Wird ein ablauffähiges Programm erstellt oder ist die Option `-c` gesetzt und wird nur ein Objektmodul erstellt, dann heißt die Ausgabedatei *ausgabedatei*.

ausgabedatei

Name der Datei, in die `cc` das Ergebnis schreibt.

-l*archivkuerzel*

`ld` bindet die Bibliothek mit dem Kürzel *archivkuerzel* dazu.
Standardmäßig durchsucht `ld` folgende Dateiverzeichnisse in der angegebenen Reihenfolge nach der Bibliothek:

- `/usr/ccs/lib`
- `/usr/lib`
- `/usr/ces/lib`
- `/opt/lib`

Wird dynamisch gebunden, sucht `ld` zuerst nach der dynamischen Version der Bibliothek mit dem Namen `lib`*archivkuerzel*`.so`.
Wird statisch gebunden, sucht `ld` nur die statische Version und bindet diese dazu.

archivkuerzel

Statt des absoluten Dateinamens `lib`*archivkuerzel*`.a` bei statischem Binden bzw. `lib`*archivkuerzel*`.so` bei dynamischem Binden geben Sie mit *archivkuerzel* nur den für die Bibliothek spezifischen Teil an.

Haben Sie mit der Option -L*dv* ein Dateiverzeichnis angegeben, das zusätzlich nach Bibliotheken durchsucht werden soll, müssen Sie eine Bibliothek aus diesem Dateiverzeichnis mit der Option `-l` in der Form -l*archivkuerzel* angeben. Die Bibliothek muß unter dem Namen `lib`*archivkuerzel*`.so` oder `lib`*archivkuerzel*`.a` im Dateiverzeichnis vorhanden sein.

`-l` nicht angegeben:

`cc` ruft den Binder immer mit der Option `-lc` auf, so daß automatisch die Standard-C-Bibliothek `/usr/ccs/lib/libc.so` bzw. `/usr/ccs/lib/libc.a` dazugebunden wird.

Dateien und Dateiverzeichnisse

datei.c	Eingabedatei C-Quellcode eventuell mit Präprozessor-Anweisungen
datei.i	Ausgabedatei des Präprozessors C-Quellcode ohne Präprozessor-Anweisungen
datei.s	Ausgabedatei des Übersetzers Assemblercode
datei.o	Ausgabedatei des Assemblers Objektcode im ELF-Format
`a.out`	Ausführbare Datei
`/var/tmp/*`	Zwischendateien
`/usr/ces/bin/cc`	Steuerprogramm für den Übersetzer
`/usr/ces/bin/c01`	Scanner, Präprozessor und Parser
`/usr/ces/bin/oc23`	Optimierer, Codegenerator und Assembler
`/usr/ces/lib/libfm.a` `/usr/ces/lib/libfm.so`	Optimierte mathematische Bibliotheken Diese Bibliotheken können mit der Option `-lfm` anstelle der Standardbibliothek für mathematische Funktionen `libm.a` dazugebunden werden.
`/usr/ces/lib/libp/libfmp.a`	wie `libfm.a`, jedoch mit Profiler-Informationsausgabe generiert
`/opt/lib/nls/msg/En/ces.cat`	Englischer Message-Katalog
`/opt/lib/nls/msg/De/ces.cat`	Deutscher Message-Katalog