构建工具：C

在本练习中，您将练习从源代码构建 C 项目的传统方法。我们将使用 sqlite 数据库作为示例项目来构建。

使用或类似命令将源文件https://sqlite.org/2021/sqlite-autoconf-3340100.tar.gz下载到您的虚拟机中 `wget` ，并使用 解压 `tar -zxvf FILENAME` 。这将创建一个子文件夹， `cd` 在其中执行 a 操作。

您可以看到一个名为的文件 `INSTALL` ，您可以在文本编辑器中打开该文件以查找标准说明：

> 简而言之，shell 命令 `./configure; make; make install` 应该配置、构建和安装这个包。

配置

如果您查看配置脚本的第一行，就会发现 `#! /bin/sh` 该路径在几乎任何与 posix 兼容的系统上都应该有效。整篇文章只有 16000 多行，所以你不会想读完它。

使用 运行脚本 `./configure` 。您可以看到它做了很多检查，包括：

- 您的系统是否有 C 编译器。
- 你的C编译器是否是gcc。
- 你的 C 编译器是否真的可以工作。
- 标准标头是否喜欢 `string.h` 或 `stdlib.h` 存在。
- 您的系统上是否安装了 readline 库。

您的配置脚本应该运行并 `creating Makefile` 在其最后一行之一上打印。

配置脚本基本上是针对 autoconf 开发人员已知的可能破坏构建的任何系统上发现的每个错误和奇怪现象的测试集合。例如，有人曾经报告过 Sun OS 4（1988 年发布）上的构建中的错误，因此在配置脚本的第 2422 行及后续内容中，我们读到

```
# Use test -z because SunOS4 sh mishandles braces in ${var-val}.

# It thinks the first close brace ends the variable substitution.

test -z "$INSTALL_PROGRAM" && INSTALL_PROGRAM='${INSTALL}'
```

制作

输入 `make` 构建 sqlite。如果未安装， `sudo apt install make` 将修复该问题。

某些编译器命令可能需要一段时间才能运行。当它们运行时，请注意所涉及的配置变量的数量（所有通过 传递的内容 `-D` ）；其中一些打开/关闭功能（例如，如果在系统上找不到相应的头文件，则 readline 支持将关闭），其中一些设置特定于操作系统和编译器的选项，例如定义 `-DHAVE_STRING_H` 是否 `string.h` 存在于你的系统。

这些转换为 `#ifdef` 源文件中的命令，例如， `shell.c` 如果头文件存在，则从第 121 行开始我们包含 readline：

```
#if HAVE_READLINE
# include <readline/readline.h>
# include <readline/history.h>
#endif
```

makefile 运行的最后一个命令是

```
gcc [lots of options] -g -O2 -o sqlite3 sqlite3-shell.o sqlite3-sqlite3.o -
lreadline -lcurses
```

这应该构建一个 `sqlite3` 可以运行的可执行文件（用于 `.q` 再次退出）。

如果需要，您现在可以键入 `sudo make install` 将可执行文件复制到 `/usr/local/bin`.

Advanced note

如果它说找不到文件 `.h` ，或者无法将其链接到库文件 (a `.so`)，您该怎么办？C 早于具有包管理器的现代语言，因此这可能意味着您尚未安装代码所依赖的库。幸运的是 `apt-file` ，这里确实很有帮助：运行 `apt-file search <name of file>` 以找出哪个包提供了您缺少的文件并安装它。

我试图构建一个抱怨找不到库的包 `libffi.so` ： 哪个包可能提供了它?

如果您正在构建的软件无法干净地构建，请不要惊慌！阅读错误消息并修复错误。通常安装一个库，或者改变源代码中的路径就足以修复它。能够自己修复简单的错误是 Linux（和其他操作系统）真正强大的原因！