SWI-Prolog for MS-Windows

Jan Wielemaker
VU University Amsterdam
University of Amsterdam
The Netherlands
E-mail: jan@swi-prolog.org

Abstract

This document gets you started using SWI-Prolog on MS-Windows. It also describes the components and issues that are specific to MS-Windows. It is by no means a manual or Prolog tutorial. The reference manual is available online or can be downloaded in HTML and PDF format from the SWI-Prolog website, which also provides links to books, online tutorials and other Prolog-related material.

Table of Contents

1 Using SWI-Prolog

1.1 Starting Prolog and loading a program

The SWI-Prolog executable **swipl-win.exe** can be started from the *StartMenu* or by opening a `.pl` file holding Prolog program text from the Windows explorer.[1] The installation folder (by default `C:\Program Files\swipl`) contains a subfolder `demo` with the file `likes.pl`. This file can be opened in Prolog from the StartMenu, by opening `likes.pl` in the Windows explorer or by using the following command in the Prolog application. Be sure to get the quotes right and terminate the command with a full-stop (`.`).

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

If Prolog is started from the start menu it is passed the option `--win_app`, which causes it to start in the local equivalent of `MyDocuments\Prolog`. This folder is created if it does not exist.

1.2 Executing a query

After loading a program, one can ask Prolog queries about the program. The query below asks Prolog what food `sam' likes. The system responds with `X = <value>` if it can prove the goal for a certain *X*. The user can type the semi-colon (;) or spacebar. If you want another solution. Use the return key if you do not want to see the more answers. Prolog completes the output a full stop (.) if the user uses the return key or Prolog *knowns* there are no more answers. If Prolog cannot find (more) answers, it writes **false.** Finally, Prolog can answer using an error message to indicate the query or program contains an error.

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

?-
```

Note that the answer written by Prolog is a valid Prolog program that, when executed, produces the same set of answers as the original program.

1.3 Menu commands

The SWI-Prolog console provided by **swipl-win.exe** has a menu for accessing the most commonly used commands. We assume not all menu entries need to be explained in details. We make some exceptions:

File/Reload modified files
: This menu reloads all loaded source files that have been modified using the make/0 command described in section 1.5.

File/Navigator ...
: Opens an explorer-like view on Prolog files and the predicates they contain.

Settings/Font ...
: Allows for changing the font of the console. On some installations the default font gives redraw and cursor

dislocation problems. In this case you may wish to select an alternative. Some built-in commands assume non-proportional fonts.

Settings/User init file ...
Edits the user personalisation file. If no such file exists, it first installs a default file as `pl.ini` that contains commonly used settings in comments.

Settings/Stack sizes ...
Allows for defining the maximum size to which the various Prolog stacks are allowed to grow. The system defaults are chosen to make erroneous programs fail quickly on modest hardware. Programs with large data structures or many choice-points often need larger stacks. Note that an active Prolog process growing over the size of the physical memory of your computer can make the system extremely slow.

Run/Interrupt
Try to interrupt the running Prolog process. This is the same as using *Control-C*. Sometimes interrupts are not honoured or take very long to process. Closing the window twice provides a way to force Prolog to stop.

Run/New thread
Creates a new interactor window running in a separate thread of execution. This may be used to inspect the database or program while the main task continues.

Debug/Edit spy points ...
Edit break-points on predicates. From the PceEmacs editor (see section 1.4) break-points can also be set on specific calls from specific clauses.

Debug/Graphical debugger ...
Use the source-level debugger on the next spy- or break-point or other call that enables the debugger.

Help
The help menu provides various starting points to related documents. Items flagged with **(on www)** open your default internet browser on a page of the SWI-Prolog website.

1.4 Editing Prolog programs

There are three options for editing. One is to run an editor of choice in a separate window and use the make/0 command described below to reload modified files. In addition to this option Prolog can be used to locate predicates, modules and loaded files by specifying the editor of choice for use with the edit/1 command, described below. This is achieved by editing the personalisation file (see section 1.3) and following the instructions in the comments.

The default editor is built-in editor called *PceEmacs*. This editor provides colourisation support based on real-time parsing and cross-reference analysis of the program.

Other options for editing include GNU-Emacs, SWI-Prolog-Editor and the Eclipse-based PDT environment. See http://www.swi-prolog.org/IDE.html for an up-to-date overview.

1.5 Some useful commands

This section provides a very brief overview of important or commonly used SWI-Prolog predicates to control the environment.

consult(*:File*)
: Load a source file. On Windows, folders may be specified with the DOS/Windows `\`, which must be escaped, or by using the POSIX standard `/`. Especially when used in source code, `/` is to be preferred as it is portable. A Prolog list ([...]) can be used to abbreviate the consult command. The file extension (`.pl` as well as the selected alternative) can be omitted. Here are some examples:

`?- consult(likes).`	Load `likes.pl` from the current folder (see pwd/0).
`?- ['C:/Program Files/pl/demo/likes']`	Load `likes.pl` using absolute path.
`?- ['C:\\Program Files\\pl\\demo\\likes']`	Same using Windows-style path name

pwd
: Print working directory (folder).

ls
: List files in current directory.

edit
: If Prolog is started by opening a `.pl` file in the explorer, edit this file. Also available from the menu.

edit(+*Spec*)
: Edit file, predicate, module, etc., with the given name. If multiple items are named *Spec* it prompts for the desired alternative.

make
: Reload all files that have been changed since they were last loaded. Normally used after editing one or more files.

gtrace
: Start the graphical debugger. There are three ways to use this. Entered as a single goal at the top-level, the next query will be traced. Alternatively it can be used in conjunction with the goal to be debugged: `?- gtrace, run.` and finally you can include it in your program to start tracing at a particular point or under a particular condition:

```
        ...,
        (var(X) -> gtrace ; true),
        ...,
```

trace
: Same as gtrace, but text-based on the console.

apropos(+*Keyword*)
Search for all predicates that contain *Keyword* in their name or short description. If a GUI environment is available the results are hyperlinks. Otherwise use help/1 to get details on selected hits.

help(+*Spec*)
Give help on *Spec*, which is normally the name of a predicate or C interface function.

2 Using SWI-Prolog with C/C++

Using MinGW or a compiler with a compatible calling format you can write C or C++ code that can be loaded into SWI-Prolog and called as a predicate. You can also embed SWI-Prolog in C/C++ applications.

Details on how to interact with Prolog are in the SWI-Prolog reference manual. The mailing list archives and TWiki web provide problems and solutions to the many problems that may occur. Documentation of the `SWI-cpp.h` C++ include file is available from the package documentation. This section only discusses some Windows-specific issues.

2.1 Using MSVC

Because the current versions of SWI-Prolog are compiled and linked with MinGW, we are unsure about the status with regard to compiling extensions using MSVC and embedding SWI-Prolog into MSVC projects.

Please send your comments to the SWI-Prolog mailinglist, and/or mailto:bugs@swi-prolog.org.

First of all, add the `include` folder of the installation to the search path for headers and the `lib` folder to the search path for libraries. Both DLLs (extensions) or embedded executables should link to `libswipl.dll.a` and, if appropriate, to the multi-threaded DLL version of the MSVC runtime library.

To create extensions, create a Win32 DLL. To embed Prolog, care should be taken that Prolog can find the Prolog installation. For *development*, the simplest way to ensure this is by adding the installation `bin` folder to the `%PATH%` environment and calling **PL_initialise()** as illustrated below. **PL_initialise()** uses the path of the loaded `libswipl.dll` module to find the Prolog installation folder.[2]

```
{ static char *av[] = { "libswipl.dll", NULL };

  if ( !PL_initialise(1, av) )
  { <error>
  }
}
```

To create an executable that does not rely on Prolog one must create a saved state of the required Prolog code and attach this to the executable. Creating saved states is described with **qsave_program/2** in the reference manual. This can be attached to a state using the DOS command below to create `final.exe` from the executable produced by MSVC and the generated saved state.

```
> copy /b file.exe+file.state final.exe
```

2.2 Using swipl-ld.exe

The **swipl-ld.exe** automates most of the above complications and provides compatibility for common tasks on many platforms supported by SWI-Prolog. To use it with MinGW, set the `PATH` environment variables to include the SWI-Prolog binary folder as well as the MinGW binary folders (typically `C:\MinGW\bin`) to find **gcc**. An extension `myext.dll` can be created from the source `myext.c` using the command below. Add `-v` to see what commands are executed by **swipl-ld.exe**.

```
> swipl-ld.exe -shared -o myext myext.c
```

An embedded executable is created from C, C++ and Prolog files using

