

ARM® コンパイラ

バージョン 5.05

armar ユーザガイド

ARM®

ARM® コンパイラ

armar ユーザガイド

Copyright © 2010-2014 ARM. All rights reserved.

リリース情報

ドキュメント履歴

発行	日付	機密保持ステータス	変更点
A	28 5 月 2010	非機密扱い	ARM コンパイラ v4.1 リリース
B	30 9 月 2010	非機密扱い	ARM コンパイラ v4.1 のアップデート 1
C	28 1 月 2011	非機密扱い	ARM コンパイラ v4.1 パッチ 3 のアップデート 2
D	30 4 月 2011	非機密扱い	ARM コンパイラ v5.0 リリース
E	29 7 月 2011	非機密扱い	ARM コンパイラ v5.0 のアップデート 1
F	30 9 月 2011	非機密扱い	ARM コンパイラ v5.01 リリース
G	29 2 月 2012	非機密扱い	ARM コンパイラ v5.01 リリースマニュアルの更新 1
H	27 7 月 2012	非機密扱い	ARM コンパイラ v5.02 リリース
I	31 1 月 2013	非機密扱い	ARM コンパイラ v5.03 リリース
J	16 12 月 2013	非機密扱い	ARM コンパイラ v5.04 リリース
K	10 9 月 2014	非機密扱い	ARM コンパイラ v5.05 リリース

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of ARM. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, ARM makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to ARM’s customers is not intended to create or refer to any partnership relationship with any other company. ARM may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any signed written agreement covering this document with ARM, then the signed written agreement prevails over and supersedes the conflicting provisions of these terms.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited or its affiliates in the EU and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow ARM's trademark usage guidelines at <http://www.arm.com/about/trademark-usage-guidelines.php>

Copyright © [2010-2014], ARM Limited or its affiliates. All rights reserved.

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

非機密著作権情報

本書は、著作権などの権利により保護されており、本書に含まれる手順または実装に関する情報は1つ以上の特許または申請中の特許により保護されている可能性があります。本書のいかなる部分も、ARMから事前に書面による明示的な承諾なく、何らかの形式や方法で無断複製することは許可されていません。特に記載がない限り、明示的であるか黙示的であるかを問わず、また禁反言やその他のいかなる知的財産権のライセンスを許諾するものではありません。

本書の情報には、実装により、いかなる第三者の特許も侵害されないことを確認する目的で情報を使用せず、第三者にもそれを許可しないと承諾することを条件としてアクセスすることができます。

本書は、「現状」のまま提供されます。ARMは、明示的、黙示的、または制定法上のいずれを問わず、いかなる表明も保証も行いません。これには、本書に関連した商品性、品質基準、非侵害、または特定目的への適合性に関する黙示的保証を含むが、これに限定されません。疑義を避けるため、ARMは第三者の特許、著作権、営業機密、または他の権利の範囲および内容に関して、いかなる表明も行わず、識別や理解のための分析も行いません。

本書には、技術的に不正確な箇所および誤記が含まれる場合があります。

法により禁止されていない限りにおいて、ARMは本書の使用により生じた直接的、間接的、特別、付随的、懲罰的、または結果的損害などを含むすべての損害に対して、たとえそのような損害の可能性が事前に告知されていた場合でも、その原因および責任理論の如何に関わらず一切の責任を負わないものとします。

本書には、商品のみが含まれています。本書の使用、複製、または開示が関連するあらゆる輸出法および輸出規制に完全に準拠し、本書が全体であれ一部であれ、該当する輸出法に違反して直接的または間接的に輸出されることがないことを保証する責任を負うものとします。ARMのお客様に関連して「パートナー」という言葉が使用されている場合でも、他会社と提携関係を設立することや、言及することを意図するものではありません。ARMは、通知することなくいつでも本書を変更することができます。

本契約のいずれかの規定と、ARMと締結された本書の内容を含む署名済みの書面契約の間に矛盾がある場合、署名済みの書面契約を本契約の規定より優先するものとします。本書は、便宜上、他言語に翻訳される場合がありますが、本書の英語版と翻訳との間に矛盾がある場合、契約書の英語版に含まれる規定を優先することに同意するものとします。

記号 (® または ™) が付いた言葉およびロゴは、ARM Limited や関連会社の EU またはその他の国における登録商標および商標です。All rights reserved. 本書に記載されている他の製品名は、各社の所有する商標です。ARMの商標の使用に関する次のガイドラインに従ってください。 <http://www.arm.com/about/trademark-usage-guidelines.php>

Copyright © [2010-2014], ARM Limited or its affiliates. All rights reserved.

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

機密保持ステータス

本書は非機密扱いであり、本書を使用、複製、および開示する権利は、ARM および ARM が本書を提供した当事者との間で締結した契約の条項に基づいたライセンスの制限により異なります。

無制限アクセスは、ARM 社内による分類です。

製品ステータス

本書の情報は最終版であり、開発済み製品に対応しています。

Web アドレス

<http://www.jp.arm.com>

目次

ARM® コンパイラ armar ユーザガイド

	序章	
	本書について	8
第 1 章	ARM ライブラリアントピックの概要	
1.1	ARM ライブラリアンについて	1-11
1.2	ライブラリファイル使用時の注意事項	1-12
1.3	armar のコマンドラインのシンタックス	1-13
1.4	armar コマンドのヘルプの取得オプション	1-14
第 2 章	armar コマンドラインオプション	
2.1	archive	2-17
2.2	-a pos_name	2-18
2.3	-b pos_name	2-19
2.4	-c	2-20
2.5	-C	2-21
2.6	--create	2-22
2.7	-d	2-23
2.8	--debug_symbols	2-24
2.9	--diag_error=tag[,tag,...]	2-25
2.10	--diag_remark=tag[,tag,...]	2-26
2.11	--diag_style={arm ide gnu}	2-27
2.12	--diag_suppress=tag[,tag,...]	2-28
2.13	--diag_warning=tag[,tag,...]	2-29

