Red Hat Developer Tools 1

LLVM 14.0.6 Toolset の使用

LLVM 14.0.6 Toolset のインストールおよび使用

Last Updated: 2023-03-05

Red Hat Developer Tools 1 LLVM 14.0.6 Toolset の使用

LLVM 14.0.6 Toolset のインストールおよび使用

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

LLVM Toolset は、Red Hat Enterprise Linux (RHEL) オペレーティングシステム上の開発者向けの Red Hat 製品です。本書では、LLVM Toolset の概要、さまざまなバージョンの LLVM ツールを起動して使用する方法、および詳細な情報を含むリソースを検索する方法を説明します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 LLVM TOOLSET

LLVM Toolset は、Red Hat Enterprise Linux (RHEL) 上の開発者向けの Red Hat 製品です。これは、LLVM コンパイラーインフラストラクチャーフレームワーク、C 言語および C++ 言語用の Clang コンパイラー、LLDB デバッガー、コード解析の関連ツールを提供します。

LLVM Toolset は、Red Hat Developer Tools for Red Hat Enterprise Linux 7 の一部として配布されています。Red Hat Enterprise Linux 8 では、LLVM Toolset はモジュールとして利用できます。LLVM Toolset は、Red Hat Enterprise Linux 9 のパッケージとして利用できます。

1.1. LLVM TOOLSET コンポーネント

以下のコンポーネントは、LLVM Toolset の一部として利用できます。

Name	バージョン	説明
clang	14.0.6	C および C++ の LLVM コンパイラーフロントエンド。
lldb	14.0.6	LLVM の一部を使用した C および C++ デバッガー。
compiler-rt	14.0.6	LLVM および Clang のランタイムライブラリー。
llvm	14.0.6	一連のモジュールおよび再利用可能なコンパイラーおよびツールチェーン技術。
libomp	14.0.6	並列プログラミングに Open MP API 仕様を使用するためのライブラリー。
lld	14.0.6	LLVM リンカー。
python-lit	14.0.6	LLVM および Clang ベースのテストスイート用のソフトウェアテストツール。

注記

CMake ビルドマネージャーは、LLVM Toolset の一部ではありません。Red Hat Enterprise Linux 7 では、CMake が別のパッケージとして提供されます。Red Hat Enterprise Linux 8 では、CMake がシステムリポジトリーから利用できます。Red Hat Enterprise Linux 9 では、CMake がシステムリポジトリーから利用できます。CMake のインストール方法は、Installing CMake on Red Hat Enterprise Linux を参照してください。

1.2. LLVM TOOLSET の互換性

LLVM Toolset は、以下のアーキテクチャーで Red Hat Enterprise Linux 7、Red Hat Enterprise Linux 8、および Red Hat Enterprise Linux 9 で利用できます。

- AMD および Intel 64 ビット
- 64 ビット ARM (RHEL 8 および RHEL 9)
- IBM Power Systems (リトルエンディアン)
- IBM Power Systems、ビックエンディアン (RHEL 7 のみ)
- 64 ビット IBM Z

1.3. RED HAT ENTERPRISE LINUX 7 で LLVM TOOLSET へのアクセス

LLVM Toolset を Red Hat Enterprise Linux 7 にインストールするには、Red Hat Developer Tools リポジトリーおよび Red Hat Software Collections リポジトリーにアクセスして有効にする必要があります。
これらのリポジトリーがすでにシステムに割り当てられている場合は、LLVM Toolset のインストールを参照してください。

手順

1. 以下を実行して **Wget** をインストールします。

```
# yum install wget
```

2. 以下を実行して最新のサブスクリプションデータをダウンロードします。

```
# subscription-manager refresh
```

3. 以下を実行してシステムを登録します。

```
# subscription-manager register
```

   グラフィカルユーザーインターフェイス (GUI) を使用してシステムを登録するには、システムの登録および登録解除 ガイドを参照してください。

4. 利用可能なサブスクリプションの一覧を表示し、以下のコマンドを実行してプール ID を特定します。

```
# subscription-manager list --available
```

5. **Pool ID** で始まる行でプール ID を検索します。
6. 以下を実行して、**Red Hat Developer Tools** リポジトリーへのアクセスを提供するサブスクリプションをシステムに割り当てます。

```
# subscription-manager attach --pool=<pool ID from the subscription>
```

