SICStus Prolog Release Notes

by the Intelligent Systems Laboratory

Swedish Institute of Computer Science
PO Box 1263
SE-164 29 Kista, Sweden

Release 3.11.0
October 2003

Swedish Institute of Computer Science
sicstus-request@sics.se http://www.sics.se/sicstus/

Copyright © 1995-2003 SICS

Swedish Institute of Computer Science
PO Box 1263
SE-164 29 Kista, Sweden

Permission is granted to make and distribute verbatim copies of these notes provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of these notes under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of these notes into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by SICS.

Table of Contents

1 Supported platforms 1

2 Release Notes and Installation Guide for UNIX 2

2.1 Installation 2
2.1.1 Prerequisites 2
2.1.1.1 C Compiler and Linker 2
2.1.2 The Installation Script 2
2.1.3 The Uninstallation Script 3
2.2 Platform Specific Notes 3

3 Release Notes and Installation Guide for Windows 6

3.1 Requirements 6
3.2 Installation 6
3.3 Windows Notes 7
3.4 Command Line Editing 7
3.5 The Console Window 8
3.5.1 Console Preferences 9
3.6 Windows Limitations 9

4 Tcl/Tk Notes 12

4.1 The Tcl/Tk Terminal Window 12

5 Jasper Notes 13

5.1 Supported Java Versions 13
5.2 Getting Started 15
5.2.1 Windows 15
5.2.2 UNIX 16
5.2.3 Running Java from SICStus 16
5.2.4 Running SICStus from Java 17
5.3 Jasper Package Options 18
5.4 Multi Threading 19
5.5 Known Bugs and Limitations 20
5.6 Java Examples Directory 20
5.7 Resources 21

6 Visual Basic Notes 22

7 Berkeley DB Notes 23

8 The Emacs Interface 24
8.1 Installation 24
8.1.1 Installing On-Line Documentation 24

9 Revision history 25
9.1 Changes in Release 3 25
9.2 Changes Introduced in 3#4 26
9.3 Changes Introduced in 3#5 27
9.4 Changes Introduced in 3#6 27
9.5 Changes Introduced in Version 3.7 27
9.6 Changes Introduced in Version 3.7.1 29
9.7 Changes Introduced in Version 3.8 29
9.7.1 Wide Character Support 30
9.7.2 Breakpointing Debugger 30
9.7.3 ISO Compliance 31
9.7.4 Generic New Features 31
9.7.5 New Features in library(jasper) 32
9.7.6 New Features in library(clpfd) 33
9.7.7 Bugs Fixed in Version 3.8 33
9.8 Changes Introduced in Version 3.8.1 34
9.9 Changes Introduced in Version 3.8.2 35
9.10 Changes Introduced in Version 3.8.3 36
9.11 Changes Introduced in Version 3.8.4 37
9.12 Changes Introduced in Version 3.8.5 39
9.13 Changes introduced in version 3.8.6 42
9.14 Changes introduced in version 3.8.7 43
9.15 Changes Introduced in Version 3.9 45
9.15.1 Message and Query System 45
9.15.2 Determinacy Checker 45
9.15.3 Cross-Referencer 45
9.15.4 Common Object Model Client 46
9.15.5 The PiLLoW Web Programming Library 46
9.15.6 New CLPFD Features 46
9.15.7 All-in-one Executables 47
9.15.8 Multiple SICStus Run-Times in a Process 47
9.15.9 Miscellaneous 47
9.15.10 Incompatibilities with Previous Versions 49
9.15.11 Bugs Fixed in 3.9 51
9.15.12 Known Problems with This Release 51
9.16 Changes Introduced in Version 3.9.1 52
9.17 Changes Introduced in Version 3.10 54
9.17.1 New Features 54
9.17.2 Bugs Fixed 55
9.17.3 Other Changes 56
9.18 Changes Introduced in Version 3.10.1 57
9.18.1 New Features 57
9.18.2 Debugger Changes 57
9.18.3 Bugs Fixed 58

9.18.4 Other Changes 58
9.19 Changes Introduced in Version 3.11.0 59
9.19.1 New Features 59
9.19.2 Bugs Fixed 60
9.19.3 Other Changes 60

10 Generic Limitations 62

11 Contact Information 63

1 Supported platforms

Release 3.11 is actively being supported on the following platforms:

- Compaq Tru64 UNIX V5.1 (formerly Digital UNIX) on ALPHA.
- Linux Intel. SICStus is developed on RedHat 7.x.
- HP-UX 11.x. PA32, HP-UX 1.6i (B.11.22) Itanium 64bit (IPF64).
- IRIX 6.5.
- Solaris 7 SPARC 32/64 bit. Later versions of Solaris is expected to work.
- Solaris 7 Intel. Later versions of Solaris is expected to work.
- Windows 95/98/ME/NT/2000/XP. Windows 95 is deprecated, Windows 2000 or newer Windows NT-based OS is recommended. SICStus is developed on Windows XP.
- AIX 4.3.3, AIX 5L 5.1 32/64 bit.
- MacOS X 10.2. We build and test on the latest version available.

In addition, earlier versions of SICStus have been verified to run on a number of other platforms. Contact `sicstus-support@sics.se` if you have any questions about a particular platform (listed or not listed).

2 Release Notes and Installation Guide for UNIX

This chapter assumes that the environment variable `PATH` includes `<prefix>/bin`, where `<prefix>` points to the SICStus installation directory. The installation directory is specified during installation; see Section 2.1 [UNIX installation], page 2. For example:

```
csh,tcsh> setenv PATH "/usr/local/bin:$PATH"
sh,bash,ksh> export PATH="/usr/local/bin:$PATH"
```

2.1 Installation

Installation of SICStus under UNIX is performed by an installation (Shell) script `InstallSICStus`, which interacts with the user to obtain options such as where to install SICStus. As of SICStus 3.9.1, the Java based *SICStus Installer Tool* is a graphical front-end to the installation script, which automates downloading and installation. The SICStus Installer Tool is available from the download page. Use of the SICStus Installer Tool is strictly optional but may be convenient, especially on platforms such as Mac OS X, that, by default, lack C compiler.

2.1.1 Prerequisites

2.1.1.1 C Compiler and Linker

A full SICStus installation requires a C compiler and a linker to perform final link steps on the installation machine.

If a C compiler is not available, it is possible to use a *pre-built installation* on some platforms (e.g. Mac OS X, 64bit SPARC Solaris).

Pre-built installation is only recommended as a last resort; it is available from the SICStus Installer Tool or by invoking `InstallSICStus` with the '`--all-questions`' argument.

A disadvantage with the pre-built installation is that SICStus libraries that interface to third-party products (Tcl/Tk, Berkeley DB, Java) may not work, or may require environment variables such as `LD_LIBRARY_PATH` to be set. Another disadvantage is that `spld` and `splfr` may not work unless you manually adjust the `spld` configure file. Of course, neither `spld` nor `splfr` will work anyway if you do not have a C compiler.

2.1.2 The Installation Script

Most users will install SICStus from a binary distribution. These are available for all supported platforms. Information on how to download and unpack the binary distribution is sent by email when ordering SICStus.

Binary distributions are installed by executing a interactive installation script called `InstallSICStus`. Type

```
% ./InstallSICStus
```

and follow the instructions on the screen. As an alternative, the SICStus Installer Tool can be used to download the SICStus files and invoke the installation script.

During installation, you will be required to enter your site-name and license code. These are included in the download instructions.

The installation program does not only copy files to their destination, it also performs final link steps for some of the executables and for the library modules requiring third-party software support (currently `library(bdb)`, `library(tcltk)`, and `library(jasper)`). This is done in order to adapt to local variations in installation paths and versions.

Invoke `InstallSICStus` with the '`--help`' argument to get a list of options.

Compiling SICStus from the sources requires a source code distribution, available on request for customers with maintenance contract. Contact `sicstus-support@sics.se` for more info.

Instructions for compiling and installing SICStus from the source code is available in the files `README` and `INSTALL` in the source code distribution.

2.1.3 The Uninstallation Script

To uninstall SICStus the script `UnInstallSICStus` can be run. It is created during installation in the same directory as `InstallSICStus`.

2.2 Platform Specific Notes

This section contains some installation notes that are platform specific under UNIX.

- **Solaris SPARC 64bit**: You cannot install (or build) the 64 bit version of SICStus using `gcc 2.x`. You need to use the Sun Workshop/Forte compiler, version 5.0 or later. `InstallSICStus` will try to find it during installation but if that fails, you can set the environment variable `CC` to e.g. '`/opt/SUNWspro/bin/cc`' before invoking `InstallSICStus`. Using `gcc 3.x` does seem to work but has not yet received much

testing. To install with `gcc 3.x`, set the environment variable `CC` appropriately before invoking `InstallSICStus`.

- **Solaris SPARC 64bit**: The following libraries are not supported: `library(bdb)`, `library(tcltk)`.
- **Solaris** Library `timeout` does not work with Java. The problems seems to be a limitation in the Solaris `setitimer` for process with multiple threads. A possible workaround on Solaris 8 is to use an alternative thread library by setting `LD_LIBRARY_PATH` to '`/usr/lib/lwp`'. This alternative thread library is not available prior to Solaris 8 and it is the default in Solaris 9.
- **MacOS X**: Installing and using `library(tcltk)` requires `Tk`, which is not installed on MacOS X by default (see Chapter 4 [Tcl/Tk Notes], page 12).
- **MacOS X**: An executable built with `spld` will only work if there is a properly configured subdirectory '`sp-3.11.0`' in the same directory as the executable; see section "Runtime Systems on UNIX Target Machines" in *the SICStus Prolog Manual*.

  Alternatively, the option '`--wrapper`' can be passed to `spld`. In this case a wrapper script is created that will set up various environment variables and invoke the real executable.
- **MacOS X**: It is not possible to prelink dynamic foreign resources into a dynamically linked Prolog executable. That is, except for data resources, `spld --resources ...` does not work, whereas `spld --static --resources ...` will. This is no great loss; pre-linking dynamic foreign resources is pointless, at best.
- **MacOS X**: When using third-party products like `BDB`, you may need to set up `DYLD_LIBRARY_PATH` so that the MacOS X dynamic linker can find them. When using the SICStus development executable (`sicstus`), a wrapper script does this automatically.
- **MacOS X**: File names are encoded in UTF-8 on MacOS X. This is handled correctly by SICStus. If SICStus encounters a file name that is not encoded in UTF-8, it will interpret the name as Latin 1 (ISO 8859/1) instead. This can happen on file systems where files have been created by some other OS than MacOS X, e.g., on network file servers accessed by other UNIX flavors or Windows.
- **MacOS X**: Sometimes, the default limit on the process's data-segment is unreasonably small, which may lead to unexpected memory allocation failures. To check this limit, do

  ```
  tcsh> limit data
  datasize  6144 kbytes
  bash> ulimit -d
  6144
  ```

  This indicates that the maximum size of the data-segment is only 6 Mb. To remove the limit, do

  ```
  tcsh> limit datasize unlimited
  datasize  unlimited
  bash> ulimit -d unlimited
  bash> ulimit -d
  unlimited
  ```

  Please note: `limit` (`ulimit`) is a shell built-in in `csh`/`tcsh` (`sh`/`bash`). It may have a different name in other shells.

> **Please note**: The limit will also affect SICStus when started from within Emacs, e.g., with *`M-x run-prolog`*. To change the limit used by Emacs and its sub-processes (such as SICStus) you will need to change the limit in the shell used to start Emacs. Alternatively you can create a shell wrapper for the `emacs` command.

- **MacOS X 10.2**: When running `spld` or `splfr` the Apple linker sometimes issues a warning similar to:

  ```
  ld: warning dynamic shared library: ... not made a weak library in
  output with MACOSX_DEPLOYMENT_TARGET environment variable set to: 10.1
  ```

  This warning is harmless and can be ignored.
- **IRIX** `spld --moveable` implies '`--wrapper`'. This is due to limitations in the IRIX run-time loader.
- **IRIX** Pre-linked dynamic resources will not be found at run-time if the executable was built with `spld --moveable`. This should not be a problem; pre-linked dynamic resources are not recommended anyway.
- **AIX** Applications that embed the SICStus run-time need to use the '`Large Address-Space Model`'. This is done automatically by `spld`. If you do not use `spld`, you need to set this option yourself. This is achieved by linking the executable using the '`-bmaxdata`' option. An alternative may be to set the environment variable `LDR_CNTRL` appropriately. See the documentation for the AIX command `ld`.

3 Release Notes and Installation Guide for Windows

This chapter assumes that the environment variable `PATH` includes `%SP_PATH%\bin`, where `SP_PATH` points to the SICStus installation directory (typically `C:\Program Files\SICStus Prolog 3.11.0`. Here, `%SP_PATH%` is just a placeholder; you usually do not need to set the environment variable `SP_PATH`, but see section "CPL Notes" in *the SICStus Prolog Manual*. For example:

```
C:\> set PATH=C:\Program Files\SICStus Prolog 3.11.0\bin;%PATH%
```

You may also want to include the paths to Tcl/Tk (see Chapter 4 [Tcl/Tk Notes], page 12), Java (see Section 5.2 [Getting Started], page 15), and Berkeley DB (see Chapter 7 [Berkeley DB Notes], page 23).

3.1 Requirements

- Operating environment: Microsoft Windows 98, ME, NT 4.0, 2000, XP. Windows 2000 or newer NT-based OS is recommended
- Processor: 386 or better.
- Available user memory: 16 Mbytes, more is recommended.
- Available hard drive space: 20 Mbytes (approximate)
- For interfacing with C or C++, or for using `spld` or `splfr`: Microsoft Visual C++ 6.0 or later.

  As an unsupported alternative to the Microsoft Visual Studio tools, you can use '`Microsoft eMbedded Visual C++ 4.0`' or '`Microsoft eMbedded Visual Tools 3.0`' available *for free* at the download area of `http://msdn.microsoft.com/`. These tools come with compilers and linkers not only for embedded platforms but also for the ordinary x86 Windows environment. In addition to the compiler tools, you will need the Microsoft Platform SDK available from the same place.

3.2 Installation

The development system comes in two flavors:

1. A console-based executable suitable to run from a DOS-prompt, from batch files, or under Emacs. See Section 3.4 [Command Line Editing], page 8.
2. A windowed executable providing command line editing and menus.

The distribution consists of a single, self-installing executable ('`InstallSICStus.exe`') containing development system, runtime support files, library sources, and manuals. Note that the installer itself asks for a password, when started. This is different from the license code.

Installed files on a shared drive can be reused for installation on other machines.

SICStus Prolog requires a license code to run. You should have received from SICS your site name, the expiration date and the code. This information is normally entered during installation:

```
Expiration date: ExpirationDate
Site: Site
License Code: Code
```

but it can also be entered later on by executing the following commands at a command prompt (sicstus3.11_linux differs between platforms):

```
% splm -i Site
% splm -a sicstus3.11_linux ExpirationDate Code
```

under Windows NT/2000/XP, `splm` must be run by a user with Administrative rights. As of SICStus 3.10, the windowed version of SICStus (`spwin.exe`) has a menu item for license entry, making `splm` unnecessary under Windows.

3.3 Windows Notes

- The file name arguments to `splfr` and `spld` should not have embedded spaces. For file names with spaces, you can use the corresponding short file name.
- Windows 95/98/ME: The shortcut installed in the '`Start`' menu (e.g. '`Start\Programs\SICStus Prolog 3.11.0`') may not work immediately after installation. Restarting after installing SICStus appears to cure this. If this does not help, you can add your own shortcut to e.g. '`C:\Program Files\SICStus Prolog 3.11.0\bin\spwin.exe`'.
- Selecting the '`Manual`' or '`Release Notes`' item in the '`Help`' menu may give an error message similar to '`... \!Help\100#!Manual.lnk could not be found`'. This happens when Adobe Acrobat Reader is not installed or if it has not been installed for the current user. Open '`C:\Program Files\SICStus Prolog 3.11.0\doc\pdf\`' in the explorer and try opening '`relnotes.pdf`'. If this brings up a configuration dialog for Adobe Acrobat, configure Acrobat and try the '`Help`' menu again. Alternatively, you may have to obtain Adobe Acrobat. It is available for free from http://www.adobe.com/.
- Windows NT, 2000 and later: We recommend that SICStus be installed by a user with administrative privileges and that the installation is made '`For All Users`'.

