arm-none-eabi-objcopy 使用說明

譯文

NAME

objcopy - copy and translate object files

概要

objcopy [**-F** bfdname|**--target=**bfdname]
[**-I** bfdname|**--input-target=**bfdname]
[**-O** bfdname|**--output-target=**bfdname]
[**-B** bfdarch|**--binary-architecture=**bfdarch]
[**-S**|**--strip-all**]
[**-g**|**--strip-debug**]
[**-K** symbolname|**--keep-symbol=**symbolname]
[**-N** symbolname|**--strip-symbol=**symbolname]
[**--strip-unneeded-symbol=**symbolname]
[**-G** symbolname|**--keep-global-symbol=**symbolname]
[**--localize-hidden**]
[**-L** symbolname|**--localize-symbol=**symbolname]
[**--globalize-symbol=**symbolname]
[**-W** symbolname|**--weaken-symbol=**symbolname]
[**-w**|**--wildcard**]
[**-x**|**--discard-all**]
[**-X**|**--discard-locals**]
[**-b** byte|**--byte=**byte]
[**-i** [breadth]|**--interleave**[=breadth]]
[**--interleave-width=**width]
[**-j** sectionpattern|**--only-section=**sectionpattern]
[**-R** sectionpattern|**--remove-section=**sectionpattern]
[**-p**|**--preserve-dates**]
[**-D**|**--enable-deterministic-archives**]
[**-U**|**--disable-deterministic-archives**]
[**--debugging**]
[**--gap-fill=**val]
[**--pad-to=**address]
[**--set-start=**val]
[**--adjust-start=**incr]
[**--change-addresses=**incr]
[**--change-section-address** sectionpattern{=,+,-}val]
[**--change-section-lma** sectionpattern{=,+,-}val]

[**--change-section-vma** sectionpattern{=,+,-}val]
[**--change-warnings**] [**--no-change-warnings**]
[**--set-section-flags** sectionpattern=flags]
[**--add-section** sectionname=filename]
[**--dump-section** sectionname=filename]
[**--update-section** sectionname=filename]
[**--rename-section** oldname=newname[,flags]]
[**--long-section-names** {enable,disable,keep}]
[**--change-leading-char**] [**--remove-leading-char**]
[**--reverse-bytes=**num]
[**--srec-len=**ival] [**--srec-forceS3**]
[**--redefine-sym** old=new]
[**--redefine-syms=**filename]
[**--weaken**]
[**--keep-symbols=**filename]
[**--strip-symbols=**filename]
[**--strip-unneeded-symbols=**filename]
[**--keep-global-symbols=**filename]
[**--localize-symbols=**filename]
[**--globalize-symbols=**filename]
[**--weaken-symbols=**filename]
[**--add-symbol** name=[section:]value[,flags]
[**--alt-machine-code=**index]
[**--prefix-symbols=**string]
[**--prefix-sections=**string]
[**--prefix-alloc-sections=**string]
[**--add-gnu-debuglink=**path-to-file]
[**--keep-file-symbols**]
[**--only-keep-debug**]
[**--strip-dwo**]
[**--extract-dwo**]
[**--extract-symbol**]
[**--writable-text**]
[**--readonly-text**]
[**--pure**]
[**--impure**]
[**--file-alignment=**num]
[**--heap=**size]
[**--image-base=**address]
[**--section-alignment=**num]

[**--stack=**size]
[**--subsystem=**which:major.minor]
[**--compress-debug-sections**]
[**--decompress-debug-sections**]
[**--elf-stt-common=**val]
[**-v**|**--verbose**]
[**-V**|**--version**]
[**--help**] [**--info**]
infile [outfile]

