SWI-Prolog 中文文档

目录

快速入门

1.1 在 Unix 上开始 SWI-Prolog

默认情况下，SWI-Prolog 安装为“swipl”。SWI-Prolog 本身及其实用程序的命令行参数使用标准 Unix 手册页记录。SWI-Prolog 通常作为交互式应用程序运行，只需启动程序即可：

```
$ swipl
Welcome to SWI-Prolog ...
...

1 ?-
```

启动 Prolog 后，通常使用 consult/1 将程序加载到其中，可以通过将程序文件的名称放在方括号中来缩写。以下目标加载文件 likes.pl，其中包含谓词 likes/2 的子句：

```
?- [likes].
true.

?-
```

或者，源文件也可以作为命令行参数给出：

```
$ swipl likes.pl
Welcome to SWI-Prolog ...
...

1 ?-
```

以上两个假设 likes.pl 位于您的工作目录中。如果您使用命令行版本 swipl，则工作目录与您启动 SWI-Prolog 的 shell 相同。如果您启动的是 GUI 版本 (swipl-win)，则这在很大程度上取决于操作系统。您可以使用 pwd/0 和 cd/0 来查找和更改工作目录。实用程序 ls/0 列出了工作目录的内容。

```
?- pwd.
% /home/janw/src/swipl-devel/linux/
true.
?- cd('~/tmp').
true.

?- pwd.
% /home/janw/tmp/
true.
```

文件 likes.pl 也安装在 SWI-Prolog 安装目录内的子目录 demo 中，可以使用以下命令加载，而不管工作目录如何。有关 SWI-Prolog 如何指定文件位置的详细信息，请参阅 absolute_file_name/3 和 file_search_path/2。

```
?- [swi(demo/likes)].
true.
```

此后，Unix 和 Windows 用户统一起来，因此如果您使用 Unix，请继续阅读第 1.2 节。

1.2 在 Windows 上启动 SWI-Prolog

在 Windows 系统上安装 SWI-Prolog 后，用户可以使用以下重要的新功能：

一个名为 swipl 的文件夹（本文档的其余部分称为目录），其中包含系统的可执行文件、库等。此目录之外不安装任何文件。 程序 swipl-win.exe，提供与 Prolog 交互的窗口。程序 swipl.exe 是在控制台窗口中运行的 SWI-Prolog 版本。 文件扩展名 .pl 与程序 swipl-win.exe 相关联。打开 .pl 文件将导致 swipl-win.exe 启动，将目录更改为要打开的文件所在的目录，并加载此文件。

启动第 2.1.1.1 节中提到的 likes.pl 文件的正常方法是简单地在 Windows 资源管理器中双击此文件。

1.2 从控制台添加规则

尽管我们强烈建议将程序放在文件中，也可以选择编辑它并使用 make/0 重新加载它（参见第 2.1.4 节），但可以从终端管理事实和规则。添加一些子句的最方便的方法是咨询伪文件用户。输入以系统文件结束字符结束。

```
?- [user].
|: hello :- format('Hello world~n').
|: ^D
true.

?- hello.
Hello world
true.
```

谓词 assertz/1 和 retract/1 是添加和删除规则和事实的替代方法。

1.3 执行查询

加载程序后，可以向 Prolog 询问有关该程序的查询。下面的查询询问 Prolog 食物‘sam’喜欢什么。如果系统可以证明某个 X 的目标，则系统会响应 X = 。如果用户想要另一个解决方案，可以键入分号 (;) 或空格键 7。如果不想看到更多答案，请使用回车键。如果用户使用回车键或 Prolog 知道没有更多答案，则 Prolog 会用句号 (.) 完成输出。如果 Prolog 找不到（更多）答案，它会写入 false。最后，Prolog 使用错误消息来回答，以表明查询或程序包含错误。

```
?- likes(sam, X).
X = dahl ;
X = tandoori ;
...
X = chips.

?-
```