  If SICStus is installed for a single user, then SICStus will not find the license information when started by another user. In this case, the windowed version of SICStus (`spwin`) will put up a dialog where a license can be entered.
- The first time the installer is run, it will install necessary system files for supporting the new '`Windows Installer`' technology from Microsoft. This will fail unless the user has administrative rights. A typical symptom is an error message asking for '`msiexec`'. The Windows Installer technology is already part of Windows 2000 and later.

3.4 Command Line Editing

Command line editing supporting Emacs-like commands and IBM PC arrow keys is provided in the console-based executable. The following commands are available:

`^h`	erase previous char
`^d`	erase next char
`^u`	kill line
`^f`	forward char
`^b`	backward char
`^a`	begin of line
`^e`	end of line
`^p`	previous line
`^n`	next line
`^i`	insert space
`^s`	forward search
`^r`	reverse search
`^v`	view history
`^q`	input next char blindly
`^k`	kill to end of line

Options may be specified in the file '`%HOME%\spcmd.ini`' as:

```
Option Value
```

on separate lines. Recognized options are:

`lines` *Value* is the number of lines in the history buffer. 1-100 is accepted; the default is 30.

`save` *Value* is either 0 (don't save or restore history buffer) or 1 (save history buffer in '`%HOME%\spcmd.hst`' on exit, restore history from the same file on start up.

The command line editing is switched off by giving the option '`-nocmd`' when starting SICStus. Command line editing will be automatically turned off if SICStus is run with piped input (e.g. from Emacs).

3.5 The Console Window

The console window used for the windowed executable is based on code written by Jan Wielemaker `<jan@swi.psy.uva.nl>`.

In SICStus 3.8 the console was enhanced with menu access to common Prolog flags and file operations. Most of these should be self explanatory. The '`Reconsult`' item in the '`File`' menu reconsults the last file consulted with use of the '`File`' menu. It will probably be replaced in the future with something more powerful.

Note that the menus work by simulating user input to the Prolog top level or debugger. For this reason, it is recommended that the menus only be used when SICStus is waiting for a goal at the top-level (or in a break level) or when the debugger is waiting for a command.

3.5.1 Console Preferences

The stream-based console window is a completely separate library, using its own configuration info. It will look at the environment variable `CONSOLE`, which should contain a string of the form *name*:*value*{,*name*:*value*} where *name* is one of:

`sl` The number of lines you can scroll back. There is no limit, but the more you specify the more memory will be used. Memory is allocated when data becomes available. The default is 200.

`rows` The initial number of lines. The default is 24.

`cols` The initial number of columns. The default is 80.

`x` The X coordinate of the top-left corner. The default is determined by the system.

`y` The Y coordinate of the top-left corner. The default is determined by the system.

Under Windows 95 or 98, you will normally specify this in your '`autoexec.bat`' file. Here is an example:

```
% set CONSOLE=sl:600,x:400,y:400
```

Under Windows NT and Windows 2000 or newer, you would use the '`System`' Control Panel.

Many of these settings are also accessible from the menu '`Settings`' of the console.

3.6 Windows Limitations

- File paths with both '`/`' and '`\`' as separator are accepted. SICStus returns paths using '`/`'. Note that '`\`', since it is escape character, must be given as '`\\`' unless the Prolog flag `character_escapes` is set to `off`.

- All file names and paths are converted to lowercase when expanded by `absolute_file_name/3`.
- Interruption by *C-c* only works in certain circumstances
  - *C-c* always works while Prolog code is executing. This is true for both `sicstus` and `spwin`.
  - *C-c* always works while `spwin` is in a blocking read from the GUI window.
  - *C-c* will interrupt a blocking read from the (default) standard input if `sicstus` is attached to a console window. That is, if it is started from a command prompt window.
  - *C-c* will not interrupt a blocking read from a pipe or other non-terminal. In particular, it will not interrupt a blocking read in SICStus if SICStus gets its input from a pipe, such as when running SICStus within Emacs.
  - Blocking system calls, such as those used by `library(sockets)`, are not interruptible by *C-c* in any kind of SICStus executable.
- In the windowed executable, the `user_error` stream is line buffered.
- Emacs Issues: Running under Emacs has been tried with recent versions of GNU Emacs and XEmacs. See Chapter 8 [The Emacs Interface], page 24.
  - In both GNU Emacs and XEmacs *C-c C-c* (`comint-interrupt-subprocess`) will *not* interrupt a blocking read from standard input. The interrupt will be noted as soon as some character is sent to SICStus. The characters typed will not be discarded but will instead be used as debugger commands, sometimes leading to undesirable results.
  - Choosing '`Send EOF`' from the menu, i.e. `comint-send-eof`), closes the connection to the SICStus process. This will cause SICStus to exit. This problem cannot be fixed in SICStus; it is a limitation of current versions of FSF Emacs and XEmacs (at least up to FSF Emacs 20.7 and XEmacs 21.5).

    Instead of sending and end of file, you can enter the symbol `end_of_file` followed by a period. Alternatively, a *C-z* can be generated by typing *C-q C-z*.
- As of SICStus 3.10, under Windows NT/2000/XP, `statistics(runtime, ...)` measures user time of the thread running SICStus (the main thread) instead of process user time. This makes `statistics(runtime, ...)` meaningful also in a multi-threaded program. For a single-threaded program this is the same as process user time.
- Tcl/Tk: The `top_level_events` option to `tk_new/2` is not supported.
- `library(timeout)` is supported. Under Windows 95/98/ME time is measured in real time (walltime), as opposed to process virtual time. As of SICStus 3.10, the time is user time of the main thread under Windows NT/2000/XP.
- `library(sockets)`: The `AF_UNIX` address family is (unsurprisingly) not supported; `socket_select/[5,6]` support only socket streams for arg 4(5).
- `library(system)`: `popen/3` is not supported. `kill/2` attempts to terminate the requested process irrespectively of the 2nd arg. You should not use it as it bypasses the killed process cleanup routines.
- `library(system)`: `shell/[1,2]` does not work reliably except for the simples cases of a single command with no embedded spaces or quote characters. This is because

of limitations of agument passing in Windows. A possible workaround is to create a temporary file and invoke that instead.

4 Tcl/Tk Notes

Tcl/Tk itself is not included in the SICStus distribution. It must be installed in order to use the interface. It can be downloaded from the Tcl/Tk primary website:

http://tcl.sourceforge.net

As of SICStus 3.10.1, SICStus for MacOS X use Aqua Tcl/Tk. The Aqua version of Tcl/Tk uses the native Aqua user interface. Earlier versions of SICStus for MacOS X used a Tcl/Tk based on X Windows. Aqua Tcl/Tk (currently version 8.4.2) can be downloaded, for free, from:

http://www.apple.com/downloads/macosx/unix_open_source/

On HP-UX 1.6i (B.11.22) on Itanium (IPF64), SICStus uses Tcl/Tk 8.4.4.

The Tcl/Tk interface module included in SICStus Prolog 3.11.0 (`library(tcltk)`) is verified to work with Tcl/Tk 8.3. The current version of the interface is expected to work with version 8.1 and newer.

Under UNIX, the installation program automatically detects the Tcl/Tk version (if the user does not specify it explicitly). Except as noted above, the distributed files are compiled for Tcl/Tk 8.3.

Under Windows, the binary distribution is compiled against Tcl/Tk 8.3. If you need to use another version of Tcl/Tk, you have to recompile `library(tcltk)`; see section "Configuring the Tcl/Tk library module under Windows" in *the SICStus Prolog FAQ*.

> **Please note**: You need to have the Tcl/Tk binaries accessible from your `PATH` environment variable, e.g. '`C:\Program Files\Tcl\bin"`'.

The GUI version of SICStus `spwin`, like all Windows non-console applications, lacks the C standard streams (`stdin`,`stdout`,`stderr`) and the Tcl command `puts` and others that use these streams will therefore give errors. The solution is to use `sicstus` instead of `spwin` if the standard streams are required.

4.1 The Tcl/Tk Terminal Window

The Tcl/Tk interface includes a experimental terminal window based on Tcl/Tk. It is opened by using the (undocumented) predicate:

`tk_terminal(Interp, TextWidget, InStream, OutStream, ErrStream)`
: Given a TextWidget, e.g. `.top.myterm`, this predicate opens three Prolog streams for which the text widget acts as a terminal.

There is also a `library(tkconsol)`, making use of `tk_terminal/5`, which switches the Prolog top level to a Tk window. This is done by simply loading the library module.

5 Jasper Notes

5.1 Supported Java Versions

Jasper requires at least Java 2 to run. Except under Windows the full development kit, not just the JRE, is needed. **Jasper does not work with Visual J++ or Visual Café**. Unless indicated otherwise, you can download the JDK from http://java.sun.com.

Except where indicated, Jasper is built with JDK 1.3.1 and is expected to work with the latest version of JDK 1.4.

Jasper is *only* supported under the following configurations:

Solaris 2.x (SPARC and x86)

JDK 1.2

JDK 1.2.2_06 is expected to work, but is no longer tested.

JDK 1.3.1 JDK 1.3.1 is supported; see the Linux entry below. See Section 5.5 [Known Bugs and Limitations], page 20, for when JDK 1.2 or JDK 1.4 is preferred over JDK 1.3.

Note that JDK 1.4 is required for 64 bit SPARC Solaris.

There is a bug in Sun JDK 1.3.1 that surfaces when `sicstus` is the top-level executable, e.g., '`sicstus -l library/jasper/examples/jqueens.pl`'. The symptom similar to:

```
java.lang.UnsatisfiedLinkError: exception occurred in JNI_OnLoad
        ...
at java.awt.Component.<clinit>(Component.java:356)
        ...
```

The underlying problem is that '`motif21/libmawt.so`' cannot find '`libmlib_image.so`' A work-around for SPARC is to set `LD_LIBRARY_PATH` to '*`java-install-dir`*`/jre/lib/sparc`' before invoking `sicstus`. A similar work-around should work on x86. This bug is not present in JDK 1.4.1.

JDK 1.4 64 bit SPARC Solaris is built and tested with JDK 1.4.0.

Linux (x86)

JDK 1.2 Blackdown's JDK (Version 1.2.2 FCS for Linux) is expected to work but is no longer tested. Downloadable from http://www.blackdown.org/java-linux.html.

Sun's JDK 1.2.2 does not support native threads and therefore does *not* work.

JDK 1.3.1

JDK 1.3.1 uses signals in a way that are incompatible with the way signals are used by the SICStus development system (`sicstus`).

Note that this is a problem only with development systems. SICStus run-time systems do not use signals, (e.g. when embedding SICStus in Java using the Jasper package.)

Most of the signal handlers used by the SICStus development system (`sicstus`) are now turned off automatically before `library(jasper)` starts Java. In JDK 1.3.1 the problem with conflicting uses of signals was recognized and the Java initialization option '`-Xrs`' was added to reduce Java's use of signals. It makes JDK use signals in a way that is compatible with the SICStus development system.

There are several ways to pass this flag to Java. The recommended way is to pass it with `jasper_initialize`:

```
bash> sicstus
...
| ?- use_module(library(jasper)),
     jasper_initialize(['-Xrs', <other op-
tions here>], JVM).
```

Alternatively, you can pass it using the (not documented in the JDK documentation) environment variable `_JAVA_OPTIONS`:

```
bash> export _JAVA_OPTIONS='-Xrs'
bash> sicstus
```

Using `_JAVA_OPTIONS` is currently the only way to pass this flag if you rely on the automatic Java initialization done when invoking a Java foreign resource.

JDK 1.4.1 JDK 1.4.1 is expected to work but is not tested.

MacOS X

MacOS X 10.2 comes with JDK 1.3.1 pre-installed. Also see the Linux entry above.

JDK 1.4.1 is now available for MacOS X. It does not work well when SICStus is the top-level application, e.g., when using the SICStus development system. In particular, opening GUI components such as Swing, will hang. The cause for this is not known. One work-around is to force the use of JDK 1.3.1 by setting the environment variable `DYLD_LIBRARY_PATH` before invoking SICStus.

```
bash> ex-
port DYLD_LIBRARY_PATH=/System/Library/Frameworks/JavaVM.framework/Versions/
bash> sicstus -i
```

MacOS X Using Jasper from Java may require that `DYLD_LIBRARY_PATH` be set up so that Java can find the SICStus run-time library. That is, you may need to set `DYLD_LIBRARY_PATH` to the location of the SICStus run-time `libsprt311.dylib`.

Tru64 V5.1

Only the 64bit version (`java -fast64`) of JDK 1.3.1 is supported. Also see the Linux entry above.

Tru64 V5.1

When using Java to call SICStus, you will need to set `LD_LIBRARY_PATH` to the directory where the SICStus run-time library is located (e.g. '`/usr/local/lib`').

Windows Verified using Sun's JDK 1.3.1.

Sun's JDK 1.2.2 and JDK 1.4.x are also expected to work, but are not tested.

AIX JDK 1.3.1 is supported.

The AIX version of JDK 1.3.1 requires some environment variables to be set before invoking an application that embeds the Java VM. For this reason, the following environment variables should be set before starting a SICStus executable that uses `library(jasper)`:

```
bash$ export AIXTHREAD_SCOPE=S
bash$ export AIXTHREAD_MUTEX_DEBUG=OFF
bash$ export AIXTHREAD_RWLOCK_DEBUG=OFF
bash$ export AIXTHREAD_COND_DEBUG=OFF
bash$ export LDR_CNTRL=USERREGS
bash$ sicstus
...
```

See the AIX JDK 1.3.1 readme ('`/usr/java131/README.HTML`') for futher details.

IRIX JDK 1.3.1 is supported.

HP-UX PA32

Java (JDK 1.3.1) is not currently supported. Please contact SICStus support if you need Jasper for this platform.

HP-UX 1.6i (B.11.22) on Itanium (IPF64)

JDK 1.4.1 is supported.

You will need to set `LD_PRELOAD` to the location of `libjvm.so` before invoking `sicstus` in order to use `library(jasper)`. For instance, to start 64bit sicstus with JDK 1.4.1

```
bash$ LD_PRELOAD=/opt/java1.4/jre/lib/IA64W/server/libjvm.so sic-
stus -i
```

This is not an issue when Java is the top-level application.

5.2 Getting Started

This section describes some tips and hints on how to get the interface started. This is actually where most problems occur.

5.2.1 Windows

Under Windows, you should add SICStus Prolog's and Java's DLL directories to your `%PATH%`. This will enable

Windows library search method to locate all relevant DLLs. For SICStus, this is the same as where '`sicstus.exe`' is located, usually `C:\Program Files\SICStus Prolog 3.11.0\bin`. For Java, it is usually '`C:\jdk1.3.1\jre\bin\hotspot`' (for JDK 1.2.2 it would be '`C:\jdk1.2.2\jre\bin\classic`').

For example (Windows NT/2000/XP):

```
C:\> set PATH=C:\jdk1.3.1\jre\bin\hotspot;%PATH%
C:\> set PATH=C:\Program Files\SICStus Prolog 3.11.0\bin;%PATH%
```

To make this change permanent under Windows NT, Windows 2000 or Windows XP, you would use the '`Advanced`' tab in the '`System`' Control Panel. Under Windows 95, 98 and ME you would change the file '`autoexec.bat`'. Consult your OS documentation for details.

5.2.2 UNIX

When `library(jasper)` is used to embed Java in a SICStus development system or run-time system, the run-time linker needs to be told where to find the Java libraries (e.g. '`libjvm.so`'). During installation, '`InstallSICStus`' will build either the `sicstus` executable or the `jasper` foreign resource so that it contains the necessary information; the details are platform dependent.

If you use `spld` to relink SICStus or to build a run-time system, you can use the command line option '`--resource=-jasper`' (note the minus sign). This tells `spld` to include the search path (*rpath*) in the executable needed to ensure that `library(jasper)` can find the Java libraries.