描述

GNU **objcopy** 實用程序將一個對象文件的內容復制到另一個文件中。 **objcopy** 使用 GNU BFD 庫讀取和寫入目標文件。它可以寫目的地
目標文件的格式與源目標文件的格式不同。確切行為
的 **objcopy** 把由命令行選項來控制。請注意，**objcopy** 應該能夠
在任何兩種格式之間復制完全鏈接的文件。但是，復制可重定位的對象
兩種格式之間的文件可能無法正常工作。

objcopy 創建臨時文件以進行翻譯，然後將其刪除。
objcopy 使用 BFD 進行所有翻譯工作；它可以訪問所有格式
在 BFD 中進行了描述，因此無需明確說明就可以識別大多數格式。

objcopy 可用於通過使用 **srec** 的輸出目標（例如，使用**-O srec**）來生成 S 記錄。

objcopy 可用於通過使用**二進制**輸出目標
（例如，使用**-O binary**）來生成原始二進制文件。當 **objcopy** 生成原始二進制文件時，它將基本上
產生輸入目標文件內容的內存轉儲。所有符號和
重定位信息將被丟棄。內存轉儲將從加載地址開始
復制到輸出文件的最低部分的。

生成 S 記錄或原始二進制文件時，使用**-S** 刪除可能會有所幫助
包含調試信息的部分。在某些情況下，**-R** 對刪除很有用
包含二進制文件不需要的信息的部分。

注意--- **objcopy** 無法更改其輸入文件的字節序。如果輸入
格式具有字節序（某些格式沒有），**objcopy** 只能將輸入復制到文件中
具有相同字節序或不具有字節序的格式（例如 **srec**）。（然而，
請參閱**--reverse-bytes** 選項。）

選項

infile
outfile
分別是輸入文件和輸出文件。如果未指定 outfile，則 **objcopy 將**
創建一個臨時文件，並以 infile 的名稱破壞性地重命名結果。

-I bfdname
--input-target = bfdname
認為源文件的對象格式為 bfdname，而不是嘗試
推論它。

-O bfdname
--output-target = bfdname
使用對象格式 bfdname 寫入輸出文件。

-F bfdname
--target = bfdname
使用 bfdname 作為輸入文件和輸出文件的對象格式；即，簡單地
無需翻譯即可將數據從源傳輸到目標。

-B bfdarch
--binary-architecture = bfdarch
將無架構的輸入文件轉換為目標文件時很有用。在這個
可以將輸出體系結構設置為 bfdarch 的情況。如果此選項將被忽略
輸入文件具有已知的 bfdarch。您可以在程序中訪問此二進制數據
通過引用轉換過程中創建的特殊符號。這些
符號稱為_binary_ objfile _start，_binary_ objfile _end 和
binary objfile _size。例如，您可以將圖片文件轉換為目標文件，然後
然後使用這些符號在您的代碼中對其進行訪問。

-j sectionpattern
--only-section = sectionpattern
僅將指示的部分從輸入文件復制到輸出文件。這個選項
可能會被多次給予。請注意，不當使用此選項可能會使
輸出文件不可用。sectionpattern 中接受通配符。

-R sectionpattern
--remove 截面= sectionpattern
刪除任何部分匹配 sectionpattern 從輸出文件。此選項可能是
給予不止一次。請注意，不當使用此選項可能會使輸出

文件無法使用。sectionpattern 中接受通配符。同時使用**-j**
和**-R** 選項會導致未定義的行為。

-S-
全部
不要從源文件復制重定位和符號信息。

-g
--strip-調試
不要從源文件中復制調試符號或節。

-不需要條紋
去除重定位處理不需要的所有符號。

-K symbolname
--keep-symbol = symbolname
剝離符號時，即使通常會剝離，也要保留符號 symbolname。
可以多次給此選項。

-N symbolname
--strip-symbol = symbolname
不復制源文件中的 symbol symbolname。可以給這個選項更多
不止一次。

--strip-unneeded-symbol = symbolname 除非源文件需要，否則
不要從源文件中復制符號 symbolname。
搬遷。可以多次給此選項。

-G symbolname
--keep-global-symbol = symbolname
僅保留全局符號 symbolname。使所有其他符號位於文件本地，以便
它們在外部不可見。可以多次給此選項。