请注意，Prolog 编写的答案是一个有效的 Prolog 程序，执行时会产生与原始程序相同的答案集。

1.4 检查和修改程序

如果配置正确，谓词 edit/1 会根据参数启动内置或用户配置的编辑器。参数可以是任何可以链接到位置的内容：文件名、谓词名称、模块名称等。如果参数仅解析为一个位置，则编辑器将在此位置启动，否则将向用户提供选择。

如果有图形用户界面，编辑器通常会创建一个新窗口，系统会提示下一个命令。用户可以编辑源文件，保存它并运行 make/0 来更新任何已修改的源文件。如果无法在窗口中打开编辑器，它将在同一控制台中打开，并让编辑器运行 make/0 来重新加载已修改的任何源文件。

```
?- edit(likes).

true.
?- make.
% /home/jan/src/pl-devel/linux/likes compiled 0.00 sec, 0 clauses

?- likes(sam, X).
...
```

程序也可以使用 listing/1 进行反编译，如下所示。listing/1 的参数只是一个谓词名称，一个形式为 Name/Arity 的谓词指示符，例如 ?- listing(mild/1). 或一个头，例如 ?- listing(likes(sam, _)).，列出所有匹配子句。谓词 listing/0，即不带参数的谓词，列出了整个程序。9

```
?- listing(mild).
mild(dahl).
mild(tandoori).
mild(kurma).

true.
```

1.5 停止 Prolog

交互式顶层可以通过两种方式停止：输入系统文件结束符（通常是 Control-D）或执行 halt/0 谓词：

```
?- halt.
$
```

用户的初始化文件

系统初始化后，系统会查阅（参见 consult/1）用户的初始化文件。使用路径别名 app_config（参见 file_search_path/2）absolute_file_name/3 搜索此文件。这是一个名为 swi-prolog 的目录，位于操作系统默认名称下，用于放置应用程序配置数据：

在 Windows 上，CSIDL 文件夹 CSIDL_APPDATA，通常为 C:Documents and SettingsusernameApplication Data。

如果设置了环境变量 XDG_DATA_HOME，请使用此变量。这遵循自由桌面标准。 /.config 的扩展。

可以使用以下调用找到该目录：

```
?- absolute_file_name(app_config(.), Dir, [file_type(directory)]).
Dir = '/home/jan/.config/swi-prolog'.
```

找到第一个启动文件后，它会被加载，Prolog 会停止寻找其他启动文件。可以使用“-f file”选项更改启动文件的名称。如果 File 表示绝对路径，则加载此文件，否则使用与默认启动文件相同的约定来搜索文件。最后，如果 file 为 none，则不加载任何文件。

安装程序提供了一个文件 customize/init.pl，其中包含通常用于自定义 Prolog 行为的（注释）命令，例如与编辑器交互、颜色选择或历史参数。许多开发工具都提供了菜单项，用于编辑启动文件并从系统框架启动新的启动文件。

另请参阅第 2.4 节和第 2.3 节中的 -s（脚本）和 -F（系统范围初始化）。

初始化文件和目标

使用命令行参数（参见第 2.4 节），可以强制 SWI-Prolog 加载文件并执行查询以进行初始化或非交互式操作。最常用的选项是 -f file 或 -s file，用于使 Prolog 加载文件，-g goal 用于定义初始化目标，-t goal 用于定义顶级目标。以下是直接从命令行启动应用程序的典型示例。

```
machine% swipl -s load.pl -g go -t halt
```

它告诉 SWI-Prolog 加载 load.pl，使用入口点 go/0 启动应用程序，并在完成 go/0 后退出 — 而不是进入交互式顶层 —。

命令行可能有多个 -g 目标出现。目标按顺序执行。各个目标的可能选择点被删减。如果目标失败，执行将以退出状态 1 停止。如果目标引发异常，则打印异常并以退出代码 2 停止进程。