   - **<pool ID from the subscription>** を、直前の手順で特定したプール ID に置き換えます。

7. 以下を実行して、システムに割り当てられているサブスクリプションを確認します。

```
# sudo subscription-manager list --consumed
```

8. 以下を実行して **rhel-7-variant-devtools-rpms** リポジトリーを有効にします。

```
# subscription-manager repos --enable rhel-7-<variant>-devtools-rpms
```

- **<variant>** を、Red Hat Enterprise Linux システムのバリアント (**server** または **workstation**) に置き換えます。
  server を使用して、最も幅広い開発ツールにアクセスします。

9. 以下を実行して **rhel-variant-rhscl-7-rpms** リポジトリーを有効にします。

```
# subscription-manager repos --enable rhel-<variant>-rhscl-7-rpms
```

- **<variant>** を、Red Hat Enterprise Linux システムのバリアント (**server** または **workstation**) に置き換えます。

10. 以下を実行して Red Hat Developer Tools GPG キーをシステムに追加します。

```
# cd /etc/pki/rpm-gpg
# wget -O RPM-GPG-KEY-redhat-devel https://www.redhat.com/security/data/a5787476.txt
# rpm --import RPM-GPG-KEY-redhat-devel
```

関連情報

- システムを登録し、サブスクリプションに関連付ける方法は、Red Hat Subscription Management ガイドコレクションを参照してください。

1.4. LLVM TOOLSET のインストール

すべての開発ツールおよびデバッグツールと依存パッケージを含む LLVM Toolset をインストールするには、以下の手順を実行します。

前提条件

- Red Hat Enterprise Linux 7 では、Red Hat Developer Tools コンテンツセットへのアクセスを提供するサブスクリプションがシステムに割り当てられます。
  サブスクリプションを割り当てるには、Red Hat Enterprise Linux 7 の LLVM Toolset へのアクセス を参照してください。

- 利用可能な Red Hat Enterprise Linux のすべての更新がインストールされている。

手順

Red Hat Enterprise Linux 7 に、**llvm-toolset-14.0** コレクションをインストールします。

```
# yum install llvm-toolset-14.0
```

Red Hat Enterprise Linux 8 で、以下のコマンドを実行して **llvm-toolset** モジュールをインストールします。

```
# yum module install llvm-toolset
```

Red Hat Enterprise Linux 9 で、以下のコマンドを実行して **llvm-toolset** パッケージをインストールします。

```
# dnf install llvm-toolset
```

1.5. CMAKE ビルドマネージャーのインストール

CMake ビルドマネージャーは、ソースコードのビルドプロセスをコンパイラーとは別に管理するツールです。CMake は、ソースコードのコンパイル、ライブラリーの作成、ラッパーの生成、実行可能なファイルの構築を行うネイティブビルド環境を生成できます。
CMake ビルドマネージャーをインストールするには、以下の手順を実行します。

前提条件

- LLVM Toolset がインストールされている。
  詳細は、Installing LLVM Toolset を参照してください。

手順

CMake をインストールするには、以下のコマンドを実行します。

- Red Hat Enterprise Linux 7 の場合:

```
# yum install llvm-toolset-{llvm-ts-cmake-ver-rhel7}-cmake
```

- Red Hat Enterprise Linux 8 の場合:

```
# yum install cmake
```

- Red Hat Enterprise Linux 9 の場合

```
# dnf install cmake
```

関連情報

- CMake ビルドマネージャーの詳細は、公式の CMake ドキュメント About CMake を参照してください。
- CMake ビルドマネージャーの使用の概要は、以下を参照してください。
  - CMake リファレンスドキュメントの 紹介
  - 公式の CMake ドキュメント CMake Tutorial。

1.6. LLVM TOOLSET ドキュメントのインストール

LLVM Toolset のドキュメントは、ローカルシステムにインストールできます。

前提条件

- LLVM Toolset がインストールされている。
  詳細は、Installing LLVM Toolset を参照してください。

手順

llvm-doc パッケージをインストールするには、以下のコマンドを実行します。

- Red Hat Enterprise Linux 7 の場合:

```
# yum install llvm-toolset-13.0-llvm-doc
```

このドキュメントは、**/opt/rh/llvm-toolset-14.0/root/usr/share/doc/llvm-toolset-14.0-llvm-14.0/html/index.html** パスにあります。