--localize-hidden
在 ELF 對象中，將所有具有隱藏或內部可見性的符號標記為局部。
此選項適用於特定於符號的本地化選項（例如**-L**）。

-L symbolname
--localize-symbol = symbolname
將符號 symbolname 設置為文件本地，以使其在外部不可見。這

選項可能會多次給出。

-W symbolname
--weaken-symbol = symbolname
使符號 symbolname 變弱。可以多次給此選項。

--globalize-symbol = symbolname
賦予符號 symbolname 全局作用域，以便在文件的外部可見
它是定義的。可以多次給此選項。

-w
--wildcard
允許在其他命令行選項中使用的 symbolname 中的正則表達式。這
問號（？），星號（*），反斜槓（\）和方括號（[]）運算符可以
可在符號名稱中的任何位置使用。如果符號名稱的第一個字符是
感嘆號（！），則該符號的開關方向相反。為了
例子：

```
-w -W !foo -W fo *
```

會導致 objcopy 削弱所有以“ fo”開頭的符號，除了該符號
“ foo”。

-x
--discard，所有
不要從源文件復制非全局符號。

-X
--discard-locals
不要復制編譯器生成的本地符號。（這些通常以 **L** 或。開頭。）

-b 字節
--byte =字節
如果通過**--interleave** 選項啟用了**交錯**，則開始范圍
字節保持在字節個字節。 字節的范圍可以從 **0** 到寬度**-1**，其中
寬度是**--interleave** 選項給出的值。

-i [廣度**]**
--interleave [=廣度**]**
只復制的范圍超出每廣度字節。（標題數據不受影響）。選擇

范圍中的哪個字節以**--byte** 選項開始復制。選擇寬度**--interleave-width** 選項的范圍。

此選項對於創建要編程 ROM 的文件很有用。通常與“ srec”輸出目標。請注意，如果您也未指定**--byte** 選項，則 **objcopy** 將抱怨。

默認的交錯寬度為 4，因此當**--byte** 設置為 0 時，**objcopy** 將復制從輸入到輸出的每四個字節中的第一個字節。

--interleave-width = width

與**--interleave** 選項一起使用時，一次復制寬度字節。的開始要復制的字節范圍由**--byte** 選項設置，范圍的范圍使用**--interleave** 選項設置。

此選項的默認值是 1 的值寬度加上字節值集合通過該**--byte** 選項必須不被超過交錯廣度集**--interleave** 選項。

此選項可用於為兩個交錯的 16 位閃光燈創建圖像。通過將**-b 0 -i 4 --interleave-width = 2** 和**-b 2 -i 4 --interleave-width = 2** 傳遞給兩個 **objcopy** 命令來實現 32 位總線。如果輸入為“ 12345678”，則輸出分別為“ 1256”和“ 3478”。

-p

--preserve-日期

將輸出文件的訪問和修改日期設置為與輸入文件。

-D-

啟用確定性歸檔

以確定性模式運行。復制檔案成員並寫入檔案時索引，將 UID，GID，時間戳記為零，並將所有文件使用一致的文件模式文件。

如果 binutils 配置了**--enable-deterministic-archives**，則此模式為開默認情況下。可以使用下面的**-U** 選項禁用它。

-U

--disable 確定性的存檔

千萬不能在運行確定性模式。這與上面的**-D** 選項相反。
復制存檔成員並寫入存檔索引時，請使用其實際的 UID，GID，
時間戳記和文件模式值。

這是默認設置，除非 binutils 配置了
--enable-deterministic-archives。

- 調試

盡可能轉換調試信息。這不是默認值，因為
支持某些調試格式，轉換過程可能很耗時
消耗。

--gap 填充 VAL

與段之間填補空白 VAL。此操作適用於加載 地址（LMA）
部分。這是通過增加該部分的大小來降低的
地址，並填充使用 val 創建的額外空間。

--pad-to address

將輸出文件填充到加載地址 address。這是通過增加
最後一節的大小。多餘的空間用以下值指定
--gap-fill（默認為零）。

--set-start val

將新文件的起始地址設置為 val。並非所有目標文件格式都支持
設置起始地址。

--change-start incr
--adjust-start incr

通過添加 incr 更改起始地址。並非所有目標文件格式都支持設置
起始地址。

--change- addresss 增量
--adjust-vma 增量

通過以下方法更改所有部分的 VMA 和 LMA 地址以及起始地址
添加 incr。某些目標文件格式不允許更改節地址
任意地。請注意，這不會重定位這些部分；如果程序期望
節將加載到某個地址，並且此選項用於更改
如果將它們裝入不同的地址，則程序可能會失敗。