-q 可用于抑制所有信息消息以及初始化目标失败时通常打印的错误消息。

在 MS-Windows 中，可以使用具有适当定义的命令行参数的快捷方式来实现相同的目的。一种常见的替代方法是编写一个文件 run.pl，其内容如下所示。双击 run.pl 将启动应用程序。

```
:- [load].                      % load program
:- go.                          % run it
:- halt.                        % and exit
```

2.11.1.1 节讨论了更多脚本选项，第 14 章讨论了运行时可执行文件的生成。运行时可执行文件是一种不需要 Prolog 系统即可交付可执行文件的方法。

命令行选项

SWI-Prolog 可以在以下模式之一中执行：

swipl –help swipl –version swipl –arch swipl –dump-runtime-variables 这些选项必须作为唯一选项出现。它们使 Prolog 打印一条信息消息并退出。请参阅第 2.4.1 节。 swipl [option ...] script-file [arg ...] 如果执行以 #!/path/to/executable [option ...] 开头的文件，则在 Unix 系统上传递这些参数。脚本文件后的参数在 Prolog 标志 argv 中可用。 swipl [option ...] prolog-file ... [[–] arg ...] 这是启动 Prolog 的正常方式。选项在第 2.4.2 节、第 2.4.3 节和第 2.4.4 节中进行了描述。 Prolog 标志 argv 提供对 arg 的访问... 如果选项后面跟着一个或多个 Prolog 文件名（即扩展名为 .pl、.prolog 或（在 Windows 上）在安装期间注册的用户首选扩展名的名称），则会加载这些文件。第一个文件在 Prolog 标志 related_file 中注册。此外，pl-win[.exe] 使用 working_directory/2 切换到此主要源文件所在的目录。 swipl -o output -c prolog-file ... -c 选项用于将一组 Prolog 文件编译成可执行文件。请参阅第 2.4.5 节。 swipl -o output -b bootfile prolog-file ... 引导编译。请参阅第 2.4.6 节。

4.1 信息性命令行选项

–arch 当作为唯一选项给出时，它会打印体系结构标识符（请参阅 Prolog 标志 arch）并退出。另请参阅 –dump-runtime-variables。 –dump-runtime-variables [=format] 当作为唯一选项给出时，它会打印一系列变量设置，这些变量设置可用于 shell 脚本来处理 Prolog 参数。swipl-ld 也使用此功能（请参阅第 12.5 节）。下面是使用此功能的典型示例。

```
eval `swipl --dump-runtime-variables`
cc -I$PLBASE/include -L$PLBASE/lib/$PLARCH ...
```

该选项后面可以跟 =sh，以 POSIX shell 格式转储（默认）或 =cmd，以 MS-Windows cmd.exe 兼容格式转储。 –help 当作为唯一选项给出时，它会总结最重要的选项。 –version 当作为唯一选项给出时，它会总结版本和体系结构标识符。 –abi-version 打印一个键（字符串），表示多个方面的二进制兼容性。请参阅第 2.21 节。

4.2 运行 Prolog 的命令行选项

请注意，布尔选项可以写为 –name (true)、–noname 或 –no-name (false)。它们在下面写为 –no-name，因为默认值为“true”。

-D name[=value] 将 Prolog 标志名称设置为值。加载初始保存状态后立即设置标志。如果标志已定义，则值将转换为标志的类型。如果标志未定义，则将其设置为一个数字，否则值表示数字，否则表示原子。如果没有给出 =value，则使用布尔值。如果 name 为无标志，则标志设置为 false。否则，标志名称设置为 true。name[=value] 可以紧跟在 -D 后面，也可以作为下一个命令行参数出现。

请注意，许多命令行选项都由 Prolog 标志反映。我们打算将它们作为同义词处理。目前，一些命令行标志在加载已保存状态完成之前会影响 Prolog 初始化，而其他一些标志在 Prolog 初始化之后可能不会更改。例如，未来版本将支持 -Dhome=dir 来更改 Prolog 安装目录的概念。

–debug-on-interrupt 立即启用中断信号 (Control-C、SIGINT) 上的调试。通常在进入交互式顶层时启用中断调试。此标志可用于在执行来自 -g 或初始化/[1,2] 的目标时在中断上启动调试器。另请参阅 Prolog 标志 debug_on_interrupt。