2.14	--entries	2-30
2.15	file_list	2-31
2.16	--help	2-32
2.17	-i pos_name	2-33
2.18	-m pos_name	2-34
2.19	-n	2-35
2.20	--new_files_only	2-36
2.21	-P	2-37
2.22	-r	2-38
2.23	-s	2-39
2.24	--show_cmdline	2-40
2.25	--sizes	2-41
2.26	-t	2-42
2.27	-T	2-43
2.28	-u	2-44
2.29	-v	2-45
2.30	--version_number	2-46
2.31	--via=filename	2-47
2.32	--vsn	2-48
2.33	-x	2-49
2.34	--zs	2-50
2.35	--zt	2-51

第 章3

via ファイルの構文

3.1	via ファイルの概要	3-53
3.2	via ファイルの構文規則	3-54

付録 A

armar ドキュメント改訂

A.1	armar ライブラリマネージャユーザガイドの改訂	付録-A-57
-----	---------------------------------	---------

表の一覧

ARM® コンパイラ armar ユーザガイド

表 A-1	発行 F と発行 J の相違点	付録-A-57
表 A-2	発行 C と発行 F の相違点	付録-A-57
表 A-3	発行 C と発行 D の相違点	付録-A-57

序章

この前書きでは、次について紹介します。ARM® コンパイラ *armar* ユーザガイド。
このドキュメントは、次で構成されています。

- [本書について\(8 ページ\)](#)。

本書について

『ARM コンパイラ `armar` ユーザガイド』で、`armar` ユーティリティを使用する方法を説明します。

本書の構成

本書は以下の章から構成されています。

第 1 章 ARM ライブラリアントピックの概要

ARM® コンパイラに付属する ARM ライブラリアン、`armar` の概要について説明します。

第 2 章 `armar` コマンドラインオプション

ARM ライブラリアン `armar` のコマンドラインオプションについて説明します。

第 3 章 `via` ファイルの構文

`armar` でサポートされている `via` ファイルの構文について説明します。

付録 A `armar` ドキュメント改訂

『`armar` ライブラリマネージャユーザガイド』に対して加えられた技術的変更について説明します。

用語集

「ARM 用語集」は、ARM マニュアルで使用されている用語とその定義のリストです。一般に認められている意味と ARM での意味が異なる場合を除いて、「ARM 用語集」に業界標準の用語は含まれていません。

詳細については、「[ARM 用語集](#)」を参照して下さい。

表記規則

italic

重要用語、相互参照、引用箇所を示します。

bold

メニュー名などのユーザインタフェース要素を太字で記載しています。また、必要に応じて記述リスト内の重要箇所、ARM プロセッサの信号名、重要用語、および専門用語にも太字を使用しています。

monospace

コマンド、ファイル名、プログラム名、ソースコードなど、キーボードから入力可能なテキストを示しています。

monospace

コマンドまたはオプションに使用可能な略語を示しています。コマンド名またはオプション名をすべて入力する代わりに、下線部分の文字だけを入力することができます。

monospace italic

引数が特定の値で置き換えられる場合のモノスペーステキストの引数を示しています。

monospace bold

サンプルコード以外に使用される言語キーワードを示しています。

<and>

コードまたはコードの一部のアセンブラ構文で置換可能な項が使用されている場合に、その項を囲みます。例えば、

```
MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>
```

スモールキャピタル

「ARM 用語集」で定義されている専門的な意味を持つ用語について、本文中で使用されません。例えば、IMPLEMENTATION DEFINED、IMPLEMENTATION SPECIFIC、UNKNOWN、UNPREDICTABLE などです。

ご意見、ご感想

本製品に関するフィードバック

本製品についてのご意見やご提案がございましたら、以下の情報を添えて購入元までお寄せ下さい。

- 製品名
- 製品のリビジョンまたはバージョン
- 説明にはできるだけ多くの情報を含めて下さい。適宜、症状と診断手順も含めて下さい。

内容に関するフィードバック

内容に関するご意見につきましては、電子メールを errata@arm.com まで送信して下さい。その際には、以下の内容を記載して下さい。

- タイトル
- 文書番号 (ARM DUI0476KJ)。
- 問題のあるページ番号
- 問題点の簡潔な説明

また、補足すべき点や改善すべき点についての全般的なご提案もお待ちしております。

—— 注 ——

ARM では、この PDF を Adobe Acrobat および Acrobat Reader でのみテストしており、その他の PDF リーダーを使用した場合の表示品質については、保証いたしかねます。

その他の情報

- [ARM 情報センター](#)。
- [ARM Technical Support Knowledge Articles](#)
- [サポートおよびメンテナンス](#)。
- [ARM 用語集](#)

第 1 章

ARM ライブラリアントピックの概要

ARM® コンパイラ に付属する ARM ライブラリアン、`armar` の概要について説明します。

以下のセクションから構成されています。

- [1.1 ARM ライブラリアンについて\(1-11 ページ\)](#).
- [1.2 ライブラリアンファイル使用時の注意事項\(1-12 ページ\)](#).
- [1.3 `armar` のコマンドラインのシンタックス\(1-13 ページ\)](#).
- [1.4 `armar` コマンドのヘルプの取得オプション\(1-14 ページ\)](#).

1.1 ARM ライブラリアンについて

ARM ライブラリアン `armar` を使用すると、ELF 形式のオブジェクトファイル群を標準形式の `ar` ライブラリにまとめて保管できます。

これにより、複数の ELF オブジェクトファイルの代わりにこれらのライブラリをリンカに渡すことができます。

`armar` によって以下の操作を実行できます。

- 新しいライブラリを作成する。
- ライブラリにファイルを追加する。
- ライブラリの個別ファイルを置き換える。
- 単一操作によりライブラリのすべてのファイルを指定のファイルと置き換える。
- ライブラリのファイルの配置を制御する。
- 指定したライブラリに関する情報を表示する。たとえば、ライブラリのすべてのメンバをリストできます。

ライブラリに追加されたファイルや置換されたファイルには、タイムスタンプも関連付けられます。

—— 注 ——

ライブラリでオブジェクトファイルの作成、追加、または置換を行う場合は、`armar` によりデフォルトでシンボルテーブルが作成されます。ただし、デバッグシンボルはデフォルトでは含まれていません。