--change-section-address *sectionpattern* **{=，+，-}** *val*
--adjust-section-vma *sectionpattern* **{=，+，-}** *val*
設置或更改任何匹配部分的 VMA 地址和 LMA 地址
節模式。如果使用**=**，則段地址設置為 *val*。否則，*val* 為
添加到節地址或從節地址中減去。請參閱下面的評論
--change-addresses，上面。如果 *sectionpattern* 與輸入中的任何部分都不匹配
文件，將發出警告，除非使用**--no-change-warnings**。

--change-section-lma *sectionpattern* **{=，+，-}** *val*
設置或更改與 *sectionpattern* 匹配的任何節的 LMA 地址。LMA
address 是在程序加載時該段將被加載到內存中的地址
時間。通常，這與 VMA 地址相同，即 VMA 的地址。
部分在程序運行時進行，但在某些系統上，尤其是在其中有程序的系統
保存在 ROM 中，兩者可以不同。如果使用**=**，則段地址設置為
val。否則，將 *val* 添加到節地址或從節地址中減去。見上面**--change-addresses**
下的注釋。如果 *sectionpattern* 不匹配
輸入文件中的節，將發出警告，除非**--no-change-warnings** 為
用過的。

--change-section-vma *sectionpattern* **{=，+，-}** *val*
設置或更改任何與 *sectionpattern* 匹配的節的 VMA 地址。VMA 地址
是程序啟動後該部分將位於的地址
執行。通常，這與 LMA 地址相同，即
該部分將被加載到內存中，但是在某些系統上，尤其是那些
程序保存在 ROM 中，兩者可以不同。如果使用**=**，則段地址
設置為 *val*。否則，將 *val* 添加到節地址或從節地址中減去。
請參閱上方**--change-addresses** 下的注釋。如果 *sectionpattern* 不匹配
輸入文件中的任何部分，都會發出警告，除非**--no-change-warnings**
用來。

--change-warnings
--adjust-warnings
如果使用**--change-section-address** 或**--change-section-lma** 或**--change-section-vma**，
並且節模式與任何節都不匹配，請發出警告。這是
默認。

--no-change-warnings

--no-adjust-warnings

如果使用**--change-section-address** 或**--adjust-section-lma** 或 **--adjust-section-vma**，即使該節位於模式與任何部分都不匹配。

--set-section-flags sectionpattern =標志

設置與 sectionpattern 匹配的任何部分的標志。該標志參數是一個逗號 標記名稱的分隔字符串。可識別的名稱是 **alloc**，內容，**load**， **noload**，**readonly**，代碼，**data**，**rom**，**share** 和 **debug**。您可以設置內容標志

對於沒有內容的部分，但清除該部分沒有意義
具有內容的部分的內容標志-只需刪除該部分即可。
並非所有標志對於所有目標文件格式都有意義。

--add-section sectionname = filename 復制文件時

添加一個名為 sectionname 的新部分。新內容
部分取自文件 filename。該部分的大小將是
文件。此選項僅適用於支持以下格式的文件格式：
任意名稱。注意-可能需要使用**--set-section-flags** 選項
設置新創建的部分的屬性。

--dump-section sectionname = filename

將名為 sectionname 的節的內容放入文件 filename 中，覆蓋
以前可能在那裡的任何內容。此選項是
--add-section。該選項類似於**--only-section** 選項，除了它
不創建格式化文件，它只是將內容轉儲為原始二進制數據，
而不應用任何重定位。可以多次指定該選項。