–home[=DIR] 使用 DIR 作为主目录。详情请参阅第 12.6 节。如果省略 DIR，则打印找到的位置并退出进程。如果找不到位置，则打印错误并以状态 1 退出进程。

–quiet 将 Prolog 标志 verbose 设置为 silent，抑制信息和横幅消息。也可用作 -q。

–no-debug 禁用调试。详情请参阅 current_prolog_flag/2 标志 generate_debug_info。

–no-signals 禁止 Prolog 处理任何信号，该属性有时对于嵌入式应用程序来说是理想的。此选项将标志 signals 设置为 false。详情请参阅第 12.4.25.1 节。请注意，仍安装了用于解除系统调用阻止的处理程序。可以使用 –sigalert=0 额外阻止此操作。请参阅 –sigalert。

–no-threads 在运行时禁用多线程版本的线程。另请参阅标志 threads 和 gc_thread。

–no-packs 不附加扩展包（附加组件）。另请参阅 attach_packs/0 和 Prolog 标志包。

–no-pce 启用/禁用 xpce GUI 子系统。默认情况下，如果已安装并且系统可以访问图形，则将其作为自动加载组件提供。使用 –pce 在用户空间中加载 xpce 系统，而 –no-pce 使其在会话中不可用。

–on-error =style 如何处理错误。有关详细信息，请参阅 Prolog 标志 on_error。

–on-warning =style 如何处理警告。有关详细信息，请参阅 Prolog 标志 on_warning。

–pldoc [=port] 在空闲网络端口上启动 PlDoc 文档系统，并在 http://localhost:port 上启动用户的浏览器。如果指定了端口，则服务器将在给定端口启动，并且不会启动浏览器。

–sigalert=NUM 使用信号 NUM（1 ... 31）来提醒线程。这是使 thread_signal/2 和派生的 Prolog 信号处理在目标线程被可中断的系统调用（例如，sleep/1，对大多数设备的读/写）阻塞时立即起作用所必需的。默认使用 SIGUSR2。如果 NUM 为 0（零），则未安装此处理程序。请参阅 prolog_alert_signal/2 以在运行时查询或修改此值。

–no-tty 仅限 Unix。控制终端的开关允许向跟踪器和 get_single_char/1 发送单字符命令。默认情况下，除非系统检测到它未连接到终端或它正在作为 GNU-Emacs 下级进程运行，否则将启用对终端的操作。另请参阅 tty_control。

–win-app 此选项仅在 swipl-win.exe 中可用，用于开始菜单项。如果导致 plwin 在文件夹... My Documents
Prolog 或其本地等效文件夹中启动（请参阅 win_folder/2）。如果不存在 Prolog 子目录，则创建该子目录。

-O 优化编译。有关详细信息，请参阅 current_prolog_flag/2 标志优化。

-l 文件 加载文件。此标志提供与其他一些 Prolog 系统的兼容性。10 它在 SWI-Prolog 中用于跳过使用初始化/2 指令指定的程序初始化。另请参阅第 2.11.1.1 节和初始化/0。

-s 文件 将文件用作脚本文件。在使用 -f 文件选项指定的初始化文件之后加载脚本文件。与 -f 文件不同，使用 -s 不会阻止 Prolog 加载个人初始化文件。

-f file 使用 file 作为初始化文件，而不是默认的 init.pl。‘-f none’ 停止 SWI-Prolog 搜索启动文件。此选项可用作 -s file 的替代，可阻止 Prolog 加载个人初始化文件。另请参阅第 2.2 节。

-F script 从 SWI-Prolog 主目录中选择一个启动脚本。脚本文件名为 .rc。默认脚本名称从可执行文件推导而来，从程序名称中获取前导字母数字字符（字母、数字和下划线）。-F none 停止查找脚本。旨在简单管理略有不同的版本。例如，可以编写脚本 iso.rc，然后使用 pl -F iso 选择 ISO 兼容模式，或者从 iso-pl 到 pl 建立链接。