If you want to run `sicstus` with another Java than what was specified during installation, you can use `spld` without the '`--resources`' option to get a SICStus executable without any embedded Java paths. In this case, you need to set the environment variable `LD_LIBRARY_PATH` (or similar) appropriately. One example of this is to use the JDK 1.3 server version instead of the default (client) version.

Alternatively, you can use `spld` with the '`--resource=-jasper`' and '`--with-jdk=`*DIR*' options to generate a development system with embedded paths to another Java directory tree. This will only work if the alternative directory tree has the same structure as the JDK directory seen by '`InstallSICStus`'.

5.2.3 Running Java from SICStus

If SICStus is used as parent application, things are usually really simple. Just execute the query

```
| ?- use_module(library(jasper)).
```

After that, it is possible to perform meta-calls as described in section "Jasper Library Predicates" in *the SICStus Prolog Manual*.

When Jasper is used in run-time systems, additional constraints apply as described in section "Runtime Systems on Target Machines" in *the SICStus Prolog Manual*. The Java to SICStus interface relies on dynamically loading the SICStus run-time system. For this reason, it is not possible to use `library(jasper)` from an executable that links statically with the SICStus run-time.

5.2.4 Running SICStus from Java

If Java is used as parent application, things are a little more complicated. There are a couple of things that need to be taken care of. The first is to specify the correct class path so that Java can find the Jasper classes (`SICStus`, `SPTerm`, and so on). This is done by specifying the pathname of the file '`jasper.jar`':

```
% java -classpath $SP_PATH/bin/jasper.jar ...
```

`SP_PATH` does not need to be set; it is only used here as a placeholder. See the documentation of the Java implementation for more info on how to set classpaths.

The second is to specify where Java should find the Jasper native library ('`libspnative.so`' or '`spnative.dll`'), which the `SICStus` class loads into the JVM by invoking the method `System.loadLibrary("spnative")`. The `loadLibrary` method uses a platform dependent search method to locate the Jasper native library, and quite often this method fails. A typical example of such a failure looks like:

```
% java -classpath [...]/jasper.jar se.sics.jasper.SICStus
Trying to load SICStus.
Exception in thread "main" java.lang.UnsatisfiedLinkError: no spnative
in java.library.path
at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1133)
at java.lang.Runtime.loadLibrary0(Runtime.java:470)
at java.lang.System.loadLibrary(System.java:745)
at se.sics.jasper.SICStus.loadNativeCode(SICStus.java:37)
at se.sics.jasper.SICStus.initSICStus(SICStus.java:80)
at se.sics.jasper.SICStus.<init>(SICStus.java:111)
at se.sics.jasper.SICStus.main(SICStus.java:25)
```

Under UNIX, this can be fixed by explicitly setting the Java property `java.library.path` to the location of '`libspnative.so`', like this:

```
% java -Djava.library.path=/usr/local/lib [...]
```

Under Windows, Java must be able to find '`spnative.dll`' through the `PATH` environment variable (see Section 5.2.1 [Windows], page 15). Setting '`-Djava.library.path`' under Windows can lead to problems if multiple versions of SICStus has been installed.

If this works properly, SICStus should have been loaded into the JVM address space. The only thing left is to tell SICStus where the (extended) runtime library, '`sprt.sav`' ('`spre.sav`'), is located. On those platforms where the SICStus run-time system can determine its own location, e.g. Windows, Solaris and Linux, the run-time system will find the runtime library automatically. Otherwise, you may choose to specify this explicitly by either giving a second argument when initializing the `SICStus` object or by specifying the property `sicstus.path`:

Example (UNIX):

```
% java -Dsicstus.path=/usr/local/lib/sicstus-3.11.0
```

If you do not specify any explicit path, SICStus will search for the runtime library itself.

If everything is set up correctly, you should be able to call `main` (which contains a short piece of test-code) in the SICStus root class, something like this:

```
% java -Djava.library.path="/usr/local/lib" \
       -Dsicstus.path="/usr/local/lib/sicstus-3.11.0" \
       -classpath "/usr/local/lib/sicstus-3.11.0/bin/jasper.jar" \
        se.sics.jasper.SICStus
Trying to load SICStus.
If you see this message, you have successfully
initialized the SICStus Prolog engine.
```

Under Windows, it would look something like this, depending on the shell used:

```
% java -classpath "C:/Program Files/SICStus Prolog
3.11.0/bin/jasper.jar" se.sics.jasper.SICStus
Trying to load SICStus.
If you see this message, you have successfully
initialized the SICStus Prolog engine.
```

If more than one `se.sics.jasper.SICStus` instance will be created, then the SICStus runtimes named e.g. '`sprt311_instance_01_.dll`', need to be available as well. See section "Runtime Systems on Target Machines" in *the SICStus Prolog Manual*.

5.3 Jasper Package Options

The following Java system properties can be set to control some features of the Jasper package:

`se.sics.jasper.SICStus.checkSPTermAge`

This flag is unsupported.

A boolean, *true* by default. If `true`, then run-time checks are performed that attempt to detect potentially dangerous use of the `SPTerm.putXXX` family of functions. The value of this flag can be set and read with

`SICStus.setShouldCheckAge()` and `SICStus.shouldCheckAge()`. This flag was *false* by default in SICStus 3.8.

The run-time checks throws an `IllegalTermException` when there is risk that a `SPTerm` is set to point to a Prolog term *strictly newer* than the `SPTerm`. In this context *strictly newer* means that there exists an open query that was opened after the `SPTerm` object was created but before the Prolog term. See section "SPTerm and Memory" in *the SICStus Prolog Manual*, for more information.

```
% java -Dse.sics.jasper.SICStus.checkSPTermAge=true ...
```

or, from Prolog:

```
jasper_initialize(
   ['-Dse.sics.jasper.SICStus.checkSPTermAge=true'],
   JVM)
```

`se.sics.jasper.SICStus.reuseTermRefs`

This flag is unsupported.

A boolean, *on* by default. If `false`, then `SPTerm.delete()` will only invalidate the `SPTerm` object, it will *not* make the Prolog side SP_term_ref available for re-use. The value of this flag can be set and read with `SICStus.setReuseTermRefs()` and `SICStus.reuseTermRefs()`. There should be no reason to turn it *off*.

To set this flag do:

```
% java -Dse.sics.jasper.SICStus.reuseTermRefs=true ...
```

or, from Prolog:

```
jasper_initialize(
   ['-Dse.sics.jasper.SICStus.reuseTermRefs=true'],
   JVM)
```

`se.sics.jasper.SICStus.debugLevel`

This flag is unsupported.

You probably should not use it in production code. It may be removed or change meaning in future releases.

An integer, zero by default. If larger than zero, then some debug info is output to `System.out`. Larger values produce more info. The value of this flag can be set and read with `SICStus.setDebugLevel()` and `SICStus.debugLevel()`.

```
% java -Dse.sics.jasper.SICStus.debugLevel=1 ...
```

or, from Prolog:

```
jasper_initialize(
   ['-Dse.sics.jasper.SICStus.debugLevel=1'],
   JVM)
```

5.4 Multi Threading

Some exceptions thrown in multi threaded mode may be removed in the future. The user should never catch specific exceptions, but instead catch instances of `PrologException`.

See Section 5.5 [Known Bugs and Limitations], page 20, for details on the limitations of multi threaded Jasper.

5.5 Known Bugs and Limitations

- Jasper cannot be used from within applets, since Jasper relies on calling methods declared as `native`. This is due to a security-restriction enforced on applets by Java; they are not allowed to call native code.
- Only native threads are supported (as opposed to '`green threads`'). This is the default under Windows. Under UNIX, most JDKs use native threads per default in version 1.2.

  On some platforms, you need to explicitly specify the '`-native`' option when calling `java`. The following error is an example of what may happen if you do not specify '`-native`':

  ```
  % java -classpath .:[...]/lib/sicstus-3.11.0/bin/jasper.jar \
       -Djava.library.path=[...]/lib \
       -Dsicstus.path=[...]./lib/sicstus-3.11.0 Simple

  *** panic: libthread loaded into green threads
  Abort (core dumped)
  ```

  Instead, do

  ```
  % java -native [...]
  ```

  this should not be needed with JDK 1.3 or newer.

  See your JDK documentation for more info on command-line parameters to the JVM.
- Some uses of `SPTerm` will leak memory on the Prolog side. This is not really a bug but may come as a surprise to the unwary. See section "SPTerm and Memory" in *the SICStus Prolog Manual*.
- On some combinations of C-compilers and JDKs (specifically GCC with Sun's JDK), the following warning may occur:

  ```
  % splfr simple.pl
  SICStus 3.8.7 (sparc-solaris-5.7): Mon Feb 21 10:43:17 MET 2000
  Licensed to SICS
  {spk.ai82.c generated, 20 msec}

  In file included from /usr/local/jdk1.2/include/jni.h:35,
                   from spk.ai82.c:94:
  /usr/local/jdk1.2/include/solaris/jni_md.h:20: warning: \
     ignoring pragma: "@(#)jni_md.h      1.11    99/02/01 SMI
  ```

  The warning can be safely ignored. You can suppress the warnings when using `gcc` by passing the options '`--cflag=-Wno-unknown-pragmas`' to `splfr`.
- Loading multiple SICStus runtimes has not been very well tested with multi threaded Jasper.

5.6 Java Examples Directory

There is an examples directory available in `$SP_PATH/library/jasper/examples`. See the file `README` for more info.

5.7 Resources

There are almost infinitely many Java resources on the Internet. Here is a list of a few related to Jasper and JNI.

- JavaSoft Homepage (http://java.sun.com/).
- JavaSoft's Java FAQ (http://java.sun.com/products/jdk/faq.html).
- JavaSoft Documentation Homepage (http://java.sun.com/docs/index.html).
- JNI Documentation (http://java.sun.com/products/jdk/1.3/docs/guide/jni/index.html).
- The ACM student magazine *Crossroads* has published an article on the JNI (http://www.acm.org/crossroads/xrds4-2/jni.html). This article may be out of date.

6 Visual Basic Notes

The Visual Basic - SICStus Prolog interface consists of the following files:

- '`vbsp.dll`' (installed as '`SICStus Prolog 3.11.0\bin\vbsp.dll`')
- '`vbsp.bas`' (installed as '`SICStus Prolog 3.11.0\library\vbsp.bas`')

In order to use the interface, perform the following steps:

- Include the file '`vbsp.bas`' in your Visual Basic project.
- During development, the easiest way is to put e.g. '`C:\Program Files\SICStus Prolog 3.11.0\bin`' in the `PATH` environment variable (see Section 5.2.1 [Windows], page 15). Under Windows 2000 and later, `PATH` is set in the control panel '`System`'. Under Windows 9x/ME, this is set in '`autoexec.bat`'. Consult your OS documentation for details.

  Note that when running your Visual Basic project in the Visual Basic debugger, the directory of the current application is the directory that contains the Visual Basic debugger and not the directory that contains your Visual Basic project.

  In a target system, you need to make the SICStus runtime DLL etc. available; see section "Runtime Systems on Windows Target Machines" in *the SICStus Prolog Manual*. In this case, '`vbsp.dll`' should be located at the same place as the SICStus run-time ('`sprt311.dll`').

  If VB cannot find the SICStus run-time files, it will report something similar to

  ```
  File not found: VBSP
  ```

  Please note: Never move or copy SICStus-related files to the Windows system folder.

7 Berkeley DB Notes

As of SICStus 3.8, the library module `library(db)` has been replaced by `library(bdb)`. The functionality is similar, but `library(bdb)` is built on top of Berkeley DB. Berkeley DB can be downloaded from:

http://www.sleepycat.com

`library(bdb)` has been verified to work using Berkeley DB version 4.1.24. It does not work with earlier versions.

When using Berkeley DB under Windows, you should set the `PATH` environment variable to contain the path to '`libdb41.dll`' (see Section 5.2.1 [Windows], page 15). Consult the Berkeley DB documentation for further info.

8 The Emacs Interface

The Emacs Interface was originally developed for GNU Emacs 19.34 and is presently being maintained using XEmacs 21.1 and tested with GNU Emacs 19.34.1. For best performance and compatibility and to enable all features we recommend that the latest versions of GNU Emacs or XEmacs be used. For information on obtaining GNU Emacs or XEmacs; see http://www.gnu.org and http://www.xemacs.org, respectively.

8.1 Installation

Starting with SICStus 3.8, the Emacs interface is distributed with SICStus and installed by default. The default installation location for the emacs files is '`<prefix>/lib/sicstus-3.11.0/emacs/`' on UNIX platforms and '`C:\Program Files\SICStus Prolog 3.11.0\emacs\`' under Windows.

For maximum performance the Emacs Lisp files (extension .el) should be compiled. This, completely optional step, can be done from within Emacs with the command *M-x byte-compile-file*. See section "Installation" in *the SICStus Prolog Manual*, for further details.

The easiest way to configure the Emacs interface is to load the file '`sicstus_emacs_init.el`' from your '`.emacs`' file. It will find the SICStus executable and do all initialization needed to use the SICStus Emacs interface.

8.1.1 Installing On-Line Documentation

It is possible to look up the documentation for any built in or library predicate from within Emacs (using *C-c ?* or the menu). For this to work Emacs must be told about the location of the '`info`'-files that make up the documentation.

If you load the file '`sicstus_emacs_init.el`' from your '`.emacs`' file then Emacs should be able to find the SICStus documentation automatically; see section "Installation" in *the SICStus Prolog Manual*, for further details.

9 Revision history

This chapter summarizes the changes in release 3 wrt. previous SICStus Prolog releases as well as changes introduced by patch releases.

9.1 Changes in Release 3

- Backslashes ('\') in strings, quoted atoms, and integers written in '`0''` notation denote escape sequences. Character escaping can be switched off.
- Multifile declarations are required in *all* files where clauses to a multifile predicate are defined. This complies with the ISO Prolog Standard.
- The built-in predicate `call_residue/2` has been modified so that goals that are disjunctively blocked on several variables are returned correctly in the second argument.
- The built-in predicate `setarg/3` has been removed. Its functionality is provided by the new built-in predicates `create_mutable/2`, `get_mutable/2`, `update_mutable/2`, and `is_mutable/2`, which implement a timestamp technique for value-trailing with low-level support.
- The built-in predicates `unix/1` and `plsys/1` have been removed. Their functionality is provided by `prolog_flag(argv,X)`, by the new `halt/1` built-in predicate, and by the new `library(system)` module, which also contains several new predicates.
- The socket I/O built-in predicates have been moved to the new `library(sockets)` module.
- The built-in predicate `time_out/3` has been moved to the new `library(timeout)` module.
- The built-in predicates `term_hash/[2,4]`, `subsumes_chk/2`, and `term_subsumer/3` have been moved to the new `library(terms)` module, which also contains operations for unification with occurs-check, testing acyclicity, and getting the variables of a term.
- The foreign language interface (Prolog-to-C) has been extended with the types `+chars`, `-chars` and `[-chars]` for fast conversion between C strings and Prolog code-lists. Several new interface functions are available.
- The memory handling of the C-to-Prolog interface has been simplified by passing each Prolog term as a "handle" object, called an *SP_term_ref*, making the functions `SP_show_term()` and `SP_hide_term()` obsolete.
- The InterViews 2.6 based GUI module `library(gmlib)` has been replaced by the Tcl/Tk
  based `library(tcltk)`. A version of `library(gmlib)` converted to SICStus Prolog release 3 is available from '`ftp://ftp.sics.se/archive/sicstus3/gmlib.tar.gz`'.
- The `library(objects)` module has been enhanced.
  - Inheritance is static, i.e. determined at object creation time, and is implemented as module importation.
  - A new, very light-weight, type of object: *instance*.
  - *Attributes*, efficient storage of terms in objects.

  - Unprefixed goals in methods denote message passing to `self`. Prolog goals in methods must be prefixed by '`:`'.
- In `library(charsio)`, the `open_chars_stream/[3,4]` predicates have been replaced by `open_chars_stream/2` and `with_output_to_chars/[2,3]`.
- The `library(assoc)` module now implements AVL trees instead of unbalanced binary trees.
- The new `library(atts)` implements attributed variables, a general mechanism for associating logical variables with arbitrary attributes. Comes with a number of hooks that make it convenient to define and interface to constraint solvers.
- The Boolean constraint solver has been moved to the new `library(clpb)` and is implemented on top of `library(atts)`.
- New constraint solvers for rationals (`library(clpq)`) and reals (`library(clpr)`), implemented on top of `library(atts)`.
- `user:goal_expansion/3` is a new hook predicate for macro-expansion.
- `bb_put/2`, `bb_get/2`, `bb_delete/2`, and `bb_update/3` are new built-in predicates implementing blackboard primitives.
- `prolog_load_context/2` is a new built-in predicate for accessing aspects of the context of files being loaded.
- `user:file_search_path/2` is a new hook predicate providing an path expansion mechanism for filenames.
- `gcd/2` is a new built-in function.
- The statistics keyword `walltime` measures elapsed absolute time.
- In runtime systems, `ensure_loaded/1` and `use_module/[1,2,3]` have the same semantics as in development systems.
- Native code compilation available for MIPS platforms.
- Problems in native code compilation for certain SPARC models have been eliminated.
- Performance improvements include emulated code speed, native code speed, and the foreign language interface.
- The system has been ported to the DEC OSF/1 Alpha (a 64-bit platform).

9.2 Changes Introduced in 3#4

- New built-in predicates and command line tools for creating and loading foreign language modules and creating customized development and runtime systems.
- Slight changes in the C interface: hook variables are set by function calls, `SP_foreign_reinit_hook()` is not supported.
- The system has been ported to the Microsoft Win32 platform (Intel x86).
- The system has been ported to the Macintosh.
- The system has been ported to the OS/2 (32bit) platform (Intel x86).
- If the init file '`~/.sicstusrc`' is not found, SICStus looks for '`~/sicstus.ini`'.
- `library(sockets): socket_select/5` arg 1 may be a, possibly empty, list of passive sockets, arg 3 returns a, possibly empty, list of new streams.

- `library(system)`: The following new predicates are provided: `tmpnam/1`, `directory_files/2`, `file_property/2`, `delete_file/2`, `make_directory/1`.
- A new constraint solver for finite domains (`library(clpfd)`), implemented on top of `library(atts)`.

9.3 Changes Introduced in 3#5

- New built-in predicate `open/4`, enables opening files in binary mode.
- `library(charsio)`: New predicate `with_output_to_chars/4`.
- `library(heaps)`: New predicates `delete_from_heap/4`, `empty_heap/1`, `is_heap/1`.
- `library(queues)`: New predicate `is_queue/1`.
- `library(sockets)`: New predicates: `socket_accept/3`, and `socket_select/6` provide address of connecting client. `hostname_address/2` resolves name/ip-number.
- `SP_atom_length()` returns the print name length of a Prolog atom.
- Modification time instead of current time stored for loaded files.

9.4 Changes Introduced in 3#6

- `toplevel_print_options` and `debugger_print_options` are new Prolog flags controlling the toplevel's and debugger's printing behavior.
- `is_mutable/1` is a new built-in predicate, which is true for mutables.
- '`~@`' is a new spec in `format/[2,3]` for arbitrary goals.
- Mutables are initialized correctly when read in.
- The finite domain constraint solver (`library(clpfd)`) has been enhanced by a programming interface for global constraints, improved compilation to library constraints and other performance enhancements, and by a number of new exported constraints.
- `library(objects)`: New hook predicate `user:method_expansion/3`.
- `library(sockets)`: `socket_select/5` has extended functionality.
- Efficiency bugs in `format/[2,3]` fixed.
- Bug in `save_program/[1,2]` with native code fixed.
- Bugs in `library(chr)` fixed, and a couple of new constraint handlers fixed.
- A problem with source-linked debugging of grammar rules fixed.
- Prevent looping on duplicates in `module/2` decl.
- Prevent memory overrun in `library(tcltk)`.

9.5 Changes Introduced in Version 3.7

- The concept of patchlevels removed and replaced by versions.
- `library(chr)`: A new library module providing Constraint Handling Rules; see
  http://www.pst.informatik.uni-muenchen.de/~fruehwir/chr-solver.html

- *Jasper*, a bi-directional Java-interface, consisting of extensions to the existing FLI and a new library module `library(jasper)`.
- Atom garbage collection, invoked by `garbage_collect_atoms/0`, and controlled by the `agc_margin` Prolog flag. New statistics options: `atoms`, `atom_garbage_collection`. New interface functions: `SP_register_atom()`, `SP_unregister_atom()`.
- Calls with clean-up guaranteed, provided by `call_cleanup/2`, which replaces `undo/1`.
- Source-linked debugging, controlled by the `source_info` Prolog flag.
- Debugger enhancements: tracing of compiled code; a new debugger mode `zip` and built-in predicates `zip/0`, `nozip/0`; new debugger commands *out n*, *skip i*, *quasi-skip i*, *zip*, *backtrace n*, *raise exception*. Modules can be declared as *hidden*, which disables tracing of their predicates.
- Saved-states are available in runtime systems, and are portable across platforms and between development and runtime systems. `save/[1,2]` are gone. In most cases, `save_program/2` can be used in their place, with a little rearrangement of your code. Predicates can be declared as *volatile*.
- A interface function `SP_restore()` is the C equivalent of `restore/1`, which now only restores the program state, leaving the Prolog execution stacks unchanged.
- The GNU Emacs interface was enhanced: source-linked debugging, new menus, speed, help functions, electric functions, indentation, portability, bug fixes.
- The reader can return layout information about terms read in. New `read_term/3` option: `layout(-Layout)`. New hook predicate: `user:term_expansion/4`.
- Module name expansion of goals is done prior to execution of meta-calls.
- Imported predicates can be spied and abolished.
- `random:randset/3` returns a set in standard order.
- `db:db_canonical/[2,3]` are new; can be used to check whether two *TermRefs* refer to the same term.
- `clpfd:serialized_precedence/3` and `clpfd:serialized_precedence_resource/4` are new; model non-overlapping tasks with precedence constraints or sequence-dependent setup times.
- In object method bodies, goals of the form `:Goal` are translated according to the manual. Earlier versions treated arguments occurring in the '`:`' position of meta-predicates specially.
- A new interface function `SP_raise_fault()` and interface macro `SP_on_fault()` are available for handling runtime faults that cannot be caught as exceptions.
- A new interface function `SP_set_memalloc_hooks()` is available for redefining the memory manager's bottom layer. Related to that, there is a new command-line option '`-m`'.
- Development and runtime systems have been reorganized internally. All use a runtime kernel shared object or DLL, and are initialized by restoring saved-states. Development systems additionally use a development kernel shared object or DLL.
- The '`-B`' command-line option is gone in the start-up script, and some new options have appeared.
- Under UNIX: new option '`-base`' to override the executable used by the start-script.

- Under UNIX: improvements in the configure-script; better options to specify Tcl/Tk versions and paths.
- Hookable standard streams.
- Floating-point operations on Digital Alpha are now IEEE-conformant.
- `reinitialise/0` does not load any initialization files given in '`-i`' or '`-l`' command line flags.
- Under UNIX: New option '`-S`' to `spmkrs` and `spmkds` to link the SICStus Runtime Kernel (and development extensions for `spmkds`) statically into the executable.
- `[File1,File2,...]` was broken.
- `require/1` did not find all directories.
- Runtime systems could crash after GC.
- Bugs in `clp[qr]:dump/3`, `clp[qr]:expand/0`, `clp[qr]:noexpand/0`.
- The garbage collector reported too many bytes collected.
- Memory overflows were not handled gracefully.
- Imported predicates couldn't be abolished.
- `arrays:arefa/3`, `arrays:arefl/3`, `heaps:min_of_heap/5` are now steadfast.
- Most `library(clpfd)` predicates now check the type of their arguments. Bugs fixed in `relation/3`, `serialized/2`, `all_distinct/1`.
- `frozen/2` could crash on an argument of the wrong type.
- `SP_get_list_n_chars()` does not require a proper list.
- Problems with exceptions in embedded commands in source files.
- Problems with `load_files(Files, [compilation_mode(assert_all)])`.
- For `load_files(Files, [if(changed)])`, a non-module-file is not considered to have been previously loaded if it was loaded into a different module.
- Incorrect translation of `if/3` goals in grammar rules.
- On Win32, `system:mktemp/2` sometimes returned filenames with backslashes in them.

9.6 Changes Introduced in Version 3.7.1

- The type-specifier `object` in Jasper has changed to `object(Class)`.
- Under UNIX: Error-handling in `splfr`, `spmkrs`, `spmkds`.
- Jasper did not convert return values correctly when calling Java from Prolog.
- Jasper did not handle instance methods correctly.
- Some of the legal type-specifiers in Jasper were rejected by the glue code generator.
- Efficiency bugs in `format/[2,3]` fixed.
- Bug in `save_program/[1,2]` with native code fixed.
- Bugs in `library(chr)` fixed, and a couple of new constraint handlers fixed.
- A problem with source-linked debugging of grammar rules fixed.
- Prevent looping on duplicates in `module/2` declaration.
- Prevent memory overrun in `library(tcltk)`.

9.7 Changes Introduced in Version 3.8

9.7.1 Wide Character Support

Wide character handling is introduced, with the following highlights:

- character code sets up to 31 bit wide;
- three built-in wide character modes (ISO_8859_1, UTF8, EUC), selectable via environment flags;
- complete control over the external encoding via hook functions.

For programs using the default ISO_8859_1 character set, the introduction of wide characters is transparent, except for the string format change in the foreign language interface; see below.

In programs using the EUC character set, the multibyte EUC characters are now input as a single, up to 23 bit wide, character code. This character code can be easily decomposed into its constituent bytes, if needed. The encoding function is described in detail in the SICStus manual.

To support wide characters, the foreign language interface now uses UTF-8 encoding for strings containing non-ASCII characters (codes >= 128). This affects programs with strings that contain e.g. accented characters and that transfer such strings between Prolog and C. If such a string is created on the C side, it should be converted to UTF-8, before passing it to Prolog. Similarly for a string passed from Prolog to C, if it is to be decomposed into characters on the C side, the inverse transformation has to be applied.

Utility functions `SP_code_wci()` and `SP_wci_code()` are provided to support the conversion of strings between the WCI (Wide Character Internal encoding, i.e. UTF-8) format and wide character codes.

9.7.2 Breakpointing Debugger

A new general debugger is introduced, with advanced debugging features and an advice facility. It generalizes the notion of spypoint to that of the breakpoint. Breakpoints make it possible to e.g. stop the program at a specified line, or in a specified line range, or to call arbitrary Prolog goals at specified ports, etc. Highlights:

- Advice facility — useful for non-interactive debugging, such as checking of program invariants, collecting information, profiling, etc.
- Debugger hook predicate — new interactive tracer commands can be defined.
- Tracer information access — data on current and past execution states, such as those contained in the ancestor list, or the backtrace, is now accessible to the program.

- The following built-in predicates have been added: `add_breakpoint/2`, `spy/2`, `current_breakpoint/4`, `remove_breakpoints/1`, `disable_breakpoints/1`, `enable_breakpoints/1`, and `execution_state/[1,2]`. `user:debugger_command_hook/2` is a new hook predicate.

  The predicates `nospy/1` and `nospyall/0` have slighty changed meaning.

  The predicate `spypoint_condition/3` has been removed.

9.7.3 ISO Compliance

SICStus 3.8 supports standard Prolog, adhering to the International Standard ISO/IEC 13211-1 (PROLOG: Part 1—General Core). At the same time it also supports programs written in earlier versions of SICStus. This is achieved by introducing two execution modes `iso` and `sicstus`. Users can change between the modes using the Prolog flag `language`. Main issues:

- The `sicstus` execution mode is practically identical to 3.7.1, except for minor changes in error term format.
- The `iso` mode is fully compliant with ISO standard, but no strict conformance mode is provided.
- The dual mode system supports the gradual transition from legacy SICStus code to ISO Prolog compliant programs.
- Note that the built-in predicates, functions and Prolog flags, required by the ISO standard, are also available in `sicstus` execution mode, unless they conflict with existing SICStus predicates or functions. This expansion of the language carries a remote risk of name clashes with user code.

9.7.4 Generic New Features

- The `spmkds` and `spmkrs` tools for creating stand-alone executables have been replaced by a common `spld` tool, which takes several new options. Runtime systems do not always need a main program in C. Under Windows, the resulting executable can optionally be windowed. The `splfr` tool takes several new options. The development and runtime kernels have been merged into a single one.
- Partial saved-states corresponding to a set of source files, modules, and predicates can be created by the new built-in predicates `save_files/2`, `save_modules/2`, and `save_predicates/2` respectively. These predicates create files in a binary format, by default with the prefix '`.po`' (for Prolog object file), which can be loaded by `load_files/[1,2]`. The `load_type(Type)` option of `load_files/2` has been extended. Partial saved-states render '`.ql`' files obsolescent.
- The new built-in predicate `trimcore/0` reclaims any dead clauses and predicates, defragmentizes Prolog's memory, and attempts to return unused memory to the operating system. It is called automatically at every top level query.
- The value of the new read-only Prolog flag `host_type` is an atom identifying the platform, such as `'x86-linux-glibc2.1'`.

- The functionality of the `source_info` Prolog flag, introduced in release 3.7, has been extended beyond the Emacs interface. Line number information is now included in error exceptions whenever possible. This information is displayed in debugging and error messages (outside Emacs) or causes Emacs to highlight the culprit line of code. Valid values are `off`, `on`, and `emacs`.
- Predicate indicators can take the form *Name/[Arity,. . .,Arity]* in `spy/[1,2]`, `nospy/1`, `listing/1`, `abolish/1`, `profile_data/4`, `profile_reset/1`, `save_predicates/2`, and `gauge:view/1`.
- The new interface functions `SP_chdir()` and `SP_getcwd()` provide access to the current working directory.
- The interface function `SP_load()` has been generalized to correspond to `load_files/1`.
- The interface function `SP_deinitialize()` is now documented.
- Windows: the registry is no longer used by SICStus itself. The SICStus Runtime Library is located based on the location of '`sprt<xx>.dll`'. `SP_PATH` is only used as a last resort. See Section 3.3 [Windows Notes], page 7.
- Source code compilation and installation procedure has been improved and simplified. See '`README`' and '`INSTALL`' in the source code distribution for documentation.
- The layout of the Gauge graphical user interface has been improved.
- The new `library(bdb)` provides an interface to the Berkeley DB toolset for persistent storage, and replaces `library(db)`. The programming interface of the new module is similar to that of the old one, with some new concepts added such as *iterators*. The sources of the old library module are available from:

  ftp://ftp.sics.se/archive/sicstus3/libdb.tgz
- `library(db)` is obsolete and will be removed in the next major release.
- Generic runtime systems under Windows are built using `spld` and exist in three flavors: generic character based (`sprt`), generic character based interactive (`sprti`), and generic windowed (`sprtw`). See section "Generic Runtime Systems under Windows" in *the SICStus Prolog Manual*.
- The manual chapter for `library(tcltk)` has been rewritten and greatly expanded.
- `library(clpq)` and `library(clpr)`: new predicates `inf/4` and `sup/4`.
- Code fragments loaded via the Emacs interface are imported into the type-in module, unless the source file has an explicit mode line.
- `library(gcla)` has been removed.
- `initialization/[0,1]` have been replaced by ISO compliant initializations.

9.7.5 New Features in library(jasper)

- Java 2 (a.k.a. JDK 1.2) is now required. `library(jasper)` will not work using JDK 1.1.x.
- Support for native threads JDKs.
- Changed package name from `jasper` to `se.sics.jasper`, according to JavaSoft guidelines. See Section 5.2 [Getting Started], page 15.

- Classfiles are now placed in '`jasper.jar`', which is located in `$SP_PATH/bin`. See Section 5.2 [Getting Started], page 15.
- The shared library for Jasper ('`jasper.dll`' or '`libjasper.so`') is now located in the same directory as the runtime kernel (default `<installdir>/lib` under UNIX, `<installdir>/bin` under Windows). See Section 5.2 [Getting Started], page 15.
- Meta-call functionality added (`jasper_call_instance/6`, `jasper_call_static/6`, etc.). This makes it possible to call Java without having to generate any glue code (i.e. without a C-compiler).
- Support for handling local global references from Prolog (`jasper_create_global_ref/3`, `jasper_delete_global_ref/2`, `jasper_delete_local_ref/2`).
- `SPException.term` declared `protected` instead of `private`.
- New class `SPCanonicalAtom` to handle canonical representations of atoms and to make sure that they are safe with atom garbage collection. New methods `getCanonicalAtom` and `putCanonicalAtom`. New constructor for `SPPredicate`. `getAtom` and `putAtom` deprecated.
- New exception: `IllegalCallerException` is thrown if the current thread is not allowed to call SICStus.

9.7.6 New Features in library(clpfd)

- `fd_degree/2` is new; returns the number of constraints attached to a variable.
- `labeling/2` requires the list of domain variables to have bounded domains. User-defined variable and value choice heuristics can be provided.
- `element/3` is interval-consistent in its second and third arguments. Use `relation/3` if domain-consistency is required.
- `serialized/3` is new and replaces `serialized_precedence/3` and `serialized_precedence_resource/4`. A number of new options control the algorithm. The space complexity no longer depends on the domain size.
- `cumulative/5` is new and takes the same options as `serialized/3`.
- `all_different/2`, `all_distinct/2` and `assignment/3` are new and take options controlling the algorithms.
- Generally, performance and error checking have been improved.

9.7.7 Bugs Fixed in Version 3.8

- `absolute_file_name/2`: could crash under IRIX; nested compound terms allowed
- `call_cleanup/2`: efficiency
- `close/1`: efficiency; handling the standard streams
- `format/[2,3]`: '`~N`' didn't work as expected; are now meta-predicates—needed by the '`~@`' format spec
- `load_files/[1,2]`: avoid changing directory; don't loop on duplicate exports
- `load_foreign_resource/1`: filenames containing periods under Windows NT

- `print_message/2`: in runtime systems
- `prolog_load_context/2`: value of `term_position`
- `reinitialise/0`: sequencing of events
- `save_program/[1,2]`: fastcode handling; file mode creation masks; in runtime systems
- `write_term/[1,2]`: the `indented(true)` option and non-ground terms
- `library(db)`: efficiency of term deletion
- `library(heaps)`: `delete_from_heap/4`
- `library(objects)`: the `new/2` method; cyclic dependencies
- `library(random)`: determinacy and efficiency
- `library(sockets)`: noisy startup under Windows; block buffering is now the default; `socket_buffering/4` added
- `library(system)`: `sleep/1` admits floats as well as integers
- `library(terms)`: `subsumes_chk/2` and `variant/2` now don't unblock goals
- glue code generator: incorrect translation of `+chars`; syntax error messages were suppressed
- all system messages go via the `print_message/2` interface
- input argument checking is generally stricter
- resources are unloaded in LIFO order but loaded in FIFO order at save/restore
- CLP(Q,R): answer constraint projection
- problems with bignums and big terms in '`.ql`' files
- detecting invalid goals in meta-calls, asserts, `load_files/[1,2]`
- spurious redefinition warnings
- bignum quotient/remainder on 64-bit architectures
- compiler: complexity of compiling multiple clauses with same key, code generation quality for inline goals
- memory manager: avoiding dangling pointers under Windows, better reclamation of dead clauses and predicates, using dynamic hashing and *hashpjw* for atoms, keeping predicate tables as small as possible, avoiding stack overflow if multiple goals get simultaneously unblocked, better reuse of free memory blocks
- garbage collector: removing redundant trail entries for mutables, improved scope and speed of generational garbage collection
- callbacks to Prolog while reading from the terminal
- printing atoms with character codes in 27. . .31
- reading atoms with '`\c`'
- Floating point NaN (Not a Number). Now behaves consistently across platforms. In particular, fixed Windows related bugs with arithmetic on and printing of NaN.
  - Arithmetic comparisons involving NaN now fails (except `=\=`). Note that `X is nan, X =:= X` fails.
  - Term order for NaN is now defined and the same for all platforms. There is a single NaN and it lies between (the float) `+inf` and the integers.

9.8 Changes Introduced in Version 3.8.1

Version 3.8.1 is a bugfix release only, no new features has been added.

- '`configure.in`': Removed multiple occurences of the '`-n32`' flag under IRIX if `cc` is used instead of `gcc`.
- '`configure.in`': FreeBSD 3.x is now handled correctly.
- '`configure.in`': On Linux and Solaris, SICStus is now always linked with the POSIX thread library.
- `InstallSICStus`: `spld` did not log verbose output to logfile.
- `spld`, `splfr`: Eliminated use of '`..`' to specify relative paths. Caused problems under Windows 95/98.
- `library(jasper)`: Green threads JDKs not supported any longer.
- `library(tcltk)`: `Tcl_FindExecutable("")` is called when the '`tcltk`' library is loaded, before any Tcl/Tk interpreter is created. This should fix errors related to not finding '`init.tcl`' and also improve support for international character sets.
- `multifile` + `discontiguous` combination fix
- redefinition warning for multifile predicates fix
- `listing/[0,1]`, `tell/1`, `see/1` fixes
- avoid bogus line number info for native code
- trail compression fix
- `stack_shifts` (statistics/2 option) manual fix
- `load_foreign_resource/1` search algorithm fix
- atom/number handling fixes
- raise error for `a =.. [b|c]`
- avoid SP_term_ref leaks in some functions
- prevent dangling pointer problem in displaying line number info
- check representability of compiled clauses
- prevent looping at halt and elsewhere if advice has been given
- CHR: initialization fix
- CLPFD: fixes and corrections to `all_distinct/[1,2]`, `assignment/[2,3]`, `circuit/[1,2]`, `serialized/[2,3]`, `cumulative/[4,5]`, `fdset_member/2`, arithmetic
- LINDA: buffering fix

9.9 Changes Introduced in Version 3.8.2

Version 3.8.2 is a bugfix release only, no new features has been added.

- `call_residue/2`: fix bug when the goal called `copy_term/2`.

- `listing/[1,2]`, `portray_clause/[1,2]`, top-level: cope with constrained/attributed variables.
- `portray_clause/[1,2]`, `write_term/[2,3]` with `indented(true)`: do not juggle module prefixes.
- Foreign resources: problems with prelinked resources and with `clpfd`
- Foreign resources: The returned arguments from a foreign function are now properly ignored if an exception was raised with `SP_raise_exception()`.
- Foreign resources: Added some, for now, undocumented callbacks to '`<sicstus/sicstus.h>`'. Documented `SP_to_os()`, `SP_from_os()`.
- Atom garbage collector: don't reclaim undefined predicates that have pointers to them; some atom locations were not traced.
- Local stack shifter bug.
- Source info management: ensure expansion of the compiled file table.
- Backtracking from fastcode to compactcode special case.
- Bytecode relocation bug after restore.
- Compiler bug on very large clauses.
- `SP_WcxOpenHook`: incorrect prototype.
- Emulator kernel: performance bugs.
- 64-bit portability bugs.
- `library(bdb)`: a relative filename given in `db_open/5` was treated by SICStus as relative to the current working directory, but should be relative to the given BDB environment.
- `library(clpfd)`: somewhat faster arithmetic, lingering bugs in `serialized/[2,3]` and `cumulative/[4,5]`, `labeling/2` options `value/1`, `variable/1`
- The configure script did not specify the correct Irix/MIPS ABI/ISA level building with GCC.
- Added '`--with=<package>`' options to `spld` and `splfr` to override default installation path for third-party software packages.
- `spld`: Fixed bugs in argument handling. '`.pl`' file arguments are no longer compiled at `spld` time, but passed directly to `SP_load()`.
- `library(jasper)`: Multiple threads are allowed to call SICStus without `IllegalCallerException` being thrown. See section "Java Threads" in *the SICStus Prolog Manual*.
- `library(jasper)`: Argument-checking bug in `jasper_call_static/6` and `jasper_call_instance/6`.
- Recover properly from memory allocation failures.

9.10 Changes Introduced in Version 3.8.3

Version 3.8.3 is mainly a bugfix release. New features:

- New interface functions `SP_calloc()` and `SP_strdup()`.
- The Windows version is now up to twice as fast (measured on the benchmarks in http://www.sics.se/sicstus/benchmarks.html). In particular, SICStus ought to be as fast under Windows as under Linux given the same hardware. This will only affect '`pure`' Prolog code, built-in predicates such as `assert` are not affected although the Prolog part of libraries are affected. (The change is in the byte code dispatch mechanism).
- The Windows console (`spwin`) can now save a transcript of the interaction with the Prolog top-level. The command is under the '`File`' menu. You may wish to increase the number of '`save lines`' in the '`Windows Settings`' (under the '`Settings`' menu).
- `library(clpfd)`: new constraints `disjoint1/[1,2]`, `disjoint2/[1,2]` model non-overlapping lines and rectangles.

Bug fixes:

- The Windows console: '`All Files`' should now work in file selection dialogs.
- A problem that prevented `spld` and `splfr` from working under Windows 95/98 has been fixed.
- Fixed meta-quoting of regular expressions in `spld` and `splfr`.
- `spld` warns when input files are ignored
- Runtime system executables generated using `spld` return 0 when `user:runtime_entry/1` succeeds and 1 on failure or exception.
- `SP_chdir()` declares its first argument as `const char *`.
- Restore fixes for native code.
- Atom garbage collection during restore fix.
- Listing fix for disjunctions.
- Integer range manual fix.
- Avoid doing initializations twice for '`-l`' and '`-r`' files.
- Compiler fix for `once/1`.
- Buffering fix for Linda.
- Wide character handling bug fixes.
- `prolog_flag/[2,3]`: fix for runtime systems.
- `SP_unify()`: undo any bindings on failure.
- `library(bdb)`: relative filename handling fix.
- `library(clpfd)`: GC interaction, overflow detection, performance fixes.
- A problem where multiple copies of the Jasper library were loaded has been fixed. This affects all platforms. Now there is exactly one version of the Jasper shared library ('`libjasper.so`' or '`jasper.dll`').
- Jasper: `+atom` maps to `SPCanonicalAtom` instead of `SPTerm`.
- Jasper: the `+double` specifier did not work.

9.11 Changes Introduced in Version 3.8.4

Version 3.8.4 is mainly a bugfix release. New features:

- `abort/0` returns to the innermost top-level, and does not switch off the debugger.
- `library(clpfd)`: Given *Term* containing domain variables, `fd_copy_term(`*`Term`*`,`*`Template`*`,`*`Body`*`)` will compute *Template* and *Body* where *Template* is a copy of the same term with all variables renamed to new variables such that executing *Body* will post constraints equivalent to those that *Term* is attached to.
- `library(tcltk))`:
  - Added `list(CommandList)` to the possible command formats. It creates a TCL list by, in effect, calling the TCL command `list` with the result of converting each element of *CommandList*. The result is that Tcl will treat the result as a list with the same length as *CommandList* even if the elements contains spaces or other special characters.

    Current code that uses `ListOfCommands` should probably often be better off using `list(ListOfCommands)`. See the manual for details.
  - Added `writeq(Command)` and `write_canonical(Command)` as legal command specifications. Documented that `write_canonical` is the preferred way of passing Prolog terms from Prolog to Tcl and back.
  - More error checking and reporting. In particular, the output of a Prolog goal must now be in the special command format. It used to just silently generate garbage. **Potential backward compatibility issue.**

    Now the value of variables named `_` are ignored. This makes it possible to avoid errors if some uninteresting result is not in the special command format. This used to be less of a problem since such errors were silently ignored. (**Please note**: in SICStus 3.8.5 this was changed to ignore all variables with names *starting* with underscore '`_`'.)
  - International (UNICODE) character now passed between Tcl/Tk and Prolog. Made the stream used internally by `library(tcltk)` always use UTF8 so that non seven bit characters gets recognized by Tcl. This transfers character codes unchanged between SICStus and Tcl so it assumes that SICStus interprets character codes as UNICODE (as this is what Tcl does).
  - `tk_num_main_windows/2` and `tk_main_window/2` no longer segfaults under Windows if `tk_new/2` has not been called. Added a `"tk_new called"` check to some other routines as well. The segfault occurred when, due to a bug in Tk, Tk uses stubs to access Tcl. Presently Tk uses Tcl stubs by default only under Windows.
  - The empty string resulting from an empty `CommandList` now becomes properly NUL terminated.
  - `prolog_call` now resets the FLI stack to avoid space leaks when Tcl/tk is the master and Prolog the slave.
  - Corrected some bugs in the Tcl/Tk documentation. Added examples of using the new command specifications.

- `library('linda/client')`: New predicate `shutdown_server/0`. The server keeps running after receiving this signal, until such time as all the clients have closed their connections. Courtesy of Malcolm Ryan.
- Some more options are available when the user is asked about redefining predicates.

Bug fixes:

- `skip_line/1`, `at_end_of_file/0`, `tab/2`.
- Asserting, copying or throwing terms with domain variables now raises an exception instead of crashing.
- Non-existent files and the `include/1` directive.
- GC and BDD interaction.
- `save_program/[1,2]`, `save_files/[1,2]`: check for I/O errors; problems with `'$ref'/2` terms; problems with SICStus Objects.
- `unload_foreign_resource/1`: false alarm in prelinked binaries.
- Jasper: glue code sometimes crashed when returning from a Java method that throwed an exception.
- Jasper: glue code reported errors for bogus argument numbers.
- `library(tcltk))`: Bug fixes and enhancements; see the '`New Features`' section above for details.
- `spld`/`splfr` under Windows: Errors are now properly reflected in the exit code from these tools.
- Error handling determining current directory.
- `library(clpfd)`: disequations speeded up, bugs in `disjoint1/[1,2]`, `disjoint2/[1,2]`, `element/3`, propagation, entailment detection, backward compatibility.
- Workaround for crashes when static SICStus executables, i.e. built with `spld --static`, load (non-prelinked) dynamic foreign resources. With this workaround loading a dynamic foreign resource into a static SICStus executable will still, unnecessarily, load the shared version of the SICStus runtime system ('`libsprt311.so`') but the shared runtime system will not be used. This will be fixed in a forthcoming release.

9.12 Changes Introduced in Version 3.8.5

Version 3.8.5 is mainly a bugfix release. New features:

- `copy_term/2` and `call_residue/2` now support finite domain variables.
- Representation errors due to illegal usage of finite domain variables have been replaced by more useful exceptions.
- The new exported predicate `terms:term_variables_bag/2` is like `terms:term_variables/2`, but its output argument is a list of variables in order of first occurrence.
- `bdb:db_open/5` is generalized so that a cache size can be provided.

- `clpfd:fd_neighbors/2` is a new exported predicates. It is the relation that `clpfd:fd_closure/2` is the transitive closure of.
- The Java interface has been improved; see below for new features.

Bug fixes:

- `current_atom/1` now terminates correctly.
- `once/1` is now handled correctly in ISO mode.
- `predicate_property/2` now handles built-in predicates correctly.
- `prolog_flag/2` alias `current_prolog_flag/2` now behave as pure relations in SICStus execution mode.
- `read/[1,2]` now handle character code 0 correctly.
- `save_files/2`, `save_predicates/2`, and `save_modules/2` do not replace given output file extensions. A '`.po`' extension will be added if none is given. Note, however, that `load_files/[1,2]` will only recognize files with a '`.po`' extension as '`.po`' files.
- `statistics(trail,L)` and `statistics(choice,L)` are more accurate.
- `stream_code/2` now handles errors correctly.
- `stream_interrupt/3` raises an existence error under Windows.
- `stream_property/2` now handles `alias/1` property for standard streams correctly.
- `stream_select/3` now returns a valid list of streams, and raises an existence error under Windows.
- The tokenizer now does not read too far ahead on non-float tokens that start like floats.
- The `s` answer to redefinition queries is now handled correctly.
- Worked around a C compiler bug affecting garbage collection under Windows (MS bug Q263609).
- A compiler bug fixed.
- Repeated restoring caused a memory leak, now sealed.
- SP_pred_refs cannot become dangling.
- Calls from C to Prolog now undergo module name expansion and goal expansion, just like calls to `call/1`.
- Bug fixes in `SP_cons_list()` and `SP_cons_functor()`.
- Memory overflows after `SP_open_query()` are safe.
- Stream position terms now preserve after end of stream conditions.
- Cyclic terms are detected in arithmetic function arguments.
- Predicates defined by goal expansion only can be exported.
- The `character_escapes` flag is obeyed in ISO execution mode.
- Debugger bugs fixed: showing source code in Emacs; the `a`, `o`, and `r` commands.
- The Emacs interface function `prolog-comment-region` now uses triple percent signs, to cater for `indent-region`.
- The `clpfd:full_answer` functionality has been repaired, affecting `frozen/2`, `clpfd:attribute_goal/2` and `clpfd:fd_copy_term/3`. `clpfd:fd_global/3` is now a meta-predicate.

- `sockets:socket_select/[5,6]` are now steadfast; better error handling.
- `sockets:socket_select/[5,6]` now work correctly with non-socket streams that use file descriptors on systems where sockets and file descriptors are treated the same (i.e. not Windows).
- `system:working_directory/2` is now insensitive to any loads in progress. Its arguments are not subject to `absolute_file_name/2` processing—that was never intended.
- `timeout:time_out/3` now cleans up properly after abort.
- `library(bdb)` now handles wide characters, e.g. in error messages.
- `library(clpfd)` now cleans up properly after integer overflows, and does not assume a 32-bit architecture.
- `clpfd:cumulative/[4,5]` now check that the resource limit is not exceeded by any single task.
- Glue code generated for foreign resources (C and Java) did the wrong thing for `[-term]`. The problem occurred if the foreign function did many calls to `SP_new_term_ref()` or if it raised an exception.
- Fixed a problem with `spld` and `splfr` under Windows 95/98.
- On AIX `spld` and `splfr` tried to use a nonexisting file. The file ('`sprt.exp`') is now included in the distribution.
- `halt/0` and `abort/0` are handled better in runtime systems of type '`--main=load`' and '`--main=restore`'.
- The Java interface (`library(jasper)` and `se/sics/jasper/SICStus` etc.) has been improved:
  - Fixed lots of bugs.
  - The Prolog runtime system is no longer de-initialized at random by the Java garbage collector.
  - All Java methods now properly synchronize with the Prolog runtime system to ensure thread safety.
  - There are no longer any known memory leaks when calling Java from Prolog or vice versa.
  - `SPTerm` and `SPQuery` now properly detect improper usage and raise exceptions instead of crashing in the Prolog runtime system.
  - Enhanced meta-call interface `jasper_call/4` makes foreign resources and `splfr` strictly optional when calling Java from Prolog.
  - The constructor `SPTerm()` is no longer public, it was always documented as `"should really have been private"`. Use the contructor `SPTerm(SICStus)` instead.
  - The exception `IllegalCallerException` is no longer used. You should change your code to reflect this. One possible change is to change `throws IllegalCallerException` into `throws SPException`, this works for the 3.8.4 version as well.
  - The exception `IllegalTermException` is new. It is signalled when attempting to use a SPTerm where the corresponding SP_term_ref is no longer valid. You need to update your code (typically adding `throws IllegalTermException`). One

possible change is to use the less specific `throws SPException` instead of `throws IllegalTermException`, this should work for the 3.8.4 version as well.

  - It is now possible to create terms and queries by reading from a string. See `SICStus.readFromString()` and new versions of `SICStus.openQuery()` etc.
  - `SPPredicate` is now deprecated. The preferred method is to supply module and predicate name explicitly.
  - All calls to Prolog now behave as if wrapped in `call(M:Goal)` where `M` is the module specified when creating the query. This makes goal expansion and module name expansion do the right thing, i.e. behave as if entered interactively.
  - It is now possible to explicitly delete a `SPTerm` object, making the Prolog side SP_term-ref available for re-use. See `SPTerm.delete()`.
  - The documentation has been improved and expanded.
  - The examples have been updated and new examples added, notably a Swing demo with a Prolog top-level. The Prolog top-level is useful when debugging applications where Java is the top-level application.
  - `JavaServer` etc. is now more clearly marked as unsupported example code. It represents an unfinished sockets based Jasper interface. It does not belong in the `se.sics.jasper` package and will be removed at a later date.

- `library(tcltk)`. When Tcl/Tk calls Prolog, it now ignores the returned values of all unbound variables and variables with names *starting* with underscore '_'. In 3.8.4, it used to ignore only anonymous variables.

9.13 Changes introduced in version 3.8.6

Version 3.8.6 is mainly a bugfix release. New features:

- `SP_atom` documented as a data type.
- `library(jasper)` and Java foreign resources now support null object references. See section "Jasper Library Predicates" in *the SICStus Prolog Manual*.
- `library(jasper)` now works with JDK 1.3 (with some restrictions) and JDK 1.3.1 as well as JDK 1.2.2. See Section 5.1 [Supported Java Versions], page 13.
- On Linux, the path to the JDK libraries is now embedded (rpath) in the SICStus executable instead of in the Jasper foreign resource (libjasper.so). In most cases this should not be a user visible change. This was necessary due to differences between the Linux (glibc) and the Solaris dynamic loader.
- `InstallSICStus` should now understand TclPro directory structure when configuring Tcl/Tk.

Bug fixes:

- `splfr` code generation bug for `[-term]` and `+boolean`.
- `spld --sicstus=<PATH>` now works. You are unlikely to need it though.

- Better error reporting for incorrect type signatures to old style calls to `library(jasper)`.
- Jasper: `SPTerm.delete()` sometimes did not enable reuse of the deleted SP_term_ref.
- Under some versions of Windows (NT 4 but not Windows 2000) Java would crash under the following circumstances. `java` is the main application, Java tries to do `use_module(library(jasper))`, the *short* pathname of the '`SICStus Prolog\bin`' folder is on the `PATH` environment variable. As a work-around for the underlying Win32 `LoadLibrary` bug SICStus will now always use the long pathname when loading foreign resources.
- Under Windows, you can sometimes get an floating point divide by zero exception when embeding SICStus into other applications. The symptom was a crash with something like `"Exception: 0xc000008e (EXCEPTION_FLT_DIVIDE_BY_ZERO)"`. This also happened for some applications that use Visual Basic for Applications (VBA) with the SICStus Visual Basic module (vbsp.dll). A thorough discussion of this issue and a solution is available in '`library/vbsp/sp_fpwrap.h`'. This solution is now used by the SICStus Visual Basic module. We have had reports of this issue affecting FileMaker, Rational Rose and Visio.
- `library(bdb)` now complains if run with a different version of Berkeley DB than what was used for building it (BDB 2.7.7).
- Now uses `malloc()` for memory allocation when invoking SICStus from Java, also on Linux. The default memory allocation method (using `sbrk`/`brk`) is not thread safe on any platform.
- Avoid spurious error message after `C-c a`.
- Loading '`.po`' files and saved-states: work around GCC bug affecting endianness conversion.
- Standard term comparison on stream position terms didn't work after `seek/4`.
- `dif/2` and friends: memory management bug.
- `format/[2,3]`: avoid spurious time-out exceptions.
- `freeze(V,V)` behavior bug.
- `number_chars/2, number_codes/2`: bug affecting empty lists.
- `library(bdb)` now verifies that BDB version 2.7.7 is used.
- `library(clpfd)`: missing distribution files; buggy action handling of user-defined global constraints; wrong answers in `disjoint1/[1,2]`, `disjoint2/[1,2]`, `serialized/[2,3]`, and `cumulative/[4,5]`; integer overflow checks; error detection in FD set operations; complexity of `fd_closure/2` and `fd_copy_term/3`; entailment action in `element/3`.
- `library(clpq,clpr)`: bug affecting `bb_inf/3` and strict inequalities.
- `library(jasper)` now enforces the use of `malloc` for memory management, also on Linux.
- `sockets:socket_buffering/4`: bug handling 3rd arg.
- `library(tcltk)`: some demos depended on current working directory.
- `library(xref)`: handling of `catch/3`.

9.14 Changes introduced in version 3.8.7

Version 3.8.7 is mainly a bugfix release. New features:

- Support for MacOS X 10.x. This includes support for Tcl/Tk (see Chapter 4 [Tcl/Tk Notes], page 12).
- Made the Windows version faster using an improved version of the tweaks used in 3.8.3.

Bugs fixed:

- Performance problems in meta-calls, exception handling, `findall/3` and friends, `sort/2`, `keysort/2`.
- `atom_codes/2`, `atom_chars/2`, `number_codes/2`, and `number_chars/2` were not steadfast.
- `library(timeout)`: Some timeouts were ignored if they occurred during exception handling.
- `library(timeout)`: If the timer cannot be set up (using `setitimer`), then a system error exception is raised. This happens on Solaris in multi-threaded applications, e.g. when using SICStus with Java.
- `SP_deinitialize()` not working properly.
- Infinite loop in dynamic code memory management.
- Arguments to format spec '`~@`' were not called as fully general goals.
- Loading and unloading resources was vulnerable to changing file search paths.
- `listing/[0,1]` did not module prefix bodies of imported predicates correctly.
- `library(clpfd)`: some Boolean constraints were incorrectly macro-expanded; some type errors merely failed; over-zealous integer overflow detection in arithmetic.
- `sockets:socket_select/[5,6]` will work correctly also for input buffered socket streams. You no longer need to disable buffering with `sockets:socket_buffering/4` just to get `sockets:socket_select/[5,6]` to work.
- `library(sockets)`: A signal delivered to the process will no longer cause any socket predicates to give an error (proper '`EINTR`' handling). A downside to this is that SICStus no longer is interruptible with `^C` ('`SIGINT`') while blocking except in `sockets:socket_select/[5,6]`. In the case of `sockets:socket_select/[5,6]`, signal delivery will be treated as if the `select()` call had a zero timeout. Such spurious timeouts are a **potential backward compatibility issue.**
- `sockets:socket_buffering/4`: A crashing bug introduced in 3.8.6. Added argument validation so that it now fails on non-socket streams and other incorrect arguments. Note that the primary reason for changing socket buffering prior to 3.8.7 is gone since `sockets:socket_select/[5,6]` now work also for input buffered sockets.
- `system:file_property/2` failed to recognize symbolic links as such.
- Jasper: A Prolog side memory leak in `SPTerm.consFunctor`. This also affected all read-from-string type methods.
- Jasper: `SPTerm.getNumberChars` called an undefined native method.

- Under UNIX, the Prolog run-time would temporarily unblock all OS signals. This could lead to unexpected behavior when linking SICStus with other applications that use signals (such as Java).
- `library('clpqr/examples/mip')` would not load properly.
- Solaris, `spld --static` used incorrect options when specifying static linking to Sun Workshop C-compiler.

9.15 Changes Introduced in Version 3.9

9.15.1 Message and Query System

Every message issued by the Prolog system is displayed using the built-in predicate `print_message/2`. Similarly, all user input is handled by the new built-in predicate `ask_query/4`. The behavior of these predicates can be customized by a number of new hooks. For example, all I/O can be redirected from the standard streams to Tcl/Tk or similar windows. The default appearance of system messages has also changed.

The mapping from Prolog terms representing messages to printed messages is now completely transparent and customizable. The new `library('SU_messages')` file defines the mapping rules. Runtime systems can display system messages in standard format by loading this file.

A number of new hooks have been added for the message and query system: `generate_message_hook/3`, `generate_message/3`, `message_hook/3`, `print_message_lines/3`, `query_hook/6`, `query_class_hook/5`, `query_input_hook/3`, `query_map_hook/4`.

See section "Messages and Queries" in *the SICStus Prolog Manual*.

9.15.2 Determinacy Checker

The determinacy checker, `spdet`, is a new, powerful tool originally written by Dave Bowen and Peter Schachte.

The determinacy checker can help you spot unwanted nondeterminacy in your programs. This tool examines your program source code and points out places where unintended nondeterminacy may arise. Unintended nondeterminacy should be eradicated because

1. it may give you wrong answers on backtracking
2. it may cause a lot of memory to be wasted

See section "The Determinacy Checker" in *the SICStus Prolog Manual*.

9.15.3 Cross-Referencer

`library(xref)` of previous releases has been replaced by a much more powerful tool, `spxref`, originally written by Tom Howland.

The main purpose of the cross-referencer is to find unreachable code. To this end, it begins by looking for a definition for `user:runtime_entry/1` and also looks for initializations, hooks, module export lists and `public` directives to start tracing the reachable code from.

A second function is to aid in the formation of module statements. It can list all of the required `module/2` and `use_module/2` statements by file.

See section "The Cross-Referencer" in *the SICStus Prolog Manual*.

9.15.4 Common Object Model Client

Under Windows, `library(comclient)` makes it possible to control COM Automation objects. See section "Com Client" in *the SICStus Prolog Manual*.

9.15.5 The PiLLoW Web Programming Library

The PiLLoW library is a free Internet/WWW programming library, which simplifies the process of writing applications for such environment. The library provides facilities for generating HTML or XML structured documents by handling them as Prolog terms, producing HTML forms, writing form handlers, processing HTML templates, accessing and parsing WWW documents (either HTML or XML), accessing code posted at HTTP addresses, etc. See section "PiLLoW" in *the SICStus Prolog Manual*.

9.15.6 New CLPFD Features

Generally, performance has been improved, in terms of CPU time as well as memory. Specific new features:

- The predicates `assert/1`, `findall/3`, `raise_exception/1` and friends are now safe to use on non-ground terms containing domain variables.

  `labeling/2` takes a new option, which supports limited discrepancy search. To support this, the `apply_bound/1` API has been replaced by predicates called `first_bound/2` and `later_bound/2`.
- The new constraint `knapsack/3` is a domain consistent special case of `scalar_product/4`.
- The new constraint `global_cardinality/2` constrains the number of occurrences of given integers in a list.

- Arbitrary N-ary relations can be defined with the new constraint `case/4`.
- `disjoint2/2` takes a new option, which tells the constraint to perform stronger reasoning if some rectangles have the same origin and certain other conditions are fulfilled.
- `serialized/3` and `cumulative/5` take a new option, which tells the constraint whether to only adjust domain bounds. This is now the default.
- The `precedences/1` option of the same constraints has a more general format.
- The new constraint `cumulatives/[2,3]` generalizes `cumulative/[4,5]` by considering multiple resources, positive and negative resource consumption, and lower and upper bounds.

9.15.7 All-in-one Executables

It is now possible to embed saved-states into an executable. Together with static linking, this gives an executable that does not depend on external SICStus files. See section "All-in-one Executables" in *the SICStus Prolog Manual*.

9.15.8 Multiple SICStus Run-Times in a Process

It is now possible to have more than one SICStus run-time in a single process. See section "Multiple SICStus Run-Times" in *the SICStus Prolog Manual*, for details.

9.15.9 Miscellaneous

- The new built-in predicate `read_line/[1,2]` read one line of input into a code-list.
- The new built-in predicate `stream_position_data/3` accesses the fields of a stream position term.
- The new built-in predicate `goal_source_info/3` decomposes annotated goals into a goal proper and a source position. Mostly used in displaying error messages.
- The built-in predicate `undo/1`, which posts a goal for execution on backtracking, has been revived.
- `absolute_file_name/3` is a new variant of `absolute_file_name/2` taking a number of options to give full control over the conversion from relative to absolute filename.
- `read_term/[2,3]` takes a new option, which controls whether the *layout-text-item* that follows the terminating '`.`' should be consumed.
- A resource error exception is raised when there is insufficient memory to continue execution. In previous versions, this condition caused SICStus Prolog to abort the computation and to reinitialize itself.
- `trimcore/0` trims the Prolog stacks.
- Debugger changes:
  - Breakpoint conditions now can contain the usual Prolog connectives '`,`', '`;`', '`->`', and '`\+`'.

  - A breakpoint for which the test conditions evaluated successfully is always applied, even if the action conditions fail.
  - A hook, `user:breakpoint_expansion/2`, has been added for user defined breakpoint conditions.
  - Advice-points are now allowed to set all three action variables (mode, show, command).
- `library(bdb)` provides a way to flush all modified records to disk after each operation.
- `library(charsio)` provides the new predicates `read_term_from_chars/3` and `write_term_to_chars/[3,4]`.
- `library(system)` provides the new predicates `now/1` and `datime/2`.
- `library(clpq,clpr)` provides the new predicate `projecting_assert/1`.
- `SP_set_read_hook()` is now obsolescent. See section "Hooks" in *the SICStus Prolog Manual*.
- `SP_event()` can now be called from arbitrary OS threads. See section "Calling Prolog Asynchronously" in *the SICStus Prolog Manual*.
- `splfr` generates a header file from the `foreign/[2,3]` declarations. This file should be included in the corresponding C file to protect against incorrect `foreign` declarations and also to ensure compatibility with various compile time settings used by `splfr`.

  The name of the file can be specified with '`--header=`*`NAME`*'. To produce the file and nothing more, you can use the new '`--nocompile`' flag as in

  ```
  % splfr --header=foo.h --nocompile foo.pl
  ```

  This is especially useful when '`foo.h`' is a Makefile prerequisite.
- `splfr` and `spld` now use descriptive names for generated files, especially when the flag '`--keep`' is specified.
- Under Windows, it is now possible to link an application with a static version of the SICStus run-time. It is also possible to link with static versions of foreign resources.
- A statically linked SICStus run-time can load foreign resources dynamically.
- `spld --moveable` now produces an executable that can be moved, together with its directory tree (see section "Runtime Systems on UNIX Target Machines" in *the SICStus Prolog Manual*) on Solaris and Linux (it always worked under Windows).
- New `spld` options,

  '`--lang`' Selects the Prolog dialect used for run-time systems created with '`--main=load`' and '`--main=restore`'.

  '`--memhook`'
  Selects memory allocation method (`SP_set_memalloc_hooks()`).

  You can use `spld --help` to get details on these and other options.
- The development system (`sicstus` and, under Windows, `spwin`) now take option '`--iso`' and '`--sicstus`' to start up in ISO Prolog mode and SICStus Prolog mode respectively.
- New C preprocessor macros with version info are now defined when including '`<sicstus/sicstus.h>`'. For SICStus 3.9.1, they were defined as:

```
#define SICSTUS_MAJOR_VERSION 3
#define SICSTUS_MINOR_VERSION 9
#define SICSTUS_REVISION_VERSION 1
#define SICSTUS_BETA_VERSION 0

/* Two digit position for MAJOR, MINOR and REVISION */
#define SICSTUS_VERSION 30901
```

- New C API function `SP_read_from_string()` takes a string representation of a Prolog term and a table of variable values and creates a Prolog term. By reading a goal from a string and passing the resulting term to `call/1`, this also gives a very simple way to call arbitrary goals from C. See section "Mixing C and Prolog" in *the SICStus Prolog Manual*.
- New C API function `SP_define_c_predicate()` makes it possible to dynamically define a Prolog predicate that calls a C function. See section "Mixing C and Prolog" in *the SICStus Prolog Manual*.
- New C API functions `SP_mutex_lock()`, `SP_mutex_unlock()`, C type `SP_mutex` and static initializer `SP_MUTEX_INITIALIZER`. These provide a platform independant (recursive) mutual exclusion lock. These are mainly useful when more than one SICStus run-time is used (in different threads) in the same process. See section "Operating System Services" in *the SICStus Prolog Manual*.
- The environment variables `SP_APP_DIR` and `SP_RT_DIR` are set (by `SP_initialize()`) to the folder containing the executable and the folder containing the SICStus run-time, respectively. This provides a location independent way to locate files (such as saved-states) that are located together with the application. Also available as the file search aliases `application` and `runtime`. See section "Input Output" in *the SICStus Prolog Manual*.
- On Win32, the C signal `SIGBREAK` will cause `halt/0` to be called in a development system. This will ensure that `halt/0` is called in `sicstus` when the console window is closed or when the user logs off. The windowed version of SICStus (`spwin`) also sends itself a `SIGBREAK` when the GUI window is closed, causing `halt/0` to be called.

9.15.10 Incompatibilities with Previous Versions

- The semantics of `absolute_file_name/2` has changed slightly. It no longer looks for an existing file with a '`.pl`' extension if the given relative file name has no extension. The old behavior can be achieved with:

```
old_absolute_file_name(RelFileSpec, AbsFileName) :-
   Options = [ file_type(source),
               access(exist),
               file_errors(fail) ],
   (  absolute_file_name(RelFileSpec, AbsFileName, Options)
   -> true
   ;  absolute_file_name(RelFileSpec, AbsFileName, [])
   ).
```

- The hook `debugger_command_hook/2` takes a different first argument.

- The built-in predicate `print_message/2` does not take a `force/1` severity.
- The UNIX-specific built-in predicates `stream_interrupt/3` and `stream_select/3` were unsafe, and have been removed. The latter can be replaced by the following equivalent definition, which works for file descriptor streams under UNIX and, for socket streams, under Windows as well:

```
:- use_module(library(sockets)).
stream_select(Streams, TimeOut, ReadStream) :-
        socket_select([], _, TimeOut, Streams, ReadStream).
```

- Exceptions in hooks are now caught locally instead of propagating into the calling code.
- The Prolog predicates for generating glue code for the foreign language interface are no longer available. The only supported method for generating foreign resources are `splfr`. The removed predicates are `link_foreign_resource/6`, are `prepare_resource_table/2`, and `prepare_foreign_resource/3`. See section "The Foreign Resource Linker" in *the SICStus Prolog Manual*, for details on using `splfr`.
- The predicate `reinitialise/0` is gone.
- The '`--import`' flag to `splfr` is no longer supported.
- The CLPFD `apply_bound/1` API for branch & bound search has been replaced by predicates called `first_bound/2` and `later_bound/2`.
- The CLPFD constraints `all_different/[1,2]` and `all_distinct/[1,2]` require bounded domains.
- The CLPFD constraints `serialized/[2,3]` and `cumulative/[4,5]` only prune the bounds of domains, not inside them, unless the `bounds_only(false)` option is given.
- The Berkeley DB interface is now compatible with Berkeley DB 4.0.14 instead of 2.7.7.
- `library(db)` has been removed.
- `library(xref)` has been replaced.
- `library(flinkage)` is no longer supported. It is still present to help porting really old code.
- The header file '`<sicstus/sicstus.h>`' now uses stricter function type prototypes. This may expose problems in code that used to compile without warnings in previous versions of SICStus.
- Java foreign resources are no longer supported. Java foreign resources offer no advantages compared to `jasper_call/4` introduced in SICStus 3.8.5 so you should migrate your code even if you are still using SICStus 3.8.
- Java now loads '`libspnative.so`' ('`spnative.dll`' under Windows) where it used to load '`libjasper.so`'. This will probably be invisible to most users.
- The Java methods in `se.sics.jasper` that operate on a SICStus object will now throw an error if called from a thread other than the one that created the SICStus object. This is similar to the behavior in early versions of 3.8.

  SICStus 3.9 introduces a new thread safe Jasper API, which is not backwards compatible with the Jasper API of 3.8. However, the old API has been amended to be compatible with the thread safe API, making it possible to write Java programs that can run in both modes.

- Signal handlers installed by `SP_signal()` are now deferred until Prolog can call them safely. This makes `SP_signal()` unsuitable for handling synchronous signals such as '`SIGSEGV`'. See section "Signal Handling" in *the SICStus Prolog Manual*.

9.15.11 Bugs Fixed in 3.9

- `at_end_of_stream/0`, `get/1`, `get0/1`, `put/1`, `skip/1`, `tab/1` incorrectly required that the current input stream be a text stream.
- `stream_property(S, end_of_stream(...))` now works for tty streams.
- `user:goal_expansion/3` was called with the wrong second argument for imported and built-in predicates.
- `require/1` was broken.
- The escape sequence '`\c`' could be misread.
- In ISO mode, terms were printed with non-ISO escape sequences.
- `charsio:format_to_chars/[3,4]` are now meta-predicates.
- `charsio:format_to_chars/[3,4]` is now re-entrant.
- `library(clpfd)`: some Boolean constraints were incorrectly macro-expanded; some type errors merely failed; over-zealous integer overflow detection in arithmetic; unifying domain variables did not work reliably and failed silently when type errors were intended.
- `library(clpq,clpr)`: Variables introduced via `ordering/1` were not initialized fully.
- `library('linda/client')`: `shutdown_server/0` now raises an existence error if there is no connection to the server available.
- Emacs line break points now work for module-files.
- Windows now always uses `^Z` where UNIX uses `^D` to signal end of file on input from the terminal.
- Under Windows, when running SICStus under Emacs, the SICStus process sometimes got stuck in a tight loop when Emacs closed the connection to the SICStus sub-process. It will now exit instead; see Section 3.6 [Windows limitations], page 9.
- Under Windows, `splfr` now converts the resource name to lowercase. This is to match the fact that SICStus converts path names to lowercase under Windows.
- As of SICStus 3.8.7, for `sockets:socket_select/[5,6]`, signal delivery will be treated as if the `select()` call had a zero timeout. Such spurious timeouts are now handled invisibly within `library(sockets)` and will not be seen by user code.

9.15.12 Known Problems with This Release

- Some Prolog libraries that depend on foreign code (e.g. `library(bdb)`, `library(tcltk)`) can only be loaded by one SICStus run-time in the same process. The second run-time that attempts to load e.g. `library(bdb)`, will raise an exception. This only affects code that loads several SICStus run-times (e.g. using the Java interface).

- The API for associating destructors with external objects is in place but is not documented. It is, at present, only used by the COM client library. Contact `sicstus-support@sics.se` if you have any questions about it.
- See Section 9.15.10 [3.9 Incompatibilities], page 49.

9.16 Changes Introduced in Version 3.9.1

Version 3.9.1 is mainly a bugfix release. New features:

- Support for Compaq Tru64 5.1, IRIX 6.5, MacOS X 10.1 (Darwin), AIX 4.3.3 and HP-UX 11.
- Jasper (the Java interface) is now supported on most platform.
- `library('linda/server')` now also prints the address of connecting clients.
- `library('linda/server')` can apply a user specified filter to reject or accept incoming connections; see section "Server" in *the SICStus Prolog Manual*.
- Added '`--moveable`' option to `splfr`.
- Added '`--nocompile`' option to `spld`.
- `clpfd:case/4` takes a new option, which effectively extends the relation by one argument, corresponding to the *ID* of the leaf node reached by a particular tuple. Every *ID* should be a unique integer. Several consistency checks added.
- `clpfd:all_different/1` takes a new option `consistency(C)` to control the level of consistency to be achieved. This supersedes the old option `global(Bool)`.
- `clpfd:sorting/3` is a new constraint, capturing the relation between a list of values, a list of the values in ascending order, and their positions in the original list.
- `clpfd:fd_var/1` is new; checks that the argument is currently an unbound variable that is known to the CLPFD solver.
- `SP_get_integer_bytes()`, `SP_put_integer_bytes()` can be used to pass arbitrarily sized integers (bignums) between C and Prolog. See section "Accessing Prolog Terms" in *the SICStus Prolog Manual*.

Other changes:

- Changed the name of the directory where run-time systems on target machines (see section "Runtime Systems on UNIX Target Machines" in *the SICStus Prolog Manual*) can find supporting files. The directory name is now version specific, e.g. '`$SP_APP_DIR/sp-3.9.1/`'. If no such directory can be found, the old name ('`$SP_APP_DIR/lib/`') is tried as well.

Bugs fixed:

- Compiler bug affecting very large clauses.
- Dangling pointer issue with suspended goals.
- Native SPARC in-core compiler indexing bug.

- Top-level problems: phoney goal shown in syntax error messages; blocked goals suppressed if no bindings to display.
- `listing(_:_)` printing imported clauses multiple times.
- Character code 160 (no-break space) is a layout-char in ISO 8859/1, not a symbol-char.
- CHR code transformation bug.
- CLPFD: `knapsack/3` broken; problems with very large domain bounds; bugs in `disjoint2/2` with `wrap/4` option; missing solutions in `case/4`; `fd_copy_term/3` did not preserve all domain variables.
- The '`--output`' option to `splfr` broken.
- The global variable `sp_GlobalSICStus` is now defined also when '`--main=none`' is passed to `spld`.
- `spxref`: did not follow arbitrary directives; problem with absolute file name resolution.
- `Jasper`: client thread did not detect failure of server thread to create a SICStus object.
- `Jasper`: bugs in class `Jasper$JasperProlog`: `consFunctor` and `consList` gave `NoSuchMethodException`.
- `Jasper`: `[-term]` in method meta-calls to Java from Prolog did not work.
- `Jasper`: Java longs (64-bit signed ints) passed between Java and Prolog are no longer truncated to 32 bits.
- `Jasper`: bugs in class `Jasper$JasperTerm`: `compare`, `getArg`, `getDouble`, `getFunctorArity`, `getList`, and `unify` did not work properly.
- `Jasper`: bugs in class `SICStus`: `consFunctor(String, Term[])` and `newTerm(String, Term[])` gave `ClassCastException`.
- `Jasper`: Fixed bug causing `ClassCastException` when doing callbacks in thread safe Jasper. Jasper did not keep track of current caller properly.
- Foreign resources and data resources could not have names containing '`w`' or '`W`'.
- SICStus could crash in the deinit functions of '`--multi-sp-aware`' foreign resources.
- Exception (`SP_raise_exception()`) and failure (`SP_fail()`) from foreign resource init or deinit functions are now handled; see section "Init and Deinit Functions" in *the SICStus Prolog Manual*.
- `unload_foreign_resource(`*`Name`*`)` now raises an existence error instead of merely failing if no foreign resource *Name* is loaded.
- Printing and reading floats was broken in locales redefining the radix character.
- `library(fastrw)` now raises an existence error when reading a term from an already closed stream, similar to `get/1` etc. It used to give some arbitrary error in this case. The library still does not handle end of file within terms.

  `library(fastrw)` is an (undocumented) support library used by `library('linda/client')`, `library('linda/server')` and `library(bdb)`.
- `linda:linda/1` is now a meta-predicate.
- `library(comclient)` would crash for in/out arguments passed as a mutable.
- `system:mktemp/2`, `system:tmpnam/1`: Added error checking.

  On some platforms the linker would complain about possibly unsafe use of `mktemp()` when linking statically with the `system` foreign resource. This has been worked around

by using `mkstemp()` internally when available. For this reason, file names created by `system:tmpnam/1` may differ from earlier releases.

The robustness and security issues surrounding the use of the ANSI C `mktemp()` and `tmpnam()` routines also applies when using `system:mktemp/2` and `system:tmpnam/1`.

- Under UNIX `library(sockets)` now ensures that '`SIGPIPE`' is ignored in run-time systems (It was always ignored on development systems). This avoids having the process killed by SIGPIPE when SICStus writes to a socket where the other end has closed the connection.

9.17 Changes Introduced in Version 3.10

9.17.1 New Features

- On 32-bit architectures, prior to release 3.10, the total data space could not exceed 256 MB. As of release 3.10, this limit only applies to the Prolog stacks, whereas other objects (static and dynamic code, symbol tables, etc.) can be stored anywhere in virtual memory.

  The fix involves a partial redesign of the memory allocator. The API function `SP_set_memalloc_hooks()` has changed incompatibly. See section "Initializing the Prolog Engine" in *the SICStus Prolog Manual*.

  As a consequence of the changes to the memory allocator, the `sicstus` option '`-m`' and the `spld` option '`--memhook`' are no longer meaningful. They are ignored for compatibility but will be removed in some future release.

  As another consequence of the changes, the "aligned pointer" concept and all such requirements in the foreign language interface have been dropped.

  Under Windows and Linux the memory for the Prolog stacks are now pre-allocated when initializing the SICStus run-time. This makes it less likely that the memory allocations of other parts of the application, such as Java, will conflict with the allocation needs of SICStus.

  This pre-allocation affects applications that use more than one SICStus run-time in the same process, e.g. from Java. The first initialized SICStus run-time will typically reserve all memory usable for Prolog stacks. This means that subsequent SICStus run-times initialized in the same process will fail. Set the environment variable `PROLOGMAXSIZE` to a suitable value to work around this problem. See section "Environment Variables" in *the SICStus Prolog Manual*.

  As part of the memory manager re-design, the SICStus API functions `SP_malloc()`, `SP_calloc()`, `SP_realloc()` now take `size_t` size arguments instead of `unsigned int`, just as their standard C equivalents.
- `spld` option '`--more-memory`'. This option modifies the linker script on x86 Linux so that almost 256MB can be used by the Prolog stacks. It is ignored on other platform. This option should be considered experimental.
- Support for AIX 5L 5.1 32bit and 64bit.

- The Application Builder `spld` takes new options:

  '`--resources-from-sav`'
  '`--no-resources-from-sav`'
  : Automatically extract foreign resource names from embedded '`.sav`' file. Thus you do not need to know what foreign resources are used by the loaded Prolog code. This is now the default for static executables when no other resources are specified apart from an embedded saved-state. To turn off this feature, use '`--no-resources-from-sav`'.

  '`-E`'
  '`--extended-rt`'
  : Create an extended runtime system. In addition to the normal set of built-in runtime system predicates, extended runtime systems include the compiler. Extended runtime systems require the extended runtime library, available from SICS as an add-on product.

  See section "The Application Builder" in *the SICStus Prolog Manual*.
- A new environment variable `SP_LIBRARY_DIR` is set during SICStus initialization to the location of the SICStus library files. This is similar to `SP_APP_DIR` and `SP_RT_DIR`. See section "Environment Variables" in *the SICStus Prolog Manual*.
- In the Jasper package the interface Term has a new method `toString(Term options)`. The implementing class `SPTerm` calls `write_term/3`.
- Under Windows NT/2000/XP, `library(timeout)` now measures the user mode time of the SICStus thread. Under Windows 95/98/ME, the time is still walltime.
- Under Windows NT/2000/XP, `statistics(runtime, ...)` now measures the user time of the SICStus thread. This is the same as process user time for a single-threaded program but makes `statistics(runtime, ...)` meaningful also in a program with several threads (such as when using Java or multiple SICStus runtimes).
- `library(bdb)`: now requires Berkeley DB 4.1.24 instead of 4.0.14. The predicate `db_sync/1` is new.
- `library(clpfd)`:
  - The constraints `assignment/2` and `global_cardinality/2` take new options, associating with the constraint a cost, which is reflected into a domain variable.
  - The constraint `lex_chain/[1,2]` expresses the fact that several vectors of domain variables are in ascending lexicographic order.
  - The predicate `fd_global/3` has been extended with new options.

9.17.2 Bugs Fixed

- Compiler bug affecting very large clauses if `profiledcode`.
- Calling `restore/1` when `library(clpfd)` had been loaded would sometimes crash SICStus with a memory access error.
- `library(comclient)` did not dispose of all references to COM objects. `comclient_garbage_collect/0` did not work correctly (but should not be needed if you use `comclient_release/1`).

- `SP_garbage_collect_external_objects()` would reclaim live objects. This affected `comclient_garbage_collect/0`.
- The Java flag '`-Dsicstus.path`' should no longer be needed when using Jasper, except in exceptional circumstances. SICStus will use the location of the SICStus run-time library, e.g. `libsprt311.so` to set this up automatically.
- `SP_initialize()` will now return an error code if memory could not be allocated. It used to terminate the process. Among other things this ensures that creating a `se.sics.jasper.SICStus` object will throw an exception instead of terminating Java if there is insufficient memory.
- `library(timeout)` is now more precise, in particular when calls to `time_out/3` are nested. The resolution has not been improved, it is still on the order of a few tens of milliseconds.
- `library(timeout)` under Windows could sometimes deadlock or otherwise behave erratically.
- `format/[2,3]` reported errors inconsistently.
- `undo/1`: missing meta-predicate declaration.
- `library(clpfd)`: constraints without arguments handled incorrectly; inconsistent overflow handling; heap overflow detection; bugs in `element(X,L,Y)` with non-ground `L`; bugs in `disjoint2/2` with `margin/4` option.
- `library(chr)`: spurious error messages could occur when loading CHR files from all-in-one executables.
- `library(objects)`: Dangling pointer problems with instances if the class was asserted to after the instance had been created.
- `library(jasper)`: Memory leak when calling a Java method. Segmentation fault on Solaris when passing arguments declared as `+term`.
- `library(gauge)`: The help button did not work under Windows if SICStus was installed in the default location or other folder with a space in the path.
- Foreign resources generated with `splfr --exclusive-access` could not be loaded more than once into the same process. The only SICStus library affected is `library(system)`.
- Problems with loading and unloading foreign resources if the name was not unique or if file path facts had changed.
- `unload_foreign_resource/1` now takes the *name* of a foreign resource. For backward compatibility it still accepts a file name treated as for `load_foreign_resource/1`. See section "Foreign Resources" in *the SICStus Prolog Manual*. `unload_foreign_resource/1` is still a meta-predicate but the module is ignored.
- SPARC native code issues: flushing the instruction cache, redefining imported predicates.
- The `spdet` and `spxref` tools now work also when installed as part of a pre-built binary installation.
- The debugger command '`t`' (backtrace *n*) showed too much.
- `SP_read_from_string()` would leak memory.
- SICStus runtimes will work correctly for executables located at the root of a drive, e.g. as `D:\foo.exe`.

9.17.3 Other Changes

- Under Windows, the name of the static SICStus run-time has changed from `sprt310.lib` to `sprt310_static.lib`. At the same time the import library for the dynamic SICStus run-time has changed to `sprt310.lib`. This change should only affect those that do not use `spld` to build executables.
- Under Windows it is not longer possible to link with the SICStus run-time using MS Visual Studio 5. This has been unsupported for years but now it has really ceased to work.
- Under Windows the windowed version of SICStus will put up a license entry dialog if the license has expired or is invalid. There is also a menu item for changing the license, e.g. when changing from a time limited license to a permanent license. This renders `splm` obsolete under Windows.
- The Visual Basic interface no longer loads any supporting Prolog code, i.e. `vbsp.pl` and `vbsp.po` are gone. A side benefit of this is that you can call `restore/1` from Visual Basic, even though calling `load_files/2` may be better.
- `user:file_search_path/2` and `user:library_directory/1` are undefined at startup, but behave as if they were dynamic, multifile predicates.

9.18 Changes Introduced in Version 3.10.1

Version 3.10.1 is mainly a bugfix release.

9.18.1 New Features

- The Emacs interface is now easier to set up. The SICStus Emacs interface can now figure out the location of the SICStus executables etc. Simply load the new file '`sicstus_emacs_init.el`' located in the SICStus '`emacs/`' folder from your Emacs initialization file ('`.emacs`'). See '`sicstus_emacs_init.el`' for details.
- `library(comclient)`: It is now possible to pass objects as arguments to methods.
- The new procedure `PrologDeInit` now makes it possible, and recommended, to deinitialize the Prolog engine from the Visual Basic interface. See section "Deinitializing the Prolog Engine From VB" in *the SICStus Prolog Manual*.
- New one-letter top-level commands: `<` for setting the printdepth, and `^` for subterm navigation.

9.18.2 Debugger Changes

- The meaning of the `goal(G)` breakpoint test has changed: it succeeds if the current goal is an instance of `G`, instead of checking if the current goal unifies with `G`. Similarly for the `ancestor(A)` test.
- Any variable instantiations introduced in the evaluation of the test part are undone before executing the action part.

- The `parent_pred/[1,2]` tests perform module name expansion on their first argument.
- The `proceed/2` and `flit/2` commands are allowed at non-call ports, too; their module name expansion rules have changed.
- `current_breakpoint/4` is changed to `current_breakpoint/5`. It now returns the breakpoint conditions in exactly the same form as given to `add_breakpoint/2`.
- The interpretation of the first argument of `execution_state/2` has changed; the order of list elements does not matter any more.
- The new `get` condition allows to access action variables from within the action part.
- Test `exited/1` replaces test `last_port/1`.
- A breakpoint condition can be a variable at `add_breakpoint` time, as long as this variable gets instantiated by evaluation time.
- Breakpoint macros are expanded at `add_breakpoint` time, rather than during evaluation.
- It is not possible any more to redefine a built-in breakpoint condition using the breakpoint expansion mechanism.

9.18.3 Bugs Fixed

- `format/[2,3]`: bug affecting format specs '`~g`' and '`~G`'.
- `read_line/2`: was not steadfast.
- `call_cleanup/2`: the cleanup goal didn't preserve pending exceptions.
- `load_files/[1,2]`: could sometimes attempt to load a file of the wrong type.
- Switching on the debugger in an embedded command in a file loaded with the '`-l`' option didn't work.
- `save_program/[1,2]` would give a foreign resource existence error if called multiple times.
- Problems with ISO mode and some library modules.
- `library(comclient)`: fixed several problems.
- `system:file_property/2` now accepts terminating slash or backslash in directory names.
- Fixed several problems when file-names, working directory or home-directory contains non-ASCII character.
- `library(clpfd)`: could fail to unblock blocked goals; indexical propagation bug; bugs in `disjoint1/2` and `disjoint2/2` with `margin/3` resp. `margin/4` option.
- Win32 memory allocator could leak (uncommitted) memory in `SP_initialize()`.
- `SP_deinitialize()` would not free all memory.
- On AIX `spld` now builds executables with '`-brtl -bnortllib`' instead of just '`-brtl`'. This improves compatibility with Java. In particular the error '`_XmGetDefaultDisplay cannot be used prior to VendorS.Initialize`' when invoking AWT with `sicstus` as top level application.

9.18.4 Other Changes

- MacOS X: Support for the native Aqua version of Tcl/Tk (8.4.2). See Chapter 4 [Tcl/Tk Notes], page 12. The interaction between SICStus Tcl/Tk and the Mac OS X '`dock`' is still rather primitive. There is, for instance, no way to specify an icon or to handle the dock '`Quit`' command.
- The Visual Basic interface now pass UNICODE characters to `PrologOpenQuery`, `PrologOpenQuery`, `PrologQueryCutFail`, `PrologGetString`, `PrologGetStringQuoted`, and `PrologGetException`. This improves the handling of international characters (characters outside the 7-bit ASCII range).

  This change should not affect your Visual Basic code but the '`vbsp.bas`' file has changed. This change also affects those that use the `vbsp.dll` from other languages than Visual Basic.

  Characters outside the 7-bit range may not work under Windows 95.

  The '`library/vbsp/examples/train/`' sample project now use characters outside the 7-bit range.
- A better string hash function yielding faster atom lookup.
- The `open_hook` of the default '`wcx-box`' no longer ignores the `option` argument. In particular it is now posible to read and write files using ISO 8859/1 (Latin 1) even if `SP_CTYPE` is '`utf8`'; see section "WCX Hooks" in *the SICStus Prolog Manual*.

9.19 Changes Introduced in Version 3.11.0

9.19.1 New Features

- `library(spaceout)` limit the amount of memory used while running a goal; see section "Meta-call with Limit on Memory Use" in *the SICStus Prolog Manual*.
- `library(prologbeans)` is a package for issuing Prolog queries from a separate Java process, so as to avoid the memory conflicts and other undesirable interaction that can occur with Java and Prolog in the same process. See section "Access SICStus from Java" in *the SICStus Prolog Manual*. **Please note:** Feedback is solicited on this first release. Future API changes are likely.
- `library(xml)` is an unsupported package for parsing XML. See section "Parsing and Generating XML" in *the SICStus Prolog Manual*.
- There is now an uninstallation script for UNIX; see Section 2.1.3 [The Uninstallation Script], page 3.
- Jasper: The method `Term.getObject()` is no longer a no-op in multi-threaded Jasper.
- Jasper: In `class SPException` the method `toString` returns a printed representation of the underlying Prolog exception term (if one exists) limited by a print depth of 5. The method `getTerm` returns a `SPTerm` instance representing the Prolog exception term.

- Support for HP-UX 1.6i (B.11.22) on Itanium 64bit (IPF64).

9.19.2 Bugs Fixed

- The Windows installer would not install the Emacs setup file.
- Under Windows and x86 Linux, a number of problems related to floating point processing caused by non-standard C compiler floating point optimizations. Symptoms include memory access errors and incorrect printing of some numbers, e.g. `3.1E11`.
- Ill-formatted output in response to the debugger's *g* (print ancestor goals) command.
- Memory allocation bug in the Visual Basic interface.
- Due to a change in Linux kernels newer than 2.4.9 it was not possible to start more than one SICStus run-time in the same process. Linux distributions affected include Red Hat 7.3 and later (but not Red Hat 7.2).
- Under UNIX, a blocking `socket:socket_accept/[2,3]` is now interruptible with `^C` ('`SIGINT`'). Other signal handlers installed with `SP_signal()` will also run immediately if the signal arrives during a blocking socket accept operation.
- Work around a problem with the system call `inet_ntoa` that affected several predicates in `library(sockets)`. The problem affected IRIX and on 64bit AIX.
- `format/[2,3]` could intercept resource errors and report them as consistency errors; the "print float" formats were poorly documented.
- `save_program/[1,2]` had a memory leak, which could cause crashes.
- `library(clpfd)`:
  - Arithmetic constraints could cause infinite loops.
  - `in/2` and `in_set/2` could silently fail or succeed when they should raise a type or representation error.
  - `fdbg_start_labeling/1` was missing.
  - Bad Prolog term could arise from very large domains.
- `library(chr)`:
  - Attempted removal of attribute or constraint that is already gone would cause failure.
- Jasper: The implementation of `newObjectTerm(Object obj)` in multi-threaded Jasper caused a *NullPointerException: Creating or accessing SPTerm in wrong thread. (IllegalCallerException in disguise.)*
- Jasper: Exceptions are now propagated properly from Prolog to Java. This has never worked as intended before.
- The Gauge profiling tool was broken.
- `library(fastrw)`: `fast_read/2` after `peek_byte/2` and friends on the same stream did not work.
- MacOS X load_foreign_resource/1 would report run-time linker errors incorrectly.

9.19.3 Other Changes

- The most general case of `retractall/1` runs in time proportional to the number of clauses.
- When the reader sees a bad token, it reads up to the end of the token and terminates the sentence being read.

10 Generic Limitations

On 32-bit architectures, the total size of the Prolog stacks cannot exceed 256 MB. Under Linux, the default placement of the '`TEXT`' segment constrains the Prolog stack address range, so in practice the limit there is 128 MB unless the `spld` option '`--more-memory`' is used.

The number of arguments of a compound term may not exceed 255.

The number of atoms created may not exceed 262143 (33554431) on 32-bit (64-bit) architectures.

The number of characters of an atom may not exceed 65535.

NUL is not a legal character in atoms.

There are 256 "temporary" and 256 "permanent" variables available for compiled clauses.

Saved-states are not portable between 32-bit and 64-bit architectures, or from a system built with native code support to a system without native code support for the same architecture.

Indexing on big integers or floats is coarse.

11 Contact Information

Current support status for the various platforms can be found at the SICStus Homepage:

http://www.sics.se/sicstus/

Information about and fixes for bugs that have shown up since the latest release can be found there as well.

Send requests for ordering information to

sicstus-request@sics.se

Report bugs through the web interface

http://www.sics.se/sicstus/bugreport/bugreport.html.

or to

sicstus-support@sics.se

Bugs tend actually to be fixed if they can be isolated, so it is in your interest to report them in such a way that they can be easily reproduced.

The mailing list sicstus-users@sics.se is a mailing list for communication among users and implementors. To subscribe, write a message to majordomo@sics.se with the following line in the message body:

```
subscribe sicstus-users
```