--update 截面 sectionname =文件名

替換已命名的節中的現有內容 sectionname 與文件的內容
的文件名。該部分的大小將調整為文件的大小。這 sectionname 的
section 標志將保持不變。對於 ELF 格式的文件，請參閱
段映射也將保持不變，這是無法使用的
--remove-section 和**--add-section**。該選項可以指定多個
一次。

注意-可以使用**--rename-section** 和**--update-section** 來更新和
從一個命令行重命名節。在這種情況下，請傳遞原始節名稱
到**--update-section**，原始和新的部分名稱到**--rename-section**。

--add-symbol name = [section ：] value [，flags]在復制文件時

添加一個名為 name 的新符號。可以指定此選項
多次。如果給出了該部分，則該符號將與和
相對於該部分，否則將是 ABS 符號。指定一個未定義
部分將導致致命錯誤。沒有檢查該值，它將是
按指定採取。可以指定符號標志，並非所有標志都可以
對於所有目標文件格式都有意義。默認情況下，該符號為全局符號。這
特殊標志'before = othersym '將在指定的 elseym 之前插入新符號
，否則將在符號表的末尾添加符號。
他們出現的順序。

--rename-section oldname = newname [，flags]

將一個節從 oldname 重命名為 newname，可以選擇在此過程中將該節的標志更改為
flags。與使用鏈接程序腳本執行以下操作相比，這具有優勢
重命名，因為輸出將保留為目標文件，並且不會成為鏈接文件
可執行文件。

當輸入格式為二進制時，此選項特別有用，因為它將
始終創建一個名為.data 的節。例如，如果您想創建一個
包含二進制數據的名為.rodata 的部分，您可以使用以下命令行
實現它：

```
objcopy -I 二進制-O <輸出格式> -B <體系結構> \
 --rename-section .data = .rodata，alloc，load，readonly，data，contents \
 <輸入二進制文件> <輸出對象文件>
```

--long-section-names {啟用，禁用，保留}

處理“ COFF”和“ PE-COFF”時控制長節名稱的處理
對象格式。默認行為 **keep**，是保留長節名稱（如果有）
在輸入文件中。在**啟用**和**禁用**選項強制啟用或
禁止在輸出對象中使用長節名稱；當**禁用**生效時，
輸入對象中的任何長節名稱都將被截斷。在**啟用**選項
如果輸入中存在長節名，則僅發出長節名；這大致相同
與 **keep** 一樣，但是**使能**選項是否可能強制創建仍未定義
輸出文件中的空字符串表。

--change-lead-char

某些目標文件格式在符號開頭使用特殊字符。最多
下劃線通常是此類字符，編譯器通常在每個符號前添加下劃線。

此選項告訴 **objcopy** 更改每個符號的前導字符
在目標文件格式之間轉換。如果目標文件格式使用相同的前導
字符，此選項無效。否則，它將添加一個字符，或刪除一個
字符，或酌情更改字符。

-刪除領先的字符

如果全局符號的第一個字符是使用的特殊符號前導字符
通過目標文件格式，刪除字符。最常見的符號領先
字符是下劃線。此選項將從全球范圍內刪除領先的下劃線
符號。如果要將不同文件的對象鏈接在一起，這將很有用。
符號名稱使用不同約定的格式。這不同於
--change-leading-char，因為它總是在適當的時候更改符號名稱，
無論輸出文件的目標文件格式如何。

--reverse-bytes = num

反轉具有輸出內容的節中的字節。截面長度必須均勻
被給定值整除以便能夠進行交換。倒車
發生在執行交織之前。

此選項通常用於為有問題的目標系統生成 ROM 映像。
例如，在某些目標板上，將從 8 位 ROM 讀取的 32 位字重新
不論 CPU 字節順序如何，都以低位字節順序匯編。根據
在編程模型中，可能需要修改 ROM 的字節序。

考慮一個簡單的文件，該文件的節包含以下八個字節：12345678。
對於上面的示例，

使用**--reverse-bytes = 2**，輸出文件中的字節為
訂購 21436587。

對於上面的示例，

使用**--reverse-bytes = 4**，輸出文件中的字節為
訂購 43218765。

通過使用**--reverse 字節= 2** 的以上示例中，接著**--reverse 字節= 4** 上的
輸出文件，第二個輸出文件中的字節將被排序 34127856。