関連参照

[2.8 `--debug_symbols` \(2-24 ページ\)](#)。

関連情報

`--library=name` リンカオプション。
`--libpath=pathlist` リンカオプション。
`--library_type=lib` リンカオプション。
`--userlibpath=pathlist` リンカオプション。

1.2 ライブラリファイル使用時の注意事項

ライブラリファイルを使用する場合は、いくつかの点に注意する必要があります。

以下の点に注意して下さい。

- ライブラリが共有オブジェクトやダイナミックリンクライブラリ(DLL)と異なる点を以下に示します。
 - シンボルが共有オブジェクトまたは DLL からインポートされる。
 - シンボルのコードまたはデータがアーカイブからリンクしているファイルに抽出される。
- オブジェクトライブラリファイルとリンクした場合は、オブジェクトライブラリファイル内に保管されているすべてのオブジェクトファイルとリンクした場合と同じ結果が得られない場合があります。これは、リンカが入力リストとライブラリを、以下のような異なる方法で処理するためです。
 - 入力リスト内の各オブジェクトファイルは無条件で出力に表示されますが、`armlink --remove` オプションが指定されている場合、未使用の領域が削除されます。
 - ライブラリファイルのメンバは、オブジェクトファイルまたは以前に処理されたライブラリファイルによって参照されている場合にのみ、出力にインクルードされます。

リンカは、`ar` 形式のファイルとして格納された ELF ファイル群をライブラリとして認識します。各 ELF ファイルの内容がライブラリの 1 つのメンバを形成します。

関連情報

[`--remove`](#)、[`--no_remove`](#) リンカオプション。

1.3 armar のコマンドラインのシンタックス

armar コマンドには、ファイルおよびライブラリの処理方法を指定するオプションがあります。

構文

```
armar options archive [file_list]
```

options

armar コマンドラインオプション。

archive

ライブラリのファイル名。ライブラリファイルは常に指定する必要があります。

file_list

処理するファイルのリストを指定します。

関連参照

[2.1 archive\(2-17 ページ\)](#)。

[2.15 file_list\(2-31 ページ\)](#)。

1.4 armar コマンドのヘルプの取得オプション

主なコマンドラインオプションの一覧を表示するには、`--help` オプションを使用します。
これは、オプションやソースファイルを指定しない場合のデフォルトの動作です。

例

ヘルプ情報を表示するには、以下のように入力します。

```
armar --help
```

第 2 章

armar コマンドラインオプション

ARM ライブラリアン `armar` のコマンドラインオプションについて説明します。

以下のセクションから構成されています。

- [2.1 archive \(2-17 ページ\)](#).
- [2.2 -a pos_name \(2-18 ページ\)](#).
- [2.3 -b pos_name \(2-19 ページ\)](#).
- [2.4 -c \(2-20 ページ\)](#).
- [2.5 -C \(2-21 ページ\)](#).
- [2.6 --create \(2-22 ページ\)](#).
- [2.7 -d \(2-23 ページ\)](#).
- [2.8 --debug_symbols \(2-24 ページ\)](#).
- [2.9 --diag_error=tag\[,tag,...\] \(2-25 ページ\)](#).
- [2.10 --diag_remark=tag\[,tag,...\] \(2-26 ページ\)](#).
- [2.11 --diag_style={arm|ide|gnu} \(2-27 ページ\)](#).
- [2.12 --diag_suppress=tag\[,tag,...\] \(2-28 ページ\)](#).
- [2.13 --diag_warning=tag\[,tag,...\] \(2-29 ページ\)](#).
- [2.14 --entries \(2-30 ページ\)](#).
- [2.15 file_list \(2-31 ページ\)](#).
- [2.16 --help \(2-32 ページ\)](#).
- [2.17 -i pos_name \(2-33 ページ\)](#).
- [2.18 -m pos_name \(2-34 ページ\)](#).
- [2.19 -n \(2-35 ページ\)](#).

- 2.20 *--new_files_only* (2-36 ページ).
- 2.21 *-P* (2-37 ページ).
- 2.22 *-r* (2-38 ページ).
- 2.23 *-s* (2-39 ページ).
- 2.24 *--show_cmdline* (2-40 ページ).
- 2.25 *--sizes* (2-41 ページ).
- 2.26 *-t* (2-42 ページ).
- 2.27 *-T* (2-43 ページ).
- 2.28 *-u* (2-44 ページ).
- 2.29 *-v* (2-45 ページ).
- 2.30 *--version_number* (2-46 ページ).
- 2.31 *--via=filename* (2-47 ページ).
- 2.32 *--vsn* (2-48 ページ).
- 2.33 *-x* (2-49 ページ).
- 2.34 *--zs* (2-50 ページ).
- 2.35 *--zt* (2-51 ページ).

2.1 **archive**

作成、変更、または読み取るライブラリの場所を指定します。

—— 注 ——

file_list にファイルリストを含める場合は、ライブラリファイルの後に指定する必要があります。

関連参照

[2.15 *file_list* \(2-31 ページ\)](#).

2.2 *-a pos_name*

ライブラリ内の新しいファイルを、指定したライブラリメンバの後に配置します。

構文

-a= pos_name

ここで、*pos_name* は、ライブラリ内のファイルの名前です。

使用法

同じコマンドラインで *-b* (または *-i*) を指定した場合、このオプションは無効になります。

例

mylib.a で *obj2.o* の直後に *obj3.o* ファイルと *obj4.o* ファイルを追加するか、置換するには、以下のように入力します。

```
armar -r -a obj2.o mylib.a obj3.o obj4.o
```

関連参照

- [2.3 *-b pos_name* \(2-19 ページ\)](#).
- [2.17 *-i pos_name* \(2-33 ページ\)](#).
- [2.18 *-m pos_name* \(2-34 ページ\)](#).
- [2.22 *-r* \(2-38 ページ\)](#).

