Red Hat Developer Tools 1

使用 LLVM 17.0.6 Toolset

安装和使用 LLVM 17.0.6 Toolset

Last Updated: 2024-05-24

Red Hat Developer Tools 1 使用 LLVM 17.0.6 Toolset

安装和使用 LLVM 17.0.6 Toolset

Jacob Valdez
jvaldez@redhat.com

法律通告

摘要

LLVM Toolset 是 Red Hat Enterprise Linux (RHEL)操作系统的开发人员提供的红帽产品。使用本指南获取 LLVM Toolset 的概述，了解如何调用和使用不同版本的 LLVM 工具，并通过更深入的信息查找资源。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 在顶部导航栏中点 **Create**
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 LLVM TOOLSET

LLVM Toolset 是 Red Hat Enterprise Linux (RHEL)的开发人员提供的红帽产品。它提供了 LLVM 编译器基础架构框架、Clang 编译器和 C++ 语言的 Clang 编译器、LLDB 调试器以及用于代码分析的相关工具。

对于 Red Hat Enterprise Linux 8，LLVM Toolset 作为模块提供。LLVM Toolset 作为 Red Hat Enterprise Linux 9 的软件包提供。

1.1. LLVM TOOLSET 组件

以下组件作为 LLVM Toolset 的一部分提供：

Name	版本	描述
clang	17.0.6	C 和 C++ 的 LLVM 编译器前端。
lldb	17.0.6	使用 LLVM 部分的 C 和 C++ 调试器。
compiler-rt	17.0.6	LLVM 和 Clang 的运行时库。
llvm	17.0.6	模块和可重复使用的编译器和链技术的集合。
libomp	17.0.6	用于并行编程使用 Open MP API 规格的库。
lld	17.0.6	LLVM 链接器。
python-lit	17.0.6	用于 LLVM 和 Clang 测试套件的软件测试工具。

注意

CMake 构建管理器不是 LLVM Toolset 的一部分。在 Red Hat Enterprise Linux 8 中，CMake 包括在系统存储库中。在 Red Hat Enterprise Linux 9 中，CMake 包括在系统存储库中。有关如何安装 CMake 的更多信息，请参阅在 Red Hat Enterprise Linux 上安装 CMake。

1.2. LLVM TOOLSET 兼容性

LLVM Toolset 在以下构架上可用于 Red Hat Enterprise Linux 8 和 Red Hat Enterprise Linux 9：

- AMD 和 Intel 64 位
- 64-bit ARM
- IBM Power Systems, Little Endian
- 64-bit IBM Z

1.3. 安装 LLVM TOOLSET

完成以下步骤以安装 LLVM Toolset，包括所有开发和调试工具以及依赖软件包。

先决条件

- 已安装所有可用的 Red Hat Enterprise Linux 更新。

流程

在 Red Hat Enterprise Linux 8 中，运行以下命令安装 **LLVM-toolset** 模块：

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

重要

这不会在 Red Hat Enterprise Linux 8 上安装 LLDB 调试器或 **python3-lit** 软件包。要安装 LLDB 调试器和 **python3-lit** 软件包，请运行：

```
# yum install lldb python3-lit
```

在 Red Hat Enterprise Linux 9 上，运行以下命令安装 **LLVM-toolset** 软件包：

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

重要

这不会在 Red Hat Enterprise Linux 9 上安装 LLDB 调试器或 **python3-lit** 软件包。要安装 LLDB 调试器和 **python3-lit** 软件包，请运行：

```
# dnf install lldb python3-lit
```

1.4. 安装 CMAKE 构建管理器

CMake 构建管理器是一个独立于您的编译器管理源代码的构建过程的工具。Cmake 可以生成原生构建环境，以编译源代码、创建库、生成打包程序和构建可执行文件。
完成以下步骤以安装 CMake 构建管理器。

先决条件

- 已安装 LLVM Toolset。
  如需更多信息，请参阅安装 LLVM Toolset。

流程

要安装 CMake，请运行以下命令：

- 在 Red Hat Enterprise Linux 8 中：

```
# yum install cmake
```

- 在 Red Hat Enterprise Linux 9 中：

```
# dnf install cmake
```

其他资源

- 有关 CMake 构建管理器的更多信息，请参阅官方 CMake 文档概述 关于 CMake。
- 有关使用 CMake 构建管理器简介，请参阅：
  - CMake 参考文档 简介.
  - 官方 CMake 文档 CMake 教程.

