

Installation of SWI-Prolog

There are other Prolog implementations and other versions of SWI-Prolog, but we need exactly this version of SWI-Prolog, i.e. SWI-Prolog **8.0.2-1**, especially for the tester provided.

Step 1 is operating system dependent (and I encourage to use Windows or MacOS if possible, as the Linux installation is more complex).

Step 2 “Testing the installation” (outlined below) is the same for all operating systems.

(Below you also find Step 3 and Step 4 to begin and to complete the exercises.)

SWI-Prolog 8.0.2-1 – Installation on Mac

- Works as described on the SWI-Prolog web pages
- Before you install SWI-Prolog, you must install XQuartz 2.7.11!
- Then, you install SWI-Prolog
- Important: if not already done in the installation process, add the **correct name** and path of the SWI-Prolog app
(in my case: `/Applications/SWI-Prolog-8.0.2.app/Contents/MacOS`)
to the path variable `$PATH` which is set in `/Users/stb/.bash_profile`

Then check your installation, call SWI-Prolog from a terminal shell as follows:

- In the shell, type in
 `...> ./swipl`
which should open the interpreter, i.e., you should see something like
 `...`
 `?-`
If you get an error that the command `swipl` is not known,
check that your `PATH` variable includes a path to SWI-Prolog.
- When you are in the interpreter, i.e. when you see
 `?-`
you should type in goals as described below.

Basically, installation on Mac works as described on the SWI-Prolog web pages.
You also may have to add the directory of the installed SWI-Prolog to you `PATH` variable.

SWI-Prolog 8.0.2-1 – Installation on Linux

The installation on Linux is a bit more complex and has been successfully tested for Debian.
Look at the SWI-Prolog website. Make sure that you install version 8.0.2-1.

Furthermore, in the Pool in F1, there are installations for openSUSE available.

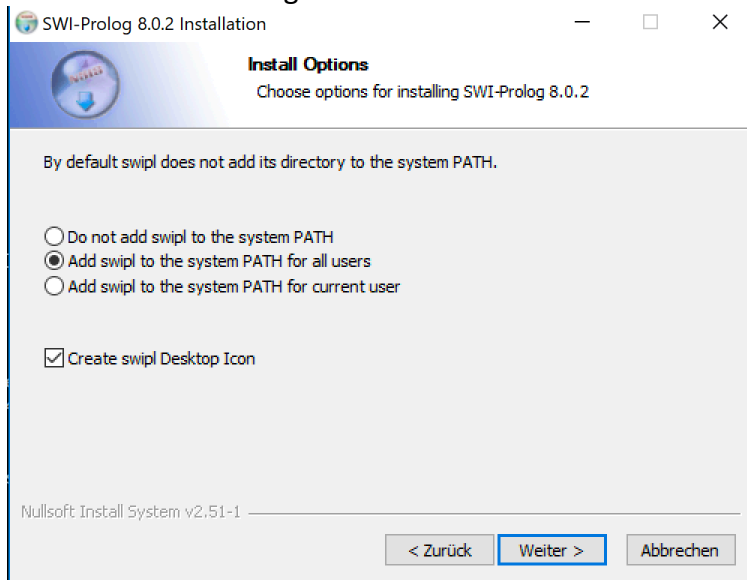
SWI-Prolog 8.0.2-1 installation on Windows:

Download the file swipl-8.0.2-1.x64.exe.

In Windows double-Click the file to install Prolog
(or in a command shell run `> swipl-8.0.2-1.x64.exe`).

Then ... Allow SWI-Prolog to install programs.

Select that SWI-Prolog should be included into the PATH variable for all users.



Then check your installation, call SWI-Prolog from a terminal shell as follows:

- In the shell, type in

...> **swipl-win**

which should open the interpreter, i.e., you should see something like

...

?-

If you get an error that the command swipl-win is not known, check that your PATH variable includes a path to SWI-Prolog. For example, if the location of your SWI-Prolog installation is `C:\Program Files\swipl\bin` , add the command

`> SET PATH="C:\Program Files\swipl\bin";{PATH}`

to your config.sys file

- When you are in the SWI-Prolog interpreter, i.e. when you see

?-

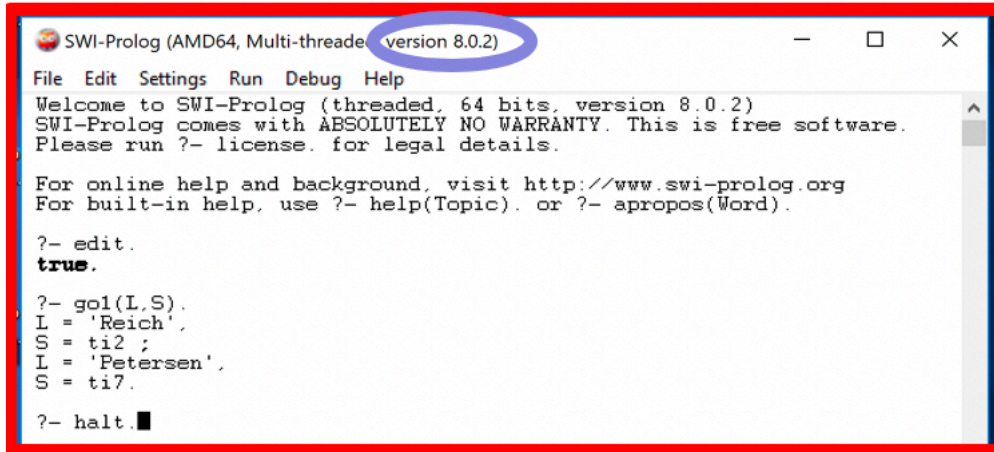
you could type in goals as described below.

Basically, SWI-Prolog 8.0.2 installation on **Windows** works as described on the SWI-Prolog web pages. You also may have to add the directory of the installed SWI-Prolog to your PATH variable.

Step 2: Testing the installation:

To work with SWI-Prolog, you double-click a .pl-file and continue as follows:

- A double click on the file **db1.pl** (or any other .pl file) opens that file in the **SWI-Prolog-Interpreter-Window**.



```
SWI-Prolog (AMD64, Multi-threaded version 8.0.2)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

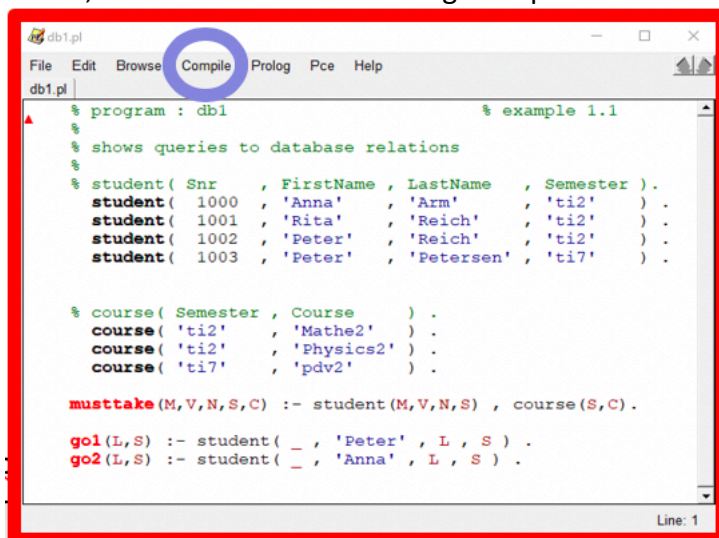
For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- edit.
true.

?- go1(L,S).
L = 'Reich',
S = ti2 ;
L = 'Petersen',
S = ti7.

?- halt.■
```

- In the SWI-Prolog-Interpreter, you could type in goals like “?- X is 3+4 . “ RETURN
- You can also type in in goals like “?- member(X,[1,2,3]) . “ RETURN , or “?- go1(L,S).“ and, after seeing the first answer, you can type in “ ; ” to see the next answer
- **To edit your program**, type in “?- edit . “ RETURN in the **SWI-Prolog-Interpreter**. **This is the way how you should open the SWI-Prolog-Editor**.
You should see a second open window. Thereafter, you should keep both windows open, one for the SWI-Prolog-Editor, the other for the SWI-Prolog-Interpreter.
(If the NotePad-Editor open ins Windows, you did call swipl instead of swipl-win.)
- **To compile an edited (and saved) program**, you should use the **SWI-Prolog-Editor(!)-Window**, click **compile**→**compile buffer** → **yes(Save changes)**.
A (non-recommended) alternative is to save the changes in the Editor, close the editor, or switch to the SWI-Prolog-Interpreter Window, and there type in “?- make. “



```
db1.pl
File Edit Browse Compile Prolog Pce Help
db1.pl
% program : db1 % example 1.1
%
% shows queries to database relations
%
% student( Snr , FirstName , LastName , Semester ) .
student( 1000 , 'Anna' , 'Arm' , 'ti2' ) .
student( 1001 , 'Rita' , 'Reich' , 'ti2' ) .
student( 1002 , 'Peter' , 'Reich' , 'ti2' ) .
student( 1003 , 'Peter' , 'Petersen' , 'ti7' ) .

% course( Semester , Course ) .
course( 'ti2' , 'Mathe2' ) .
course( 'ti2' , 'Physics2' ) .
course( 'ti7' , 'pdv2' ) .

musttake(M,V,N,S,C) :- student(M,V,N,S) , course(S,C) .

go1(L,S) :- student( _ , 'Peter' , L , S ) .
go2(L,S) :- student( _ , 'Anna' , L , S ) .

Line: 1
```

- Then, use the **SWI-Prolog- Interpreter-Window** for typing in queries, e.g.
“?- student(Snr, Firstname, Lastname, Semester). ” RETURN
- Furthermore, use “;” (semicolon) after you have seen the first solution to see more (or all) solutions, in the SWI-Prolog- Interpreter-Window.
- **To leave Prolog**, you type in “?- halt . ” RETURN

Additionally, make sure that you can also use SWI-Prolog from a terminal shell as follows:

- Go to the directory containing your file, e.g. db1.pl (or any other .pl file)
- In **MacOS** and **Unix**, type in “\$./swipl -s db1.pl” which opens the file db1.pl in the interpreter.
- In **Windows**, type in “> swipl-win -s db1.pl” which opens the file db1.pl in the interpreter.
- Then, you could type in goals as described above.

Once you have done this successfully, testing the installation is completed, and you can start with the exercises.

Step 3: Starting with the exercises:

Make sure that you have done Step 1 and Step 2 successfully, i.e., that you know how start and halt SWI-Prolog, how to call the built-in editor, how to compile files, how to submit queries, and how to see more answers of queries.

Make sure that you have the previous knowledge required and described on the prerequisites page.

- Then, download the file exercise1.pl
- Open the file by calling the **SWI-Prolog-Interpreter** -
either by double-clicking the file exercise1.pl (or by opening it from command line,
i.e. using **swipl-win -s exercise1.pl** in **Windows**
or using **./swipl -s exercise1.pl** in **MacOS** or **Linux**
- In the **SWI-Prolog-Interpreter-Window** type
?- edit.
to call the editor.
- Then, in the **SWI-Prolog-Editor** remove the comment before q1(Item) :- ...
and save and compile the file.
- Then, in the **SWI-Prolog-Interpreter-Window** type in the query
“?- q1(Item). “ RETURN
and check that you get all results i.e., use “;” (semicolon) after you have seen the first
solution to see more (or all) solutions, in the SWI-Prolog- Interpreter-Window.
- Thereafter switch to the **SWI-Prolog- Editor-Window** to type in your solution rule for
the query q2. Again, and save and compile the file, switch to the **SWI-Prolog-
Interpreter-Window** type in the query
“?- q2(Supplier). “ RETURN
- If your query qN does not show the desired results, switch to the **SWI-Prolog- Editor-
Window** to edit the rule for qN, compile as described above, and test qN again.
- Once qN is answered correctly, continue with qN+1.
- **To leave Prolog**, you type in “?- halt . “ RETURN

How to deal with typical warnings

How to react to warnings “singleton variables ...”:

- A singleton variable is a variable occurs only once in a clause – either because of a
typos or because it could (and should) be replaced by an anonymous variable “_”.
→ if it is a typos, remove the typos. Otherwise, or use “_”

How to treat warnings “redefining clauses ...”:

You can either compile your code again or ignore this warning

Step 4: Completing the exercises before you present your solutions:

Be prepared that you can explain your solutions during the exercise meetings.

Before you explain your solutions, do a double check using the following test program.

The version of the test program differs depending on the operating system:

1. for Windows use exercise1sol.exe
2. for MacOS use exercise1sol
3. for OpenSuse Linux use exercise1solsuse

1. Assuming that your program is called **exercise1.pl** and that you want to check your solution to query **q13**, in **Windows** use a CMD-Shell to call:

> exercise1sol exercise1 13

(here, you shall use the Windows version exercise1sol.exe)

2. Assuming that your program is called **exercise1.pl** and that you want to check your solution to query **q13**, in **MacOS** use a shell to call:

> ./exercise1sol exercise1 13

(here, you shall use the MacOS version exercise1sol)

3. Assuming that your program is called **exercise1.pl** and that you want to check your solution to query **q13**, in **OpenSuse Linux** use a shell to call:

> ./exercise1solsuse exercise1 13

(here, you shall use the OpenSuse Linux version exercise1solsuse)

Actions to be taken depending on test program feedback:

If these test program returns "... FATAL ERROR ...",
double check that you have installed SWI-Prolog version 8.0.2-1
and that you have used the correct operating system dependent version.

If you get an error message query q1/1 is undefined, make sure that you have named your queries q1, q2, ... and that the number of parameters is correct (i.e. each of the first queries q1, ..., q16 and q19 must have one parameter.)

If the test program tells you that your solution differs from the wanted solution look at the counter example presented to show the difference - and rewrite your query.

If the test program tells you that your solution is equivalent to the wanted solution, but it takes more clauses and/or more goals, try to improve your solution (i.e. find a shorter solution).

If the test program tells you that your solution is most likely correct, the tester could not detect any difference. Then you should prepare for explaining your solution.