2.3 -b *pos_name*

ライブラリ内の新しいファイルを、指定したライブラリメンバの前に配置します。

構文

-b= *pos_name*

ここで、*pos_name* は、ライブラリ内のファイルの名前です。

使用法

同じコマンドラインで -a を指定した場合、このオプションが優先されます。

関連参照

[2.2 -a *pos_name* \(2-18 ページ\)](#).

[2.17 -i *pos_name* \(2-33 ページ\)](#).

[2.18 -m *pos_name* \(2-34 ページ\)](#).

[2.22 -r \(2-38 ページ\)](#).

2.4 -c

通常はライブラリ作成時に `stderr` に出力される診断メッセージを非表示にします。

2.5 -C

情報の抽出が行われる際に、既存のファイルが同じような名前のファイルに置き換えられないようライブラリアンに指示します。

使用法

このオプションを `-T` と組み合わせることによって、短縮されたファイル名が同じ接頭文字の付いたファイルに置き換えられるのを防ぎます。

抽出されるファイルが現在の位置に既に存在する場合は、エラーメッセージが表示されます。

関連参照

[2.27 -T\(2-43 ページ\)](#).

[2.33 -x\(2-49 ページ\)](#).

2.6 *--create*

file_list に指定されたファイルのみを含む 新しいライブラリを作成します。ライブラリが既に存在する場合は、以前の内容が破棄されます。

使用法

--create オプションを使用して、以下のいずれかの方法でオブジェクトファイルのリストを指定します。

- コマンドラインで直接指定します。
- *via* ファイルで指定します。

—— 注 ——

ライブラリが既に存在する場合は、以前の内容が削除されます。

このオプションは、以下の互換性のあるコマンドラインオプションと組み合わせて使用できます。

- *-c*
- *--diag_style*
- *-n*
- *-v*
- *--via*

—— 注 ——

状況によっては、これ以外のオプションを使用して新しいライブラリを作成することもできます。例えば、存在しないライブラリを指定する場合は、*-r* オプションを使用します。

サンプル

現在のディレクトリにすべてのオブジェクトファイルを追加して新しいライブラリを作成するには、以下のように入力します。

```
armar --create mylib.a *.o
```

via ファイルにリストされているファイルを含む新しいライブラリを作成するには、以下のように入力します。

```
armar --create mylib.a --via myobject.via
```

関連参照

[2.15 *file_list*\(2-31 ページ\)](#).

2.7 -d

file_list に指定された 1 つ以上のファイルをライブラリから削除します。
このオプションは、他の互換性のあるコマンドラインオプションと共に使用できます。

例

ファイル `file1.o` および `file2.o` を `mylib.a` ライブラリから削除するには、以下のように入力します。

```
armar -d mylib.a file1.o,file2.o
```

関連参照

[2.15 file_list\(2-31 ページ\)](#)。

2.8 *--debug_symbols*

デフォルトでは、デバッグシンボルはアーカイブに含まれません。代わりにアーカイブにデバッグシンボルを含めるには、*--debug_symbols* を使用して下さい。

関連概念

[1.1 ARM ライブラリアンについて\(1-11 ページ\)](#)。

2.9 `--diag_error=tag[,tag,...]`

特定のタグがある診断メッセージにエラーの重大度を設定します。

構文

`--diag_error= tag[,tag,...]`

`tag` には以下のいずれかを指定できます。

- エラーの重大度を設定する診断メッセージ番号
- `warning` (すべての警告をエラーとして扱う場合)

関連参照

[2.10 `--diag_remark=tag\[,tag,...\]`](#) (2-26 ページ).

[2.11 `--diag_style={arm|ide|gnu}`](#) (2-27 ページ).

[2.12 `--diag_suppress=tag\[,tag,...\]`](#) (2-28 ページ).

[2.13 `--diag_warning=tag\[,tag,...\]`](#) (2-29 ページ).

2.10 *--diag_remark=tag[,tag,...]*

特定のタグがある診断メッセージに注釈の重要度を設定します。

構文

```
--diag_remark= tag[,tag,...]
```

tag は診断メッセージ番号のコンマ区切りのリストです。

関連参照

[2.9 *--diag_error=tag\[,tag,...\]*](#) (2-25 ページ).

[2.11 *--diag_style={arm|ide|gnu}*](#) (2-27 ページ).

[2.12 *--diag_suppress=tag\[,tag,...\]*](#) (2-28 ページ).

[2.13 *--diag_warning=tag\[,tag,...\]*](#) (2-29 ページ).

2.11 `--diag_style={arm|ide|gnu}`

診断メッセージの表示スタイルを指定します。

構文

`--diag_style= string`

string には以下のいずれかを指定できます。

arm

ARM コンパイラの形式を使用してメッセージを表示します。

ide

エラーのある行の行番号と文字数を表示します。これらの値は括弧に囲まれて表示されます。

gnu

gcc で使用される形式でメッセージを表示します。

使用法

`--diag_style=gnu` は、GNU コンパイラが報告する形式 `gcc` と一致します。

`--diag_style=ide` は、Microsoft Visual Studio が報告する形式と一致します。

デフォルト

デフォルトは `--diag_style=arm` です。

関連参照

[2.9 `--diag_error=tag\[,tag,...\]` \(2-25 ページ\)](#).

[2.10 `--diag_remark=tag\[,tag,...\]` \(2-26 ページ\)](#).

[2.12 `--diag_suppress=tag\[,tag,...\]` \(2-28 ページ\)](#).

[2.13 `--diag_warning=tag\[,tag,...\]` \(2-29 ページ\)](#).

2.12 `--diag_suppress=tag[,tag,...]`

特定のタグがある診断メッセージを非表示にします。

構文

`--diag_suppress= tag[,tag,...]`

`tag` には以下のいずれかを指定できます。

- 非表示にする診断メッセージ番号
- `error` (降格できるすべてのエラーを非表示にする場合)
- `warning` (すべての警告を非表示にする場合)

関連参照

[2.9 `--diag_error=tag\[,tag,...\]` \(2-25 ページ\)](#).

[2.10 `--diag_remark=tag\[,tag,...\]` \(2-26 ページ\)](#).