--srec-len = ival

僅對 srec 輸出有意義。設置 Srecords 的最大長度為
生產到節目。該長度涵蓋地址，數據和 crc 字段。

--srec-forceS3

僅對 srec 輸出有意義。避免生成 S1 / S2 記錄，僅創建 S3
記錄格式。

--redefine-sym old = new
將符號 old 的名稱更改為 new。當人們嘗試鏈接時，這可能會很有用
兩件事在一起，你沒有源頭，並且有名字沖突。

--redefine-SYMS =文件名
應用**--redefine -均**每個符號對**"**舊 新**"**的文件中列出的文件名。
filename 是簡單的平面文件，每行一對符號。行注釋可能是
由井號字符引入。可以多次給此選項。

- 削弱
將文件中的所有全局符號更改為弱。在構建一個
使用鏈接器的**-R** 選項與其他對象鏈接的對象。
僅當使用支持弱文件格式的目標文件格式時，此選項才有效。
符號。

--keep-symbols = filename
將**--keep-symbol** 選項應用於文件 filename 中列出的每個符號。 文件名是
只是一個平面文件，每行一個符號名稱。行注釋可以通過以下方式引入
哈希字符。可以多次給此選項。

--strip-symbols = filename
將**--strip-symbol** 選項應用於文件 filename 中列出的每個符號。 文件名是
只是一個平面文件，每行一個符號名稱。行注釋可以通過以下方式引入
哈希字符。可以多次給此選項。

--strip-unneeded-symbols = filename
將**--strip-unneeded-symbol** 選項應用於文件 filename 中列出的每個符號。
filename 是簡單的平面文件，每行一個符號名稱。行注釋可能是
由井號字符引入。可以多次給此選項。

--keep-global-symbols = filename
將**--keep-global-symbol** 選項應用於文件 filename 中列出的每個符號。
filename 是簡單的平面文件，每行一個符號名稱。行注釋可能是
由井號字符引入。可以多次給此選項。

--localize-symbols = filename

將**--localize-symbol** 選項應用於文件 filename 中列出的每個符號。 文檔名稱

只是一個平面文件，每行一個符號名稱。可能會引入行注釋
通過井號字符。可以多次給此選項。

--globalize-symbols = filename

將**--globalize-symbol** 選項應用於文件 filename 中列出的每個符號。 文檔名稱

只是一個平面文件，每行一個符號名稱。可能會引入行注釋
通過井號字符。可以多次給此選項。

--weaken-symbols = filename

將**--weaken-symbol** 選項應用於文件 filename 中列出的每個符號。 文件名是
只是一個平面文件，每行一個符號名稱。行注釋可以通過以下方式引入
哈希字符。可以多次給此選項。

--alt-machine-code = index

如果輸出體系結構具有備用機器代碼，請改用 index th 代碼
默認之一。如果為機器分配了官方代碼，並且
工具鏈採用了新代碼，但其他應用程序仍依賴於
正在使用原始代碼。對於基於 ELF 的體系結構，如果使用索引替代項
不存在，則將該值視為要存儲在
ELF 標頭的 e_machine 字段。

-可寫文本

將輸出文本標記為可寫。此選項對所有目標文件都沒有意義
格式。

--readonly-text

使輸出文本具有寫保護。此選項對所有對象都沒有意義
文件格式。

- 純的

將輸出文件標記為按需分頁。此選項對所有對象都沒有意義
文件格式。

-不純

將輸出文件標記為不純。此選項對所有目標文件都沒有意義
格式。

--prefix 符號=串

前綴與輸出文件中的所有符號串。

--prefix-sections = string

使用 string 將輸出文件中的所有節名稱前綴。

--prefix-alloc-sections = string

使用 string 將輸出文件中所有已分配節的所有名稱前綴。

--add-gnu-debuglink =文件路徑

創建一個.gnu_debuglink 節，其中包含對文件路徑的引用並添加
它到輸出文件。注意：指向文件路徑的文件必須存在。的一部分
添加.gnu_debuglink 節的過程涉及嵌入以下文件的校驗和：
將調試信息文件的內容放入該部分。