- Red Hat Enterprise Linux 8 の場合:

```
# yum install llvm-doc
```

ドキュメントは、**/usr/share/doc/llvm/html/index.html** の下にあります。

- Red Hat Enterprise Linux 9 の場合

```
# dnf install llvm-doc
```

ドキュメントは、**/usr/share/doc/llvm/html/index.html** の下にあります。

1.7. CMAKE ドキュメントのインストール

CMake ビルドマネージャーのドキュメントをローカルシステムにインストールできます。

前提条件

- Cmake がインストールされている。
詳細は、Installing the CMake build manager を参照してください。

手順

cmake-doc パッケージをインストールするには、以下のコマンドを実行します。

- Red Hat Enterprise Linux 7 の場合:

```
# yum install llvm-toolset-14.0-cmake-doc
```

このドキュメントは **opt/rh/llvm-toolset-14.0/root/usr/share/doc/llvm-toolset-14.0-cmake-3.6.2/html/cmake.html** パスにあります。

- Red Hat Enterprise Linux 8 の場合:

```
# yum install cmake-doc
```

ドキュメントは、**/usr/share/doc/llvm/html/cmake.html** の下にあります。

- Red Hat Enterprise Linux 9 の場合

```
# dnf install cmake-doc
```

ドキュメントは、**/usr/share/doc/llvm/html/cmake.html** の下にあります。

1.8. 関連情報

- LLVM Toolset の詳細は、公式の LLVM ドキュメント を参照してください。

第2章 CLANG コンパイラー

Clang は、C ベースの言語 C、C ++、Objective C/C++、OpenCL、および Cuda 用の LLVM コンパイラーフロントエンドです。

LLVM Toolset には、Clang 14.0.6 が同梱されています。

2.1. 前提条件

- LLVM Toolset がインストールされている。
  詳細は、Installing LLVM Toolset を参照してください。

2.2. ソースファイルのコンパイル

Clang コンパイラーを使用して、アセンブリー言語ソースファイルとソースファイルをコンパイルします。Clang は、コンパイルの結果として実行可能なバイナリーファイルを作成します。コードをデバッグできるようにするには、**-g** フラグを Clang コマンドに追加してデバッグ情報を有効にします。

注記

C++ プログラムをコンパイルするには、**clang** の代わりに **clang++** を使用します。

手順

プログラムをコンパイルするには、以下のコマンドを実行します。

- Red Hat Enterprise Linux 7 の場合:

```
$ scl enable llvm-toolset-14.0 'clang -o -g <binary_file> <source_file>'
```

  - **<binary_file>** を出力ファイルの必要な名前に、**<source_file>** をソースファイルの名前に置き換えます。

- Red Hat Enterprise Linux 8 の場合:

```
$ clang -o -g <binary_file> <source_file>
```

  - **<binary_file>** を出力ファイルの必要な名前に、**<source_file>** をソースファイルの名前に置き換えます。

- Red Hat Enterprise Linux 9 の場合

```
$ clang -o -g <binary_file> <source_file>
```

  - **<binary_file>** を出力ファイルの必要な名前に、**<source_file>** をソースファイルの名前に置き換えます。

2.3. プログラムの実行

Clang コンパイラーは、コンパイルの結果として、実行可能なバイナリーファイルを作成します。このファイルを実行し、プログラムを実行するには、以下の手順を実行します。

前提条件

- プログラムがコンパイルされている。
  プログラムをコンパイルする方法は、Compiling a source file を参照してください。

手順

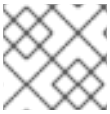
プログラムを実行するには、実行ファイルを含むディレクトリーで実行します。

```
$ ./<binary_file>
```

- **<binary_file>** は、実行可能ファイルの名前に置き換えます。

2.4. オブジェクトファイルの連結

オブジェクトファイルを連結して、プロジェクト全体ではなく、変更を含むソースファイルのみをコンパイルできます。
複数のソースファイルで設定されるプロジェクトで作業する場合は、Clang コンパイラーを使用して各ソースファイルのオブジェクトファイルをコンパイルします。次のステップとして、これらのオブジェクトファイルを連結します。Clang は、リンクされたオブジェクトファイルを含む実行ファイルを自動的に生成します。コンパイル後に、オブジェクトファイルを再度リンクします。