[2.11 `--diag_style={arm|ide|gnu}` \(2-27 ページ\)](#).

[2.13 `--diag_warning=tag\[,tag,...\]` \(2-29 ページ\)](#).

2.13 `--diag_warning=tag[,tag,...]`

特定のタグがある診断メッセージに警告の重大度を設定します。

構文

`--diag_warning= tag[, tag,...]`

`tag` には以下のいずれかを指定できます。

- 警告の重大度を設定する診断メッセージ番号
- `error` (警告に降格できるすべてのエラーを設定する場合)

関連参照

[2.9 `--diag_error=tag\[,tag,...\]`](#) (2-25 ページ).

[2.10 `--diag_remark=tag\[,tag,...\]`](#) (2-26 ページ).

[2.11 `--diag_style={arm|ide|gnu}`](#) (2-27 ページ).

[2.12 `--diag_suppress=tag\[,tag,...\]`](#) (2-28 ページ).

2.14 *--entries*

アセンブラの *ENTRY* ディレクティブを使用して定義されたエントリポイントを持つライブラリのすべてのオブジェクトファイルを一覧表示します。

使用法

一覧の形式は以下のとおりです。

```
エントリ、オフセット num、セクション name/member
```

例

以下の例は、*myasm.a* の各オブジェクトファイルのエントリポイントを一覧表示します。

```
&gt; armar --entries myasm.a ENTRY at offset 0 in section adrlabel of adrlabel.o ENTRY at  
offset 0 in section ARMex of armex.o ENTRY at offset 0 in section Block of blocks.o ENTRY at  
offset 0 in section Jump of jump.o ENTRY at offset 0 in section LDLabel of ldlabel.o ENTRY  
at offset 0 in section Loadcon of loadcon.o ENTRY at offset 0 in section StrCopy of  
strcpy.o ENTRY at offset 0 in section subrout of subrout.o ENTRY at offset 0 in section  
Tblock of tblock.o ENTRY at offset 0 in section ThumbSub of thumbsub.o ENTRY at offset 0 in  
section Word of word.o
```

関連参照

[2.25 *--sizes* \(2-41 ページ\)](#).

[2.35 *--zt* \(2-51 ページ\)](#).

関連情報

[ENTRY](#).

2.15 file_list

ELF オブジェクトや ELF ライブラリなど、ELF に準拠するファイルをスペースで区切って指定します。

使用法

各ファイルは、そのパスと名前によって正確に指定される必要があります。パスには、絶対パス、ドライブおよびルートへの相対パス、現在のディレクトリへの相対パスのいずれかを設定できます。

—— 注 ——

ファイルリストは、ライブラリファイルの後に指定する必要があります。

ライブラリ内のファイルの名前と比較する場合には、パスの終わりにあるファイル名のみが使用されません。複数のパスオペランドが同じファイル名で終わる場合の結果は定義されません。ファイルの指定には、ワイルドカード文字 * および ? を使用できます。

いずれかのファイルがライブラリである場合、**armar** はその入力ライブラリのすべてのメンバをデスティネーションのライブラリにコピーします。コマンドライン上でのメンバの順序はそのまま保持されます。したがって、ライブラリファイルを指定することは、ライブラリに格納されている順序でそのライブラリのすべてのメンバを指定することと論理的には同じです。

2.16 **--help**

主なコマンドラインオプションの一覧を表示します。

デフォルト

これは、オプションやソースファイルなしで *armar* を指定する場合のデフォルトです。

関連参照

[2.30 *--version_number* \(2-46 ページ\)](#) .

[2.32 *--vsn* \(2-48 ページ\)](#) .

2.17 **-i pos_name**

ライブラリ内の新しいファイルを、指定したライブラリメンバの前に配置します。

構文

`-i= pos_name`

ここで、*pos_name* は、ライブラリ内のファイルの名前です。

これは `-b pos_name` と同等です。

関連参照

[2.2 *-a pos_name* \(2-18 ページ\)](#).

[2.3 *-b pos_name* \(2-19 ページ\)](#).

[2.18 *-m pos_name* \(2-34 ページ\)](#).

[2.22 *-r* \(2-38 ページ\)](#).

2.18 **-m pos_name**

ライブラリのファイルを指定場所に移動します。

構文

`-m= pos_name`

ここで、*pos_name* は、ライブラリ内のファイルの名前です。

使用法

pos_name と `-a`、`-b`、または `-i` のいずれかが指定されている場合は、ファイルを新しい位置に移動します。それ以外の場合は、ファイルをライブラリの最後に移動します。

例

ファイル `file1.o` を `mylib.a` ライブラリの `file2.o` 後の新しい場所に移動するには、以下のように入力します。

```
armar -m -a file2.o mylib.a file1.o
```

関連参照

[2.2 `-a pos_name` \(2-18 ページ\)](#) .

[2.3 `-b pos_name` \(2-19 ページ\)](#) .

[2.17 `-i pos_name` \(2-33 ページ\)](#) .

2.19 -n

ライブラリ内のシンボルテーブルの作成を抑制します。

使用法

オブジェクトファイルのライブラリを作成する場合は、**armar** によりデフォルトでシンボルテーブルが常に作成されます。

-s オプションを使用してライブラリにシンボルテーブルを再作成できます。

例

シンボルテーブルを作成しないでライブラリを作成するには、以下のように入力します。

```
armar -n --create mylib.a *.obj
```

関連参照

[2.23 -s \(2-39 ページ\)](#).

2.20 ***--new_files_only***

新しいオブジェクトが新しいタイムスタンプを持つ場合にのみ、アーカイブのオブジェクトファイルを更新します。

使用法

-r オプションと組み合わせて使用すると、ライブラリ内のファイルは、対応するファイルの更新日がライブラリ内のファイルの更新日よりも新しい場合にのみ置き換えられます。

関連参照

[2.22 *-r* \(2-38 ページ\)](#).

[2.28 *-u* \(2-44 ページ\)](#).