如果調試信息文件建在一個位置，但要安裝在一個位置
稍後進入另一個位置，則不要使用安裝路徑
地點。該**--add-GNU-debuglink** 因為安裝文件執行選項將失敗
還不存在。而是將調試信息文件放在當前目錄中，並使用
--add-gnu-debuglink 選項，不包含任何目錄組件，例如：

```
objcopy --add-gnu-debuglink = foo.debug
```

在調試時，調試器將嘗試在以下位置查找單獨的調試信息文件：
一組已知位置。這些位置的確切集合取決於
使用的發行版，但通常包括：

“ *與可執行文件相同的目錄。”
“ *包含可執行文件的目錄的子目錄”
叫做.debug

“ *全局調試目錄，例如/ usr / lib / debug。

只要之前將調試信息文件安裝到這些位置之一中，
調試器正在運行，一切應該正常工作。

--keep-file-symbols

剝離文件時，可能使用**--strip-debug** 或**--strip-unneeded 時**，保留任何
指定源文件名的符號，否則將被剝離。

--only-keep-debug

剝離文件，刪除不會被剝離的任何節的內容
--strip-debug 並保持調試部分完整無缺。在 ELF 文件中，此保留
輸出中的所有注釋部分。

注意-保留已刪除部分的部分標題，包括它們的標題
大小，但該部分的內容將被丟棄。節標題是
保留，以便其他工具可以將 debuginfo 文件與實際
可執行文件，即使該可執行文件已重定位到其他地址空間。

目的是將此選項與**--add-gnu-debuglink** 結合使用
創建一個兩部分的可執行文件。一個剝離的二進制文件，它將佔用較少的空間
在 RAM 和分發中，第二個是調試信息文件
如果需要調試功能，則需要。建議的程序來創建這些
文件如下：

1. <正常鏈接可執行文件。假設被稱為>
   然後是“ foo”

1. <運行“ objcopy --only-keep-debug foo foo.dbg”到>
   創建一個包含調試信息的文件。

1. <運行“ objcopy --strip-debug foo”創建一個>
   剝離的可執行文件。

1. <運行“ objcopy --add-gnu-debuglink = foo.dbg foo”>
   將調試信息的鏈接添加到剝離的可執行文件中。

注意---選擇“ .dbg”作為調試信息文件的擴展名是任意的。
同樣，“-only-keep-debug”步驟是可選的。您可以改為：

1. <正常鏈接可執行文件。
1. <將“ foo”復制到“ foo.full”
1. <運行“ objcopy --strip-debug foo”>
1. <運行“ objcopy --add-gnu-debuglink = foo.full foo”>

即，-- **add-gnu-debuglink** 指向的文件可以是完整的可執行文件。它
不必是**--only-keep-debug** 開關創建的文件。

注意-此開關僅適用於完全鏈接的文件。它不會

在調試信息可能不完整的目標文件上使用它是有意義的。
此外，gnu_debuglink 功能目前僅支持存在一個文件名
包含調試信息，而不是每個對像一個文件包含多個文件名
基礎。

--strip-dwo
刪除所有 DWARF .dwo 節的內容，保留其餘調試信息
部分和所有符號完整無缺。此選項供編譯器使用**-gsplit-dwarf** 選項的
一部分，該選項在.o 文件之間拆分調試信息
和一個單獨的.dwo 文件。編譯器在同一時間生成所有調試信息
文件，然後使用**--extract-dwo** 選項將.dwo 部分復制到.dwo 文件中，
然後使用**--strip-dwo** 選項從原始.o 文件中刪除這些部分。

--extract-dwo
提取所有 DWARF .dwo 節的內容。有關更多信息，請參見**--strip-dwo** 選項
信息。

--file-alignment num
指定文件對齊方式。文件中的節將始終以文件偏移量開始
是此數字的倍數。默認值為 512。[此選項特定
PE 目標。]

