SICStus Prolog Release Notes

Mats Carlsson et al.

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

Release 4.2.0
March 2011

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

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

Table of Contents

1 Overview

These notes summarize the changes in release 4 wrt. previous SICStus Prolog releases as well as changes introduced by minor releases and their patch releases. Platform specific information pertaining to certain parts of the system are also documented herein.

2 Platforms

Binary distributions of Release 4.2 are available for many platforms; see http://www.sics.se/sicstus/ for an up-to-date list.

SICStus has, at one time or another, been ported to many platforms, ranging from mobile phones to mainframes. If your platform is not currently listed on the download page, please let us know (sicstus-request@sics.se).

3 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 3.1 [UNIX installation], page 3. For example:

```
csh,tcsh> setenv PATH "/usr/local/sicstus4.2.0
/bin:$PATH"
sh,bash,ksh> export PATH="/usr/local/sicstus4.2.0
/bin:$PATH"
```

3.1 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.

3.1.1 Prerequisites

3.1.1.1 C Compiler and Linker

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

For Solaris you can download the Sun Studio C compiler from `http://developers.sun.com/`. For Mac OS X you can download XCode, which contains a C compiler, from `http://developer.apple.com/`. Linux distributions typically has a C compiler installed or installable through the system software update utility.

If a C compiler is not available, it is possible to use a *pre-built installation* on some platforms.

Pre-built installation is only recommended as a last resort; it is available by invoking `InstallSICStus` with the '`--prebuilt`' 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.

3.1.2 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 an interactive installation script called `InstallSICStus`. Type:

```
% ./InstallSICStus
```

and follow the instructions on the screen.

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, e.g. `library(bdb)` and `library(tcltk)`. 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.

3.1.3 The Uninstallation Script

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

3.2 Platform Specific Notes

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

Solaris 8 SPARC 64-bit

You cannot install (or build) the 64 bit version of SICStus using `gcc 2.x`. You need to use the Sun Workshop 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`. Recent versions of gcc, i.e. gcc 3.x or 4.x, do seem to work. To install with `gcc`, set the environment variable `CC` appropriately before invoking `InstallSICStus`.

Solaris Intel 64-bit, SPARC 64-bit

The following libraries are not supported: `library(bdb)`, `library(tcltk)`.

Solaris 8

The default thread library in Solaris 8 is incompatible with SICStus. The "Alternate Thread Library (T2)" must be used instead. This is ensured automatically for executables built with the `spld` tool. It is **not** ensured automatically when loading SICStus into Java or other programs not built by `spld`. See `http://developers.sun.com/solaris/articles/alt_thread_lib.html` for further information.

Problems caused by the old thread library include:

- `library(timeout)` does not work.
- Java hangs during initialization of a Jasper `SICStus` object.

This problem does not affect Solaris 9 or later.

Mac OS X

Mac OS X 10.5 and Mac OS X 10.6 are supported on Intel 32-bit and 64-bit. Mac OS X 10.6 is recommended, due to deficiencies in Mac OS X 10.5 that affects the SICStus I/O sub-system.

An executable built with `spld` will only work if there is a properly configured subdirectory '`sp-4.2.0`' 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.

When using third-party products like BDB, you may need to set up `DYLD_LIBRARY_PATH` so that the Mac OS X dynamic linker can find them. When using the SICStus development executable (`sicstus`), this should happen automatically, if the third-party products have been installed in the standard locations; see Section 8.2 [Berkeley DB on Mac OS X], page 18.

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:

```
bash$ ulimit -d
6144
```

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

```
bash$ ulimit -d unlimited
bash$ ulimit -d
unlimited
```

Please note: `ulimit` is a shell built-in in `bash`. It may have a different name in other shells.

SICStus will set the data segment size of itself according to the value of the system property (or environment variable) `SP_ULIMIT_DATA_SEGMENT_SIZE`. If you set this variable in the initialization file for your shell you do not have to use the `ulimit` command when SICStus is started from the shell. See Section "System Properties and Environment Variables" in *the SICStus Prolog Manual* for more information about `SP_ULIMIT_DATA_SEGMENT_SIZE`. This system property is set automatically when SICStus is invoked from the SICStus Prolog IDE (SPIDER), from Emacs (via *`M-x run-prolog`*), or from the launcher script '`SICStus Prolog 4.2.0.term`' installed in '`Applications`'.

File names are encoded in UTF-8 under Mac OS X. This is handled correctly by SICStus.

If SICStus encounters a file name that is not encoded in UTF-8, it will silently ignore the file or directory. This can happen on file systems where files have been created by some other OS than Mac OS X, e.g. on network file servers accessed by other UNIX flavors or Windows.