2.21 -P

ライブラリ内のファイルの内容を `stdout` に出力します。

例

`mylib.a` の `file1.o` の内容を表示するには、以下のように入力します。

```
armar -p mylib.a file1.o
```

関連概念

[ライブラリ内のファイルの内容を表示するオプション.](#)

2.22 -r

指定したライブラリのファイルを置換するか、指定したライブラリにファイルを追加します。

使用法

ライブラリが存在しない場合は、新しいライブラリファイルが作成され、診断メッセージが標準エラーに出力されます。このオプションは、他の互換性のあるコマンドラインオプションと共に使用できます。

-q は -r のエイリアスです。

指定されているファイルがないのにそのライブラリが存在する場合の結果は定義されていません。既存のファイルに置き換わるファイルによって、ライブラリの順序が変更されることはありません。

-u オプションが指定されている場合は、更新日がライブラリファイルよりも新しいファイルのみが置き換えられます。

-a、-b、-i のいずれかのオプションが使用されている場合には、*pos_name* を使用して、新しいファイルを *pos_name* の後に配置するか(-a)、前に配置するか(-b または -i)を指定する必要があります。指定されていない場合、新しいファイルは末尾に配置されます。

例

ライブラリで *obj1.o*、*obj2.o*、および *obj3.o* ファイルを追加するか、置換するには、以下のように入力します。

```
armar -r mylib.a obj1.o obj2.o obj3.o
```

ライブラリのファイルが指定したファイルより古い場合にのみ、ライブラリ内の *k* で始まる名前のファイルを置換するには、以下のように入力します。

```
armar -ru mylib.a k*.o
```

関連参照

[2.2 -a *pos_name* \(2-18 ページ\)](#).

[2.3 -b *pos_name* \(2-19 ページ\)](#).

[2.17 -i *pos_name* \(2-33 ページ\)](#).

[2.28 -u \(2-44 ページ\)](#).

[2.15 *file_list* \(2-31 ページ\)](#).

2.23 -s

ライブラリにシンボルテーブルを作成します。

使用法

このオプションは、以下の手段で作成されたライブラリに有用です。

- `-n` オプションの使用
- シンボルテーブルを自動的に作成しないアーカイバの使用

—— 注 ——

オブジェクトファイルのライブラリを作成する場合は、`armar` によりデフォルトでシンボルテーブルが常に作成されます。

例

`-n` オプションを使用して作成されたライブラリにシンボルテーブルを作成するには、以下のように入力します。

```
armar -s mylib.a
```

関連参照

[2.19 -n \(2-35 ページ\)](#).

[2.34 --zs \(2-50 ページ\)](#).

2.24 --show_cmdline

ライブラリアン によって使用されたコマンドラインを出力します。

使用法

ライブラリアン によって処理された後のコマンドラインを表示することによって、以下の点を確認できます。

- ビルドシステムによって使用されているコマンドライン
- 指定されたコマンドラインが ライブラリアン によってどのように解釈されているか(コマンドラインオプションの順序など)

コマンドは正規化されて表示されます。また、`via` ファイルの内容は展開されます。

出力結果は標準エラーストリーム(`stderr`)に送られます。

例

To show how `armar` processes the command-line options for the replacement of file `obj1.o` in `mylib.a`, enter:

```
> armar --show_cmdline -r mylib.a obj1.o [armar --show_cmdline -r mylib.a obj1.o]
```

関連参照

[2.31 --via=filename \(2-47 ページ\)](#).

2.25 *--sizes*

ライブラリ内のメンバごとに、Code、RO Data、RW Data、ZI Data、および Debug のサイズが一覧表示されます。

例

以下の例では、*mylib.a* の *app_1.o* および *app_2.o* のサイズを表示します。

```
> armar --sizes mylib.a Code    RO Data  RW data  ZI Data  Debug  Object Name
464      0        0        0        8612   app_1.o 3356    0        0
10244    11848   app_2.o 3820     0        0        10244   20460   TOTAL
```

関連参照

[2.14 *--entries* \(2-30 ページ\)](#).

[2.35 *--zt* \(2-51 ページ\)](#).

2.26 -t

ライブラリの内容を表すテーブルを出力します。

使用法

書き込まれるリストには、*file_list* で指定されたファイルが含まれます。*file_list* が指定されていない場合は、ライブラリ内のすべてのファイルが、保存された順序で含まれます。

サンプル

mylib.a の内容を表すテーブルを表示するには、以下のように入力します。

```
> armar -t mylib.a app_1.o app_2.o
```

詳細モードでライブラリの内容を表すテーブルを一覧表示するには、以下のように入力します。

```
> armar -tv mylib.a rw-rw-rw- 0/ 0 7512 Jun 22 11:19 2009 app_1.o (offset  
736) rw-rw-rw- 0/ 0 1452 May 19 16:25 2009 app_2.o (offset 8308)
```

関連参照

[2.29 -v\(2-45 ページ\)](#).

[2.15 *file_list*\(2-31 ページ\)](#).

2.27 -T

抽出されたファイルのライブラリ名がファイルシステムでサポートされる長さよりも長い場合、ファイル名を短縮します。

使用法

デフォルトでは、名前の長すぎるファイルを抽出すると、エラーが生成されます。診断メッセージが書き込まれ、そのファイルは抽出されません。

短縮後のファイル名が同一になるファイルがライブラリ内に複数ある場合、先に抽出されたファイルが、後に抽出された同名のファイルに上書きされるので注意して下さい。これを防ぐには `-c` オプションを使用します。

関連参照

[2.5 -C\(2-21 ページ\)](#).

[2.33 -x\(2-49 ページ\)](#).

2.28 -u

指定アーカイブ内の古いファイルを更新します。

使用法

-r オプションと組み合わせて使用すると、ライブラリ内のファイルは、対応するファイルの更新日がライブラリ内のファイルの更新日と同じか、それよりも新しい場合にのみ置き換えられます。

関連参照

[2.20 --new_files_only\(2-36 ページ\)](#).

[2.22 -r\(2-38 ページ\)](#).