-x bootfile 从 bootfile 而不是系统的默认启动文件启动。引导文件是使用 -b 或 -c 选项进行 Prolog 编译后生成的文件，或者是使用 qsave_program/[1,2] 保存的程序。

-p alias=path1[:path2 ... 为 file_search_path 定义路径别名。alias 是别名的名称，arg path1 ... 是别名的值列表。在 Windows 上，列表分隔符为 ;。在其他系统上，它是 :。值可以是别名 (值) 或路径名形式的术语。计算出的别名使用 asserta/1 添加到 file_search_path/2，因此它们位于别名的预定义值之前。有关使用此文件定位机制的详细信息，请参阅 file_search_path/2。

–traditional 此标志禁用 SWI-Prolog 版本 7 的最重要的扩展（请参阅第 5 节），这些扩展会导致与早期版本不兼容。具体来说，列表以传统方式表示，双引号文本由字符代码列表表示，不支持字典上的函数符号。如果存在此标志，则字典作为语法实体，以及作用于它们的谓词仍然受支持。

–

停止扫描更多参数，因此您可以在此之后为应用程序传递参数。请参阅 current_prolog_flag/2，使用标志 argv 获取命令行参数。

4.3 控制堆栈大小

从版本 7.7.14 开始，堆栈不再单独受限。相反，只限制组合大小。请注意，32 位系统仍存在 128Mb 的限制。请参阅第 2.19.1 节。默认情况下，64 位计算机上的组合限制为 1Gb，32 位计算机上的组合限制为 512Mb。

例如，要将堆栈限制为 32Gb，请使用以下命令。请注意，堆栈限制适用于每个线程。可以使用 thread_create 的 stack_limit(+Bytes) 选项控制单个线程。任何线程都可以调用 set_prolog_flag(stack_limit, Limit)（请参阅 stack_limit）来调整堆栈限制。此限制由从此线程创建的线程继承。

```
$ swipl --stack-limit=32g
```

–stack-limit=size[bkmg] 将 Prolog 堆栈的总大小限制为指定的大小。后缀将值指定为字节、千字节、兆字节或千兆字节。 –table-space=size[bkmg] 表空间的限制。这是用于保存 memoized11 答案以进行制表的尝试的存储位置。在 64 位计算机上默认为 1Gb，在 32 位计算机上默认为 512Mb。请参阅 Prolog 标志 table_space。 –shared-table-space=size[bkmg] 共享表的表空间限制。请参阅第 7.9 节。

4.4 从命令行运行目标

-g goal 在进入顶层之前执行目标。此选项可能出现多次。有关详细信息，请参阅第 2.3 节。如果没有初始化目标，系统将调用 version/0 来打印欢迎消息。可以使用 –quiet 或 -g true 来抑制欢迎消息。目标可能是一个复杂的术语。在这种情况下，通常需要使用引号来保护它不被 shell 扩展。以下是一种非交互运行目标的安全方法。如果 go/0 成功，-g halt 会导致进程停止并返回退出代码 0。如果失败，退出代码为 1；如果引发异常，退出代码为 2。 -t goal 使用目标作为交互式顶层，而不是默认目标 prolog/0。目标可能是一个复杂的术语。如果顶层目标成功，SWI-Prolog 将以状态 0 退出。如果失败，退出状态为 1。如果顶层引发异常，则将其打印为未捕获的错误，并重新启动顶层。此标志还确定由 break/0 和 abort/0 启动的目标。如果要阻止用户进入交互模式，请使用“-g goal -t halt”启动应用程序。

4.5 编译选项

-c file ... 将文件编译为“中间代码文件”。请参阅第 2.11 节。 -o output 与 -c 或 -b 结合使用以确定编译的输出文件。

4.6 维护选项

以下选项用于系统维护。仅供参考。

-b initfile ...-c file ... 引导编译。initfile ...由 C 编写的引导编译器编译，file ...由普通 Prolog 编译器编译。仅用于系统维护。 -d token1,token2,... 打印标有指示标记之一的 DEBUG 语句的调试消息。仅当使用 -DO_DEBUG 标志编译系统时才有效。仅用于系统维护。