--heap reserve
--heap reserve，提交
指定要保留（和可選地）用作的內存字節數
該程序的堆。[此選項特定於 PE 目標。]

--image-base value
使用 value 作為程序或 dll 的基址。這是最低的內存
加載程序或 dll 時將使用的位置。減少需要
遷移並提高 dll 的性能，每個 dll 都應具有唯一的基址
並且不與任何其他 dll 重疊。可執行文件的默認值為 0x400000，並且
dll 為 0x10000000。[此選項特定於 PE 目標。]

--section-alignment num
設置截面對齊方式。內存中的部分將始終從以下地址開始
是此數字的倍數。預設為 0x1000。[此選項特定於 PE
目標。]

--stack 儲備

--stack 儲備，提交
指定要保留（和可選地）用作的內存字節數
該程序的堆棧。[此選項特定於 PE 目標。]

--subsystem 其中
--subsystem 其中：major
--subsystem 其中：major 。次要的
指定程序將在其下執行的子系統。的法律價值
分別是“本機”，“窗口”，“控制台”，“ posix”，“ efi-app”，“ efi-bsd”，“ efi-rtd”，
“ sal-rtd”和“ xbox”。您也可以選擇設置子系統版本。數字
還接受其中的值。[此選項特定於 PE 目標。]

--extract-符號
保留文件的節標記和符號，但刪除所有節數據。具體來說，
選項：

* <刪除所有部分的內容；>
* <將每個部分的大小設置為零；和>
* <將文件的起始地址設置為零。

此選項用於為 VxWorks 內核構建.sym 文件。也可以是
減小**--just-symbols** 鏈接器輸入文件大小的有用方法。

--compress-debug-sections
使用 zlib 和 ELF ABI 的 SHF_COMPRESSED 壓縮 DWARF 調試節。筆記 -
如果壓縮實際上會使一個部分變大，則它不會被壓縮。

--compress-debug-sections =無
--compress-debug-sections = zlib
--compress-debug-sections = zlib-gnu
--compress-debug-sections = zlib-gabi
對於 ELF 文件，這些選項控制 DWARF 調試節的壓縮方式。
--compress-debug-sections = none 等效於**--decompress-debug-sections**。
--compress-debug-sections = zlib 和**--compress-debug-sections = zlib-gabi** 是等效的
到**--compress 調試截面**。 **--compress-debug-sections = zlib-gnu** 壓縮 DWARF
使用 zlib 調試部分。將調試部分重命名為以**.zdebug**

而不是**.debug** 開頭。注意-如果壓縮實際上會使部分變大，則
它沒有壓縮也沒有重命名。

--decompress-debug-sections
使用 zlib 解壓縮 DWARF 調試部分。的原始部分名稱
壓縮的節將恢復。

--elf-stt-common =是
--elf-stt-common =否
對於 ELF 文件，這些選項控制是否應將通用符號轉換為
“ STT_COMMON”或“ STT_OBJECT”類型。 **--elf-stt-common = yes** 轉換常用符號類型
到“ STT_COMMON”。**--elf-stt-common = no** 將常用符號類型轉換為“ STT_OBJECT”。

-V
--version
顯示 **objcopy** 的版本號。

-v
--verbose
詳細輸出：列出所有已修改的目標文件。對於檔案，使用 **objcopy -V**
列出存檔的所有成員。

--help
顯示 **objcopy** 選項的摘要。

- 信息
顯示一個列表，其中列出了所有可用的體系結構和對象格式。

@ file
從 file 中讀取命令行選項。讀取的選項將插入
原始@文件選項。如果文件不存在或無法讀取，則該選項
將按字面意義處理，並且不會刪除。
文件中的
選項用空格分隔。可能包含空格字符
在選項中將整個選項括在單引號或雙引號中。任何
字符（包括反斜槓）可以通過在字符前面加上
包含反斜槓。該文件本身可能包含其他@文件選項；任何
這樣的選項將被遞歸處理。

另請參見

ld（1），objdump（1）和 binutils 的 Info 條目。

版權