2.29 -v

詳細な出力を行います。

使用法

出力の内容は、他に使用されているオプションによって異なります。

-d、-r、-x

ライブラリの作成、構成ファイル、保守に関する情報など、ファイルごとに詳細な情報が書き込まれます。

-P

ファイルの内容を `stdout` に書き込む前に、ファイルの名前が標準出力に書き込まれます。

-t

ライブラリ内のファイルに関する詳細な情報が出力されます。

-x

各ファイルが抽出される前にファイル名が出力されます。

関連参照

[2.7 -d\(2-23 ページ\)](#)。

[2.21 -P\(2-37 ページ\)](#)。

[2.22 -r\(2-38 ページ\)](#)。

[2.26 -t\(2-42 ページ\)](#)。

[2.33 -x\(2-49 ページ\)](#)。

2.30 *--version_number*

使用している *armar* のバージョンを表示します。

使用法

ライブラリアンは、*nnnbbb* 形式のバージョン番号を表示します。各項目には以下の意味があります。

- *nnn* はバージョン番号です。
- *bbbb* はビルド番号を示します。

例

バージョン 5.05 ビルド 0019 は *5050019* と表示されます。

関連参照

[2.16 *--help* \(2-32 ページ\)](#).

[2.32 *--vsn* \(2-48 ページ\)](#).

2.31 `--via=filename`

入力ファイル名とライブラリアン オプションの追加リストを *filename* から読み取ります。

構文

```
--via= filename
```

filename は、コマンドラインでインクルードされるオプションを含む `via` ファイルの名前です。

使用法

ライブラリアン コマンドラインでは複数の `--via` オプションを入力できます。オプション、`--via` は、`via` ファイル内に含めることもできます。

関連概念

[3.1 `via` ファイルの概要\(3-53 ページ\)](#).

2.32 --vsn

バージョン情報とライセンス情報が表示されます。

例

出力例:

```
> armar --vsn
製品: ARM Compiler N.nn
コンポーネント: ARM Compiler N.nn
ツール: armar [build_number]
```

関連参照

[2.16 --help](#) (2-32 ページ).

[2.30 --version_number](#) (2-46 ページ).

2.33 -x

`file_list` で指定されたファイルをライブラリから現在のディレクトリに抽出します。

使用法

ライブラリの内容は変更されません。ファイルが指定されていない場合は、ライブラリ内のすべてのファイルが抽出されます。

ライブラリのファイルの名前が、ファイルシステムでサポートされる長さよりも長い場合、エラーが表示され、ファイルは抽出されません。ファイル名が長いファイルを抽出するには、`-T` オプションを使用して長すぎるファイル名を短縮します。

ファイルは現在の位置に抽出されます。

このオプションは、他の互換性のあるコマンドラインオプションと共に使用できます。

例

ファイル `file1.o` および `file2.o` をディレクトリ `C:\temp` の `mylib.a` ライブラリから `C:\temp\obj` へ抽出するには、以下のように入力します。

```
C:cd \temp\obj
```

```
armar -x ..\mylib.a file1.o,file2.o
```

関連参照

[2.27 -T\(2-43 ページ\)](#).

[2.15 file_list\(2-31 ページ\)](#).

2.34 --zs

ライブラリのすべてのファイルのシンボルテーブルを表示します。

例

mylib.a のシンボルテーブルを一覧表示するには、以下のように入力します。

```
> armar --zs mylib.a __ARM_use_no_argv from hello.o at offset 412
main from hello.o at offset 412 __ARM_use_no_argv from test.o
at offset 7960 main from test.o at offset 7960 __ARM_use_no_argv
from hello_ltcg.o at offset 11408 main from hello_ltcg.o at offset 11408
__ARM_use_no_argv from h1.o at offset 18532 main from h1.o
at offset 18532 __ARM_use_no_argv from fncalls.o at offset 2072 add
from fncalls.o at offset 2072 main from fncalls.o at offset 2072
get_stacksize from get_stacksize.o at offset 9672 altstack from
get_stacksize.o at offset 9672 __ARM_use_no_argv from s.o at offset 13068
main from s.o at offset 13068 altstack from s.o
at offset 13068 _Z1fv from t.o at offset 17064 _ZN1T1fEi
from t.o at offset 17064
```

関連参照

[2.19 -n \(2-35 ページ\)](#).

[2.23 -s \(2-39 ページ\)](#).

2.35 --zt

ライブラリ内のすべてのファイルのメンバサイズとエントリポイントを一覧表示します。

例

mylib.a ライブラリ内のすべてのファイルのメンバサイズとエントリポイントを一覧表示するには、以下のように入力します。

```
&gt;armar --zt mylib.a
```

	Code	RO Data	RW Data	ZI Data	Debug	Object	Name	838
0	0	0	0	0	16	0	hello.o	0
0	2869	fncalls.o	0	893	0	0	0	0
test.o	962	0	0	0	0	0	get_stacksize.o	0
838	0	0	0	0	0	0	hello_ltcg.o	8
0	0	80	0	0	56	0	s.o	50
0	0	strcopy.o	4	0	44	0	0	168
relocs-1a.o	36	0	8	0	0	84	t.o	838
0	0	0	0	0	4489	8	94	80
3121	TOTAL							

```
ENTRY at offset 0 in section StrCopy of strcopy.o ENTRY at offset 0 in section StrCopy of emit-relocs-1a.o
```

関連参照

[2.14 --entries \(2-30 ページ\)](#).

[2.25 --sizes \(2-41 ページ\)](#).

第 3 章

via ファイルの構文

armar でサポートされている via ファイルの構文について説明します。

以下のセクションから構成されています。

- [3.1 via ファイルの概要\(3-53 ページ\)](#).
- [3.2 via ファイルの構文規則\(3-54 ページ\)](#).