```
> swipl-ld.exe -o myexe file.c ... file.pl ...
```

3 The installation

3.1 Supported Windows versions

SWI-Prolog requiring Windows XP or later (XP, Vista, Windows-7). The download site of SWI-Prolog contains older binaries that run on older versions of Windows. We provide both 32-bit and 64-bit installers.

3.2 Choosing the file extension

By default, Prolog uses the `.pl` extension to indicate Prolog source files. Unfortunately this extension conflicts with the Perl language. If you want to use both on the same Windows machine SWI-Prolog allows you to choose a different extension during the installation. The extension `.pro` is a commonly used alternative. If portability is an issue, it is advised to use the alternative extension only for the *load file*, the source file that loads the entire program, and use the normal `.pl` extension for libraries and files loaded from other files.

3.3 Installed programs

The table below lists the installed components. Some components are marked (32-bits) or (64-bits). Most of this is because the 64-bits version is built using more recent tools and from more recent versions of required libraries using different naming conventions. This will probably be synchronised in the future.

Programs	
`bin\swipl-win.exe`	Default Windows application for interactive use.
`bin\swipl.exe`	Console-based version for scripting purposes.
Utilities	
`bin\swipl-ld.exe`	Linker front-end to make single-file mixed Prolog/C/C++ executables.
`bin\plrc.exe`	Manipulate Prolog resource files.
Important directories	
`bin`	Executables and DLL files
`library`	Prolog library

boot	Sources for system predicates
include	C/C++ header files for embedding or to create extensions
xpce	XPCE graphics system
xpce\prolog\lib	XPCE/Prolog library
DLLs and other supporting files	
boot32.prc	Initial Prolog state (32-bits)
boot64.prc	Initial Prolog state (64-bits)
\bin\libswipl.dll	The Prolog kernel
\bin\plterm.dll	The window for **swipl-win.exe**
\bin\pthreadVC2.dll	POSIX thread runtime library (64-bits)
Extension DLLs (plugins)	
\bin\cgi.dll	Gather CGI GET and POST arguments
\bin\double_metaphone.dll	Soundex (sounds similar)
\bin\memfile.dll	In-memory temporary `files'
\bin\odbc4pl.dll	ODBC interface
\bin\plregtry.dll	Windows registry interface
\bin\porter_stem.dll	Porter stemming implementation
\bin\random.dll	Portable random number generator
\bin\rdf_db.dll	RDF database
\bin\readutil.dll	Fast reading utility
\bin\sgml2pl.dll	SGML/XML parser
\bin\socket.dll	Prolog socket interface
\bin\table.dll	Access structured files as tables
\bin\time.dll	Timing and alarm library
\bin\xpce2pl.dll	The XPCE graphics system
\bin\zlib1.dll	Compression library (32-bit)
\bin\zlibwapi.dll	Compression library (64-bit)
\bin\zlib4pl.dll	Compression library interface

3.4 Installed Registry keys and menus

The filetype `.pl` or chosen alternative (see section 3.2) is associated to **swipl-win.exe**. A chosen folder (default SWI-Prolog) is added to the start menu holding shortcuts to Prolog and some related utilities. The following registry keys are in use. The 64-bit version uses `Prolog64` instead of `Prolog` as a key to accommodate installation of both versions on the same machine. Note that opening a `.pl` file can be associated with one of the installed Prolog versions only.

HKEY_LOCAL_MACHINE\Software\SWI\Prolog	
fileExtension	Extension used for Prolog files
group	Start menu group
home	Installation directory
HKEY_CURRENT_USER\Software\SWI\Plwin\Console	

HKEY_CURRENT_USER\Software\SWI\Prolog\Console	
Note: thread-windows store the same info in sub-keys	
`Height`	Height of window in character units
`Width`	Width of window in character units
`X`	Left edge of window in pixel units
`Y`	Top edge of window in pixel units
`SaveLines`	Number of lines available for scrollback

3.5 Execution level

The installer asks for the `admin` execution level (Vista and later) to be able to write shortcuts and registry keys.

3.6 Creating a desktop menu item

If you want a desktop entry for SWI-Prolog, right-drag **swipl-win.exe** to the desktop and select `Create shortcut'. Then edit the properties and add `--win_app` to the commandline to make the application start in `MyDocuments\Prolog`.

4 The SWI-Prolog community and foundation

4.1 Web-site and mailing lists

The SWI-Prolog website is located at http://www.swi-prolog.org/.

4.2 About license conditions

The SWI-Prolog license allows it to be used in a wide variety of environments, including closed-source commercial applications. In practice, redistribution and embedding is allowed, as long as *modifications* to the SWI-Prolog source are published following the Free Software rules.

The SWI-Prolog kernel and foreign libraries are licensed under the *Lesser General Public License* (LGPL). The Prolog files are licensed under the normal *General Public License* GPL with an additional statement that allows for embedding in proprietary software:

> As a special exception, if you link this library with other files, compiled with a Free Software compiler, to produce an executable, this library does not by itself cause the resulting executable to be covered by the GNU General Public License. This exception does not, however, invalidate any other reasons why the executable file might be covered by the GNU General Public License.

This exception is a proven construct used for *libgcc*, the GNU C-compiler runtime library.

4.3 Supporting SWI-Prolog

There are several ways to support SWI-Prolog:

- Extend the system with contributions.

- Improve the system by submitting bug reports and patches.
- Link to http://www.swi-prolog.org and refer to SWI-Prolog in publications.
- For commercial users, it may be profitable to sponsor development projects that make SWI-Prolog more useful for you and others. Example projects from the part include adding the initial garbage collector, unbounded integer support, SSL interface, (re-)introduction of the stack-shifter, avoid C-recursion on Prolog datastructures, the PlUnit test environment and the PlDoc documentation environment.

  Sponsoring development has several benefits: (1) it solves your bottlenecks, (2) others help debugging it and (3) it strengthens SWI-Prolog's position, which gives you better guarantees that the system remains actively developed and makes it easier to find resources and programmers.

Index