UI 主题

UI（颜色）主题在两个方面发挥作用：写入控制台时以及用于基于 xpce 的开发工具（如 PceEmacs 或图形调试器）。彩色控制台输出基于 ansi_format/3。基于 print_message/2 的中央消息基础结构使用指定角色的 Prolog 术语标记消息（组件）。这通过钩子 prolog:console_color/2 映射到具体颜色。IDE 主题使用 xpce 类变量，这些变量在加载 xpce 时从 Prolog 初始化。

主题在文件搜索路径 library/theme 中作为 Prolog 文件实现。可以使用（例如）用户初始化文件中的以下指令加载主题（参见第 2.2 节）。

```
:- use_module(library(theme/dark)).
```

The theme file library(theme/auto) is provided to automatically choose a reasonable theme based on the environment. The current version detects the background color on xterm compatible terminal emulators (found on most Unix systems) and loads the dark theme if the background is‘darkish’.

The following notes apply to the different platforms on which SWI-Prolog is supported:

Unix/Linux If an xterm compatible terminal emulator is used to run Prolog you may wish to load either an explicit theme or library(theme/auto). Windows The swipl-win.exe graphical application can be themed by loading a theme file. The theme file also sets the foreground and background colours for the console.

5.1 主题支持状态

SWI-Prolog 8.1.11 中添加了主题支持。仅涵盖了部分 IDE 工具，并且唯一的附加主题（深色）不太平衡。主题文件与 IDE 组件之间的接口尚未建立。请通过改进深色主题做出贡献。一旦完成并正常运行，我们就可以开始添加新主题。

GNU Emacs 接口

SWI-Prolog 通过 sweep 包与 GNU Emacs 紧密集成。此包将 SWI-Prolog 嵌入为动态 Emacs 模块，允许直接从 Emacs Lisp 执行 Prolog 查询。随附的 Emacs 包 sweeprolog.el 可通过标准 Emacs 包管理器 package.el 安装，它基于此嵌入构建，为 GNU Emacs 中的 SWI-Prolog 提供完全集成的开发环境。

GNU Emacs 默认附带一个名为 prolog.el 的 Prolog 模式。与 sweeprolog.el 相比，此模式由于缺少合适的 Prolog 解析器而存在一些问题。Masanobu Umeda 编写的原始 prolog.el 自 1989 年以来一直包含在 GNU Emacs 中，2006 年 Stefan Monnier 在 prolog.el 中添加了对 SWI-Prolog 的明确支持。 2011 年，大部分原始实现已被 Stefan Bruda 最初为 XEmacs 端口编写的新 Prolog 模式所取代。Stefan Monnier 将 Bruda 的模式改编为 GNU Emacs，从那时起，他一直与其他 GNU Emacs 贡献者一起维护该模式。此模式的用户可以访问 https://www.metalevel.at/pceprolog/ 找到有用的配置建议。

其他可用于与 SWI-Prolog 配合使用的 Emacs 软件包包括：

https://www.metalevel.at/ediprolog/ 直接在 Emacs 缓冲区中与 SWI-Prolog 交互。 https://www.metalevel.at/etrace/ 使用 Emacs 跟踪 Prolog 代码。 https://emacs-lsp.github.io/dap-mode/page/configuration/#swi-prolog 通过 dap-mode 和 https://github.com/eshelyaron/debug_adapter 中的 debug_adapter 包，Emacs 中的 SWI-Prolog 支持调试适配器协议 (DAP) https://emacs-lsp.github.io/lsp-mode/page/lsp-prolog/ 通过 lsp-mode 和 https://github.com/jamesnvc/lsp_server 中的 lsp_server 包，Emacs 中的 SWI-Prolog 支持语言服务器协议 (LSP)