3.1 *via* ファイルの概要

via ファイルは、ライブラリアンコマンドライン引数とオプションを指定できるプレーンテキストファイルです。

通常、コマンドラインの長さの制限を解決するために *via* ファイルを使用します。ただし、以下のような複数の *via* ファイルを作成します。

- 同じような引数とオプションをグループ化するファイル。
- 異なるシナリオで使用する異なる引数とオプションのセットを含んでいるファイル。

注

一般的には、*via* ファイルを使用して、ツールに対して任意のコマンドラインオプション(`--via`を含む)を指定できます。つまり、ネストされた複数の *via* ファイルを *via* ファイル内から呼び出すことができます。

via ファイルの評価

ライブラリアン が呼び出されると、以下の処理が行われます。

1. 指定されている最初の `--via via_file` 引数を、*via* ファイルから抽出された引数ワードのシーケンスに置き換えます。この中には、再帰処理を行う、*via* ファイル内でネストされた `--via` コマンドも含まれます。
2. それ以降の `--via via_file` 引数についても、出現した順番で同じように処理します。

つまり、*via* ファイルは指定された順番で処理され、ネストされた *via* ファイルを含めて各 *via* ファイルが完全に処理されてから次の *via* ファイルが処理されます。

関連参照

- [3.2 *via* ファイルの構文規則\(3-54 ページ\)](#).
- [2.31 `--via=filename`\(2-47 ページ\)](#).

3.2 via ファイルの構文規則

via ファイルは構文規則に準拠している必要があります。

- via ファイルは、一連のワードで構成されるテキストファイルです。テキストファイル内の各ワードは、引数文字列に変換されてからツールに渡されます。
- 区切られた文字列内にある場合を除き、ワードはホワイトスペースまたは行の終わりで区切られません。以下に例を示します。

```
-d -v (2 ワード)
```

```
-d -v (1 ワード)
```

- 行の終わりはホワイトスペースとして処理されます。以下に例を示します。

```
-d-v
```

これは以下のコードと同等です。

```
-d -v
```

- 二重引用符(")またはアポストロフィ(')で囲まれた文字列は、1 ワードとして処理されます。二重引用符で囲まれたワード内で使用されているアポストロフィは通常の文字として処理されます。アポストロフィで区切られたワード内では、二重引用符は通常の文字として処理されます。

二重引用符を使用して、スペースを含むファイル名またはパス名を 1 つのワードとしてまとめます。以下に例を示します。

```
--via C:\My Project\viafile (3 ワード)
```

```
--via "C:\My Project\viafile" (2 ワード)
```

また、アポストロフィを使用して、二重引用符を含むワードを 1 つのワードとしてまとめます。以下に例を示します。

```
-DNAME='ARM コンパイラ' (1 ワード)
```

- 括弧で囲まれた文字は、1 ワードとして処理されます。以下に例を示します。

```
--option(x, y, z) (1 ワード)
```

```
--option (x, y, z) (2 ワード)
```

- 二重引用符またはアポストロフィで囲まれた文字列内では、バックスラッシュ(\) 文字を使用して、二重引用符、アポストロフィ、およびバックスラッシュ文字をエスケープできます。
- 1 つのワードとしてまとめられたワードのすぐ隣にあるワードは、1 ワードとして処理されます。以下に例を示します。

```
--via"C:\Project\viafile"
```

これは、以下の 1 ワードとして処理されます。

```
--via C:\Project\viafile
```

- 先頭にあるホワイトスペース文字を除いて、セミコロン(;)またはハッシュ(#) 文字で始まる行は、コメント行として解釈されます。行頭以外の場所にあるセミコロンまたはハッシュ文字は、コメントの開始を表す文字としては解釈されません。以下に例を示します。

```
-o objectname.axf ;これはコメントではありません
```

コメントの終わりは、行の終わりまたはファイルの終わりとなります。複数行にわたるコメントはなく、行の一部だけがコメントになることもありません。

関連概念

[3.1 via ファイルの概要\(3-53 ページ\)](#)。

関連参照

[2.31 --via=filename\(2-47 ページ\)](#).

付録 A

armar ドキュメント改訂

『armar ライブラリマネージャユーザガイド』に対して加えられた技術的変更について説明します。
以下のセクションから構成されています。

- [A.1 armar ライブラリマネージャユーザガイドの改訂\(付録-A-57 ページ\)](#).

A.1 armar ライブラリマネージャユーザガイドの改訂

『armar ライブラリマネージャユーザガイド』に対して、以下の技術的変更が加えられました。

表 A-1 発行 F と発行 J の相違点

変更点	関連するトピック
via ファイル構文に関する章を追加しました。	<ul style="list-style-type: none"> • 3 via ファイルの構文(3-52 ページ)
コマンドラインオプションの説明から重複した情報を含む章およびトピックを削除しました。	<ul style="list-style-type: none"> • ライブラリの作成。 • ライブラリの管理。 • ライブラリに関する情報の表示。
構文セクションを <code>-a</code> 、 <code>-b</code> 、 <code>-i</code> 、および <code>-m</code> に追加しました。	<ul style="list-style-type: none"> • 2.2 -a pos_name(2-18 ページ). • 2.3 -b pos_name(2-19 ページ). • 2.17 -i pos_name(2-33 ページ). • 2.18 -m pos_name(2-34 ページ).
トピック <code>--project</code> 、 <code>--reinitialize_workdir</code> 、および <code>--workdir</code> に関するトピックを削除しました。	2 armar コマンドラインオプション(2-15 ページ)
<code>--vsn</code> で報告される出力を変更しました。	2.32 --vsn(2-48 ページ) .

表 A-2 発行 C と発行 F の相違点

変更点	関連するトピック
<code>--version_number</code> および <code>--vsn</code> で報告されたバージョン番号を変更しました。	<ul style="list-style-type: none"> • 2.30 --version_number(2-46 ページ). • 2.32 --vsn(2-48 ページ).

表 A-3 発行 C と発行 D の相違点

変更点	関連するトピック
<code>--project=filename</code> 、 <code>--no_project</code> のトピックタイトルを修正しました。	<ul style="list-style-type: none"> • <code>--project=filename</code>、<code>--no_project</code>
<code>--project</code> 、 <code>--reinitialize_workdir</code> 、および <code>--workdir</code> オプションの説明についてのメモを追加しました。	<ul style="list-style-type: none"> • <code>--reinitialize_workdir</code>。 • <code>--workdir=directory</code>