1.5. 安装 LLVM TOOLSET 文档

您可以在本地系统中安装 LLVM Toolset 的文档。

先决条件

- 已安装 LLVM Toolset。
  如需更多信息，请参阅安装 LLVM Toolset。

流程

要安装 **LLVM-doc** 软件包，请运行以下命令：

- 在 Red Hat Enterprise Linux 8 中：

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

  您可以在以下路径中找到文档：**/usr/share/doc/llvm/html/index.html**。

- 在 Red Hat Enterprise Linux 9 中：

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

  您可以在以下路径中找到文档：**/usr/share/doc/llvm/html/index.html**。

1.6. 安装 CMAKE 文档

您可以在本地系统中安装 CMake 构建管理器的文档。

先决条件

- 已安装 cmake。
  如需更多信息，请参阅安装 CMake 构建管理器。

流程

要安装 **cmake-doc** 软件包，请运行以下命令：

- 在 Red Hat Enterprise Linux 8 中：

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

您可以在以下路径中找到文档： **/usr/share/doc/cmake/html/index.html**。

- 在 Red Hat Enterprise Linux 9 中：

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

您可以在以下路径中找到文档： **/usr/share/doc/cmake/html/index.html**。

1.7. 其他资源

- 有关 LLVM Toolset 的更多信息，请参阅官方 LLVM 文档。

第 2 章 CLANG 编译器

clang 是基于 C 的语言 C、C++、目标 C/C++、OpenCL 和 Cuda 的 LLVM 编译器前端。

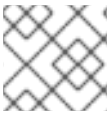
LLVM Toolset 与 Clang 17.0.6 一起发布。

2.1. 先决条件

- 已安装 LLVM Toolset。
  如需更多信息，请参阅安装 LLVM Toolset。

2.2. 编译源文件

使用 Clang 编译器编译源文件和装配语言源文件。clang 创建一个可执行二进制文件，作为编译的结果。要能够调试您的代码，请在 Clang 命令中添加 **-g** 标志来启用调试信息。



注意

要编译 C++ 程序，请使用 **clang++** 而不是 **clang**。

流程

要编译程序，请运行以下命令：

- 在 Red Hat Enterprise Linux 8 中：

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

  - 将 ***<binary_file>*** 替换为输出文件所需的名称，将 ***<source_file>*** 替换为源文件的名称。

- 在 Red Hat Enterprise Linux 9 中：

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

  - 将 ***<binary_file>*** 替换为输出文件所需的名称，将 ***<source_file>*** 替换为源文件的名称。

2.3. 运行一个程序

Clang 编译器会创建一个可执行二进制文件，作为编译的结果。完成以下步骤以执行此文件并运行您的程序。

先决条件

- 您的程序已编译。
  有关如何编译程序的更多信息，请参阅 编译源文件。

流程

要运行您的程序，请在包含可执行文件的目录中运行：

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

- 将 ***<binary_file>*** 替换为可执行文件的名称。

2.4. 将目标文件链接到

通过将目标文件链接到一起，您只能编译包含更改的源文件，而不是整个项目。
当您处理由多个源文件组成的项目时，请使用 Clang 编译器为每个源文件编译对象文件。下一步，将这些目标文件链接在一起。clang 会自动生成包含链接目标文件的可执行文件。编译后，将目标文件再次链接到一起。

注意

要编译 C++ 程序，请使用 **clang++** 而不是 **clang**。

流程

1. 要将源文件编译到目标文件中，请运行以下命令：
   - 在 Red Hat Enterprise Linux 8 中：

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

     - 将 ***<object_file>*** 替换为对象文件所需的名称，将 ***<source_file>*** 替换为源文件的名称。
   - 在 Red Hat Enterprise Linux 9 中：

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

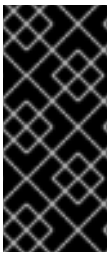
     - 将 ***<object_file>*** 替换为对象文件所需的名称，将 ***<source_file>*** 替换为源文件的名称。
2. 要将目标文件链接在一起，请运行以下命令：
   - 在 Red Hat Enterprise Linux 8 中：