注記

C++ プログラムをコンパイルするには、**clang** の代わりに **clang++** を使用します。

手順

1. ソースファイルをオブジェクトファイルにコンパイルするには、以下のコマンドを実行します。
   - Red Hat Enterprise Linux 7 の場合:

     ```
     $ scl enable llvm-toolset-14.0 'clang -o <object_file> -c <source_file>'
     ```

     - **<object_file>** をオブジェクトファイルの名前に置き換え、**<source_file>** をソースファイルの名前に置き換えます。
   - Red Hat Enterprise Linux 8 の場合:

     ```
     $ clang -o <object_file> -c <source_file>
     ```

     - **<object_file>** をオブジェクトファイルの名前に置き換え、**<source_file>** をソースファイルの名前に置き換えます。
   - Red Hat Enterprise Linux 9 の場合

     ```
     $ clang -o <object_file> -c <source_file>
     ```

     - **<object_file>** をオブジェクトファイルの名前に置き換え、**<source_file>** をソースファイルの名前に置き換えます。
2. オブジェクトファイルをリンクするには、以下のコマンドを実行します。
   - Red Hat Enterprise Linux 7 の場合:

```
$ scl enable llvm-toolset-14.0 'clang -o <output_file> <object_file_0> <object_file_1>'
```

- **<output_file>** を出力ファイルの必要な名前に置き換え、**<object_file>** をリンクするオブジェクトファイルの名前に置き換えます。

- Red Hat Enterprise Linux 8 の場合:

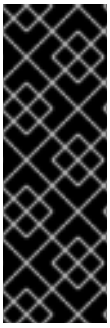
```
$ clang -o <output_file> <object_file_0> <object_file_1>
```

- **<output_file>** を出力ファイルの必要な名前に置き換え、**<object_file>** をリンクするオブジェクトファイルの名前に置き換えます。

- Red Hat Enterprise Linux 9 の場合

```
$ clang -o <output_file> <object_file_0> <object_file_1>
```

- **<output_file>** を出力ファイルの必要な名前に置き換え、**<object_file>** をリンクするオブジェクトファイルの名前に置き換えます。



重要

現時点では、Red Hat Enterprise Linux の複数のバージョンでの実行に対応するために、特定のライブラリー機能は LLVM Toolset で構築されたアプリケーションに静的にリンクされています。これにより、セキュリティーリスクが小さくなります。このリスクによりアプリケーションを再構築する必要がある場合、Red Hat はセキュリティーエラータを発行します。

Red Hat は、アプリケーション全体を静的にリンクしないことを推奨します。

2.5. 関連情報

- Clang コンパイラーの詳細は、公式の Clang コンパイラーのドキュメント を参照してください。
- LLVM Toolset に含まれる man ページを表示するには、以下を実行します。

注記

C++ プログラムをコンパイルするには、**clang** の代わりに **clang++** を使用します。

- Red Hat Enterprise Linux 7 の場合:

```
$ scl enable llvm-toolset-14.0 'man clang'
```

- Red Hat Enterprise Linux 8 の場合:

```
$ man clang
```

- Red Hat Enterprise Linux 9 の場合

```
$ man clang
```

第3章 LLDB デバッガー

LLDB デバッガーは、C および C++ プログラムのデバッグを行うためのコマンドラインツールです。LLDB を使用してデバッグするコード内でメモリーを検査し、コードの実行状態を制御し、コードの特定セクションの実行を検出します。

LLVM Toolset には、LLDB 14.0.6 が同梱されています。

3.1. 前提条件

- LLVM Toolset がインストールされている。
  詳細は、Installing LLVM Toolset を参照してください。
- コンパイラーがデバッグ情報を作成するように設定されている。
  Clang コンパイラーの設定方法は、Clang Compiler User's Manual の Controlling Debug Information を参照してください。
  GCC コンパイラーの設定方法は、Red Hat Developer Toolset User Guide の Preparing a Program for Debugging を参照してください。

3.2. デバッグセッションの開始

LLDB を使用して対話式のデバッグセッションを開始します。

手順

- デバッグするプログラムで LLDB を実行するには、以下のコマンドを使用します。
  - Red Hat Enterprise Linux 7 の場合:

    ```
    $ scl enable llvm-toolset-14.0 'lldb <binary_file>'
    ```

    - **<binary_file>** をコンパイルしたプログラムの名前に置き換えます。
      対話モードで LLDB デバッグセッションを開始している。コマンドラインの端末に、デフォルトのプロンプト **(lldb)** が表示されるようになりました。
  - Red Hat Enterprise Linux 8 の場合:

    ```
    $ lldb <binary_file_name>
    ```

    - **<binary_file>** をコンパイルしたプログラムの名前に置き換えます。
      対話モードで LLDB デバッグセッションを開始している。コマンドラインの端末に、デフォルトのプロンプト **(lldb)** が表示されるようになりました。
  - Red Hat Enterprise Linux 9 の場合

    ```
    $ lldb <binary_file>
    ```

    - **<binary_file>** をコンパイルしたプログラムの名前に置き換えます。
      対話モードで LLDB デバッグセッションを開始している。コマンドラインの端末に、デフォルトのプロンプト **(lldb)** が表示されるようになりました。
- デバッグセッションを終了してシェルプロンプトに戻るには、以下のコマンドを実行します。

```
(lldb) quit
```

3.3. デバッグセッション中のプログラムの実行

LLDB を使用して、デバッグセッション中にプログラムを実行します。最初のブレークポイントに達するか、エラーが発生した場合、またはプログラムが終了したときにプログラムの実行が停止します。

前提条件

- インタラクティブなデバッグセッションを開始している。
  詳細は、Starting a debugging session with LLDB を参照してください。

手順

- デバッグしているプログラムを実行するには、次のコマンドを実行します。

```
(lldb) run
```

- 特定の引数を使用してデバッグするプログラムを実行するには、以下を実行します。

```
(lldb) run <argument>
```

  - **<argument>** を、使用するコマンドライン引数に置き換えます。

3.4. ブレークポイントの使用

ブレークポイントを使用して、ソースコードの設定地点でプログラムの実行を一時停止します。

前提条件

- インタラクティブなデバッグセッションを開始している。
  詳細は、Starting a debugging session with LLDB を参照してください。

手順

- 特定の行に新しいブレークポイントを設定するには、以下のコマンドを実行します。

```
(lldb) breakpoint set --file <source_file_name> --line <line_number>
```

  - **<source_file_name>** をソースファイルの名前に置き換え、**<line_number>** を、ブレークポイントを設定する行番号に置き換えます。

- 特定の関数にブレークポイントを設定するには、以下のコマンドを実行します。

```
(lldb) breakpoint set --name <function_name>
```

  - **<function_name>** を、ブレークポイントを設定する関数の名前に置き換えます。

- 現在設定されているブレークポイントの一覧を表示するには、以下のコマンドを実行します。

```
(lldb) breakpoint list
```

- ブレークポイントを削除するには、次のコマンドを実行します。

```
(lldb) breakpoint clear -f <source_file_name> -l <line_number>
```

- **<source_file_name>** をソースファイルの名前に置き換え、**<line_number>** は削除するブレークポイントの行番号に置き換えます。

- ブレークポイントに達した後にプログラムの実行を再開するには、以下のコマンドを実行します。

```
(lldb) continue
```

- 特定の数のブレークポイントを省略するには、以下のコマンドを実行します。

```
(lldb) continue -i <breakpoints_to_skip>
```

- **<breakpoints_to_skip>** を、スキップするブレークポイントの数に置き換えます。

注記

ループを省略するには、**<breakpoints_to_skip>** をループの反復数に一致するように設定します。

3.5. コードを使用したステッピング

LLDB を使用して、プログラムのコードを使用して、行ポインターの後に 1 行のコードのみを実行できます。

前提条件

- インタラクティブなデバッグセッションを開始している。
  詳細は、Starting a debugging session with LLDB を参照してください。

手順

- コードの 1 行で実施するには、以下を実行します。

  1. 行ポインターを、実行する行に設定します。
  2. 次のコマンドを実行します。

```
(lldb) step
```

- コードの特定の数行を調べるには、以下を行います。

  1. 行ポインターを、実行する行に設定します。
  2. 次のコマンドを実行します。

```
(lldb) step -c <number>
```

  - **<number>** を、実行する行数に置き換えます。

3.6. ソースコードの一覧表示

デバッグしているプログラムを実行する前に、LLDB デバッガーにソースコードの最初の 10 行が自動的に表示されます。プログラムの実行が停止するたびに、LLDB は停止するソースコードの行と、周りの行を表示します。LLDB を使用すると、デバッグセッションでソースコードの表示を手動でトリガーできます。

前提条件

- インタラクティブなデバッグセッションを開始している。
  詳細は、Starting a debugging session with LLDB を参照してください。

手順

- デバッグしているプログラムのソースコードの最初の 10 行を一覧表示するには、次のコマンドを実行します。

```
(lldb) list
```

- 特定の行からソースコードを表示するには、以下を実行します。

```
(lldb) list <source_file_name>:<line_number>
```

  - **<source_file_name>** をソースファイルの名前に置き換え、**<line_number>** は表示する行数に置き換えます。

3.7. 現在のプログラムデータの表示

LLDB デバッガーは、複雑さ、有効な式、および関数呼び出しの戻り値の変数にデータを提供します。LLDB を使用して、プログラムの状態に関連するデータを表示できます。

前提条件

- インタラクティブなデバッグセッションを開始している。
  詳細は、Starting a debugging session with LLDB を参照してください。

手順

特定の変数、式、または戻り値の現在の値を表示するには、以下を実行します。

```
(lldb) print <data_name>
```

- **<data_name>** を、表示するデータに置き換えます。

3.8. 関連情報

- LLDB デバッガーの詳細は、公式の LLDB ドキュメント LLDB Tutorial を参照してください。
- GDB コマンドとその LLDB と同等のものの一覧は、GDB to LLDB Command Map を参照してください。

第4章 RHEL 8 で LLVM TOOLSET を使用したコンテナーイメージ

RHEL 8 では、Containerfiles を使用して、独自の LLVM Toolset コンテナーイメージを Red Hat Universal Base Images (UBI) コンテナー上にビルドできます。

4.1. RHEL 8 での LLVM TOOLSET のコンテナーイメージの作成

RHEL 8 では、LLVM Toolset パッケージは、Red Hat Universal Base Images (UBI) リポジトリーの一部です。コンテナーイメージのサイズを小さくするには、LLVM Toolset 全体ではなく、個々のパッケージのみをインストールします。

前提条件

- 既存の Containerfile。
  Containerfiles 作成の詳細は、Dockerfile reference ページを参照してください。

手順

- Red Hat Container Catalog にアクセスします。
- UBI を選択します。
- **Get this image** をクリックして、指示に従います。
- LLVM Toolset を含むコンテナーイメージを作成するには、以下の行を Containerfile に追加します。

```
FROM registry.access.redhat.com/ubi8/ubi:latest

RUN yum module install -y llvm-toolset
```

- 個々のパッケージのみを含むコンテナーイメージを作成するには、以下の行を Containerfile に追加します。

```
RUN yum install -y <package-name>
```

  - **<package-name>** をインストールするパッケージの名前に置き換えます。

4.2. 関連情報

- Red Hat UBI イメージの詳細は、Working with Container Images を参照してください。
- Red Hat UBI リポジトリーの詳細は、Universal Base Images (UBI): Images, repositories, packages, and source code を参照してください。

第5章 LLVM TOOLSET の変更点

LLVM Toolset が、RHEL 7、RHEL 8、RHEL 9 のバージョン 13.0.1 から 14.0.6 に更新されました。

以下は、主な変更点です。

- 64 ビット x86 では、**AVX512-FP16** 命令のサポートが追加されました。
- Armv9-A、Armv9.1-A、および Armv9.2-A アーキテクチャーのサポートが追加されました。
- PowerPC では、IBM double-double 形式を表す **__ibm128** タイプが **__attribute__mode (IF)** として追加されました。

clang の変更:

- **C++2b** の **if consteval** が実装されました。
- 64 ビット x86 では、**AVX512-FP16** 命令のサポートが追加されました。
- 実験段階の OpenCL C 3.0 および OpenCL 2021 の **C++** サポートが完了しました。
- **-E -P** プリプロセッサーの出力は、必ず空白行を省略します。これは、GCC の動作に一致します。以前は、最大 8 つの連続する空白行が出力に表示される可能性がありました。
- C89 だけでなく、**C99** 以降の標準で **-Wdeclaration-after-statement** をサポートします。これは、GCC の動作に一致します。注目すべきユースケースは、宣言とコードの混在を禁止するスタイルガイドのサポートですが、さらに新しい C 標準に移行したいと考えています。

更新の詳細については、アップストリームの LLVM 14.0.0 リリースノート を参照してください。