The default character encoding for the SICStus standard streams is based on the current locale which is POSIX/C, i.e. US ASCII, by default on Mac OS X. This will come in conflict with the default character encoding for the Terminal application which is UTF-8. A clickable launcher for SICStus is optionally installed in the Applications folder. This launcher will set the character encoding of the standard streams to UTF-8 for both the Terminal and SICStus.

4 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 4.2.0\`'). Here, `%SP_PATH%` is just a place-holder; 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 4.2.0\bin;%PATH%
```

To use `splfr` and `spld`, you must also set up the appropriate Microsoft Visual Studio tools; see Section "Setting up the C compiler on Windows" in *the SICStus Prolog Manual* for details.

To use the respective library modules, you must also include the paths to Tcl/Tk (see Chapter 6 [Tcl/Tk Notes], page 12) and Berkeley DB (see Chapter 8 [Berkeley DB Notes], page 18) onto the `PATH` environment variable if the installer for Berkeley DB and Tcl/Tk have not done so already.

4.1 Requirements

- Operating environment: Microsoft Windows XP SP, Vista SP2 or Windows 7 (including x64 but not IA64 versions). Windows Vista or later is recommended.
- For interfacing with C or C++, or for using `spld` or `splfr`: C compiler and related tools from Microsoft Visual Studio 2005 SP1 (a.k.a. VS 8). There is a separate release of SICStus for use with Microsoft Visual Studion 2008 SP1 (a.k.a. VS 9).

  Microsoft offers free editions of its C compilers. It is probably possible to make these work as well but they may require other tools or downloads.

4.2 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 SPIDER or Emacs.
2. A windowed executable providing command line editing and menus. See Section 4.4 [Command Line Editing], page 7.

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 decyption key, when started. This decryption key 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 by starting SICStus **with Administrative rights** from the Start menu (`spwin.exe`) and selecting `Enter License` from the `Settings` menu. Entering the license requires Administrative rights. Running SICStus should be possible from a limited account.

4.3 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.
- 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 4.2.0\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/.
- 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, 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.

4.4 Command Line Editing

Command line editing supporting Emacs-like commands and IBM PC arrow keys is provided in the windowed executable (`spwin.exe`). 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 '`~/spcmd4.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 25.

`save` *Value* is either 0 (don't save or restore history buffer) or 1 (save history buffer in '`~/spcmd4.hst`' on exit, restore history from the same file on start-up).

4.5 The Console Window

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

The console comes with a 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. Eventually The SICStus Prolog IDE (see Chapter 10 [The SICStus Prolog IDE], page 21) will replace the console.

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.

4.5.1 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 the following:

`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.

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

4.6 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 '`\\`'.
- All file names and paths are normalized when expanded by `absolute_file_name/3`. This is to simulate the case insensitivity used by Windows file systems. This means that files created by SICStus may have names on disk that differs in case from what was specified when the file was created.
- Emacs Issues: Running under Emacs has been tried with recent versions of GNU Emacs and XEmacs. See Chapter 11 [The Emacs Interface], page 22.
  - 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 atom `end_of_file` followed by a period.
- Under Windows, `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.

5 Managing Extended Runtime License Information

Extended runtime systems need to have a license available at runtime. This license can be embedded in the extended runtime executable or located in a separate file. The following describes the steps needed in order to enter the license information. The example assumes that you are familiar with the procedure for building runtime systems. See Section "The Application Builder" in *the SICStus Prolog Manual* for details.

Suppose that you have been provided with the following license information:

```
Your license information for platform
'extended_runtime_sicstus4.0_x86-win32-nt-4' is as follows:

Site name:          MySite
License code:       a111-b222-c333-d444-e444
Expiration date:    permanent
```

Following is a list of common tasks.

- **Making the license available to the development system.**

  Create a file '`extended_license.pl`' containing the following:

  ```
  %% LICENSE BEGIN
  site('MySite').
  product('extended_runtime_sicstus4.0_x86-win32-nt-4',
          'permanent',
          'a111-b222-c333-d444-e444').
  %% LICENSE END
  ```

  This file can be located anywhere, e.g. in the folder containing your source code.

- **Building an Extended Runtime System using `spld` which embeds license information from the above file:**

  ```
  % spld -E --license-file ./extended_license.pl [...]
  ```

  This will read the license information and embed the information in the created executable. No separate license file will be needed at runtime. This is the preferred method. This method can be used also to create an all-in-one executable; see Section "All-in-one Executables" in *the SICStus Prolog Manual*.

  On UNIX platforms, it is possible to install the license information using the `splm` tool so that you do not need create the file '`extended_license.pl`' and pass it to `spld`. However, a separate license file may be needed anyway if the license cannot be embedded; see below.

- **Building an Extended Runtime System which does not embed license information.**

  The resulting runtime system will need a way to find the license file at runtime. This variant is useful when the executable is not built with `spld`, e.g. when builing a DLL (Windows) or DSO (UNIX):

  ```
  % spld -E --no-embed-license [...]
  ```

The resulting executable will produce output similar to the following if it cannot find the license file:

```
License error:
License file not found! [...]
```

- **Ensuring that the license information is available at runtime.**

  If the license information has been embedded, no special steps are needed. Otherwise, you need to distribute the license file along with the runtime system. To make the license file available it should be located in the '`library`' folder and named '`license.pl`'. That is, copy '`extended_license.pl`' as created above into the file '`library/license.pl`' in the folder tree available at runtime. See Section "Runtime Systems on Target Machines" in *the SICStus Prolog Manual* for details. Also see Section "Locating the License Information" in *the SICStus Prolog Manual* for additional ways of making the license information available.

- **Understanding steps performed by `spld`.** As usual, you can use:

  ```
  % spld --verbose --keep [...]
  ```

  in order to see exactly what steps are performed by `spld`. This is useful if you want to embed the license but need to build the executable manually, instead of using `spld`.

6 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`

A better alternative may be to use one of the free installers available from:

`http://www.activestate.com`

SICStus for Mac OS X uses Aqua Tcl/Tk. The Aqua version of Tcl/Tk uses the native Aqua user interface. Mac OS 10.4 and later includes Aqua Tcl/Tk.

The Tcl/Tk interface module included in SICStus Prolog 4.2.0 (`library(tcltk)`) is verified to work with Tcl/Tk 8.4, and with Tcl/Tk 8.5 for some platforms. See the SICStus download web page, `http://www.sics.se/isl/sicstuswww/site/download4.html`, for details.

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 8.4.

Under Windows, the binary distribution is compiled against Tcl/Tk 8.4.

> **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.

7 Jasper Notes

7.1 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 supported for Java 1.5 or later.

For some platforms, Jasper is *only* supported under the following conditions:

Mac OS X Using Jasper from Java may require that `DYLD_LIBRARY_PATH` be set up so that Java can find the SICStus runtime library. That is, you may need to set `DYLD_LIBRARY_PATH` to the location of the SICStus runtime `libsprt4-2-0.dylib`.

Note that Java 1.6 is not available in a 32-bit version on all versions of Mac OS X.

7.2 Getting Started

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

7.2.1 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 4.2.0\bin`. For Java 1.5, it is usually '`C:\Program Files\Java\jdk1.5.0_15\jre\bin\client`'.

For example:

```
C:\> set PATH="C:\Program Files\Java\jdk1.5.0_15\jre\bin\client;%PATH%"
C:\> set PATH="C:\Program Files\SICStus Prolog 4.2.0\bin;%PATH%"
```

To make this change permanent under Windows XP, you would use the '`Advanced`' tab in the '`System`' Control Panel. Consult your OS documentation for details.

7.2.2 UNIX

When `library(jasper)` is used to embed Java in a SICStus development system or runtime system, the runtime 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 runtime 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 1.5 server version instead of the default (client) version.

7.2.3 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 runtime 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 runtime system. For this reason, it is not possible to use `library(jasper)` from an executable that links statically with the SICStus runtime.

7.2.4 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 Section "CPL Notes" in *the SICStus Prolog Manual*). 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")`. Under UNIX, Jasper can usually figure this out by itself, but in the event that Jasper is used in a non-standard installation, this will most likely fail. 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/sicstus4.2.0
/lib [...]
```

Under Windows, Java must be able to find '`spnative.dll`' through the `PATH` environment variable (see Section 7.2.1 [Windows], page 13). Setting '`-Djava.library.path`' under Windows can lead to problems if multiple versions of SICStus have been installed.

If this works properly, SICStus should have been loaded into the JVM address space.

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/sicstus4.2.0
/lib" \
       -classpath "/usr/local/sicstus4.2.0
/lib/sicstus-4.2.0/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
4.2.0/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, the SICStus runtimes named e.g. '`sprt4-2-0_instance_01_.dll`' need to be available as well. See Section "Runtime Systems on Target Machines" in *the SICStus Prolog Manual*.

7.3 Jasper Package Options

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

`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, 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)
```

7.4 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 7.6 [Known Bugs and Limitations], page 16 for details on the limitations of multi threaded Jasper.

7.5 Changes in Jasper from SICStus 