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

     - 将 ***<output_file>*** 替换为输出文件所需的名称，将 &lt ***;object_file>*** 替换为您要链接的对象文件的名称。
   - 在 Red Hat Enterprise Linux 9 中：

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

     - 将 ***<output_file>*** 替换为输出文件所需的名称，将 &lt ***;object_file>*** 替换为您要链接的对象文件的名称。



重要

目前，某些库功能会静态链接到使用 LLVM Toolset 构建的应用程序，以支持它们在多个 Red Hat Enterprise Linux 版本上执行。这会造成一个小的安全风险。如果您需要因为这个风险而需要重建应用程序，红帽会发布一个安全勘误。

红帽建议不要静态地链接整个应用程序。

2.5. 其他资源

- 有关 Clang 编译器的更多信息，请参阅官方 Clang 编译器文档。
- 要显示 LLVM Toolset 中包含的手册页，请运行：

注意

要编译 C++ 程序，请使用 **clang++** 而不是 **clang**。

- 在 Red Hat Enterprise Linux 8 中：

```
$ man clang
```

- 在 Red Hat Enterprise Linux 9 中：

```
$ man clang
```

第 3 章 LLDB 调试器

LLDB 调试器是用于调试 C 和 C++ 程序的命令行工具。使用 LLDB 检查被调试的代码内的内存，控制代码的执行状态，并检测代码的特定部分的执行。

LLVM Toolset 与 LLDB 17.0.6 一起发布。

3.1. 先决条件

- 已安装 LLVM Toolset。
  如需更多信息，请参阅安装 LLVM Toolset。
- 您的编译器被配置为创建调试信息。
  有关配置 Clang 编译器的说明，请参阅 Clang Compiler User's Manual 中的 控制 调试信息。
  有关配置 GCC 编译器的说明，请参阅 Red Hat Developer Toolset 用户指南中的 准备用于调试的程序。

3.2. 启动调试会话

使用 LLDB 启动交互式调试会话。

流程

- 要在您要调试的程序中运行 LLDB，请使用以下命令：
  - 在 Red Hat Enterprise Linux 8 中：

    ```
    $ lldb <binary_file_name>
    ```

    - 将 ***<binary_file>*** 替换为编译的程序的名称。
      您已在交互模式中启动了 LLDB 调试会话。您的命令行终端现在显示默认提示 **(lldb)**。
  - 在 Red Hat Enterprise Linux 9 中：

    ```
    $ lldb <binary_file>
    ```

    - 将 ***<binary_file>*** 替换为编译的程序的名称。
      您已在交互模式中启动了 LLDB 调试会话。您的命令行终端现在显示默认提示 **(lldb)**。
- 要退出调试会话并返回到 shell 提示符，请运行以下命令：

```
(lldb) quit
```

3.3. 在调试会话过程中执行您的程序

使用 LLDB 在调试会话期间执行您的程序。当达到第一个断点、发生错误或程序终止时，程序的执行将停止。

先决条件

- 您已启动交互式调试会话。
  如需更多信息，请参阅 使用 LLDB 启动调试会话。

流程

- 要执行您要调试的程序，请运行：

```
(lldb) run
```

- 要使用特定参数执行您要调试的程序，请运行：

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

  - 使用您要使用的命令行参数替换 < parameter >。

3.4. 使用断点

使用断点来暂停程序在源代码中的集合点执行。

先决条件

- *您已启动交互式调试会话。*
如需更多信息，请参阅 使用 LLDB 启动调试会话。

流程

- *要在特定行中设置一个新的断点，请运行以下命令：*

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

  - *将 <source_file_name > 替换为源文件的名称，< line_number > 替换为您要在其中设置断点的行号。*

- *要在特定功能上设置断点，请运行以下命令：*

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

  - *将 <function_name > 替换为您要设置断点的功能名称。*

- *要显示当前设置的断点列表，请运行以下命令：*

```
(lldb) breakpoint list
```

- 要删除断点，请运行：

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

  - 将 <source_file_name > 替换为源文件的名称，将 &lt ;line_number > 替换为您要删除的断点的行号。

- 要在达到断点后恢复程序的执行，请运行：

```
(lldb) continue
```

- 要跳过特定数量的断点，请运行以下命令：

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

  - 将 <breakpoints_to_skip > 替换为您要跳过的断点数。

注意

要跳过一个循环，设置 &lt ;breakpoints_to_skip& gt; 以匹配循环迭代计数。

3.5. 逐步浏览代码

您可以使用 LLDB 逐步浏览程序代码，以在行指针后仅执行一行代码。

先决条件

- 您已启动交互式调试会话。
  如需更多信息，请参阅 使用 LLDB 启动调试会话。

流程

- 逐步浏览一行代码：

1. 将行指针设置为您要执行的行。

2. 运行以下命令：

```
(lldb) step
```

- 逐步浏览特定数量的代码行：

1. 将行指针设置为您要执行的行。

2. 运行以下命令：

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

   - 将 <number> 替换为您要执行的行数。

3.6. 列出源代码

在执行您要调试的程序前，LLDB 调试器会自动显示源代码的前 10 行。每次程序执行停止时，LLDB 会显示源代码所在的行，其停止及其周围的行。您可以使用 LLDB 在调试会话期间手动触发源代码的显示。

先决条件

- 您已启动交互式调试会话。
  如需更多信息，请参阅 使用 LLDB 启动调试会话。

流程

- 要列出您要调试的程序的源代码的前 10 行，请运行：

```
(lldb) list
```

- 要从特定行显示源代码，请运行：

-

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

- 将 <source_file_name > 替换为源文件的名称， & lt;line_number > 替换为您要显示的行数。

3.7. 显示当前程序数据

LLDB 调试器提供有关任何复杂度、任何有效表达式和函数调用返回值的变量的数据。您可以使用 LLDB 显示与程序状态相关的数据。

先决条件

- 您已启动交互式调试会话。
  如需更多信息，请参阅 使用 LLDB 启动调试会话。

流程

要显示特定变量、表达式或返回值的当前值，请运行：

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

- 将 <data_name > 替换为您要显示的数据。

3.8. 其他资源

- 有关 LLDB 调试器的更多信息，请参阅官方 LLDB 文档 LLDB Tutorial。
- 有关 GDB 命令及其 LLDB 等效命令的列表，请参阅 GDB 到 LLDB 命令映射。

第 4 章 RHEL 8 上带有 LLVM TOOLSET 的容器镜像

在 RHEL 8 中，您可以使用 Containerfile 在 Red Hat Universal Base Images (UBI)容器之上构建自己的 LLVM Toolset 容器镜像。

4.1. 在 RHEL 8 中创建 LLVM TOOLSET 的容器镜像

在 RHEL 8 中，LLVM Toolset 软件包是 Red Hat Universal Base Images (UBI)存储库的一部分。要将容器镜像大小保持小，请只安装单个软件包，而不是整个 LLVM Toolset。

先决条件

- 现有的 Containerfile。
  有关创建 Containerfiles 的详情，请查看 Dockerfile 参考 页面。

流程

- 访问 Red Hat Container Catalog。
- 选择 UBI。
- 点 Get this image 并按照说明进行操作。
- 要创建包含 LLVM Toolset 的容器镜像，请在 Containerfile 中添加以下行：

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

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

- 要创建仅包含单个软件包的容器镜像，请在 Containerfile 中添加以下行：

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

  - 将 < package-name > 替换为您要安装的软件包的名称。

4.2. 其他资源

- 如需有关 Red Hat UBI 镜像的更多信息，请参阅使用容器镜像。
- 如需有关 Red Hat UBI 存储库的更多信息，请参阅 通用基础镜像(UBI)：镜像、存储库、软件包和源代码。

第 5 章 LLVM TOOLSET 中的更改

LLVM Toolset 已从 RHEL 8 和 RHEL 9 上的版本 16.0.1 更新至 17.0.6。主要变更包括：

- 现在完成了不透明指针迁移。
- 在中间优化中删除了对传统传递管理器的支持。

clang 更改：

- C++20 coroutines 不再被视为实验性。
- 改进了 std::move 函数的代码生成，类似于未优化构建。

有关更新的详细信息，请参阅 LLVM 和 Clang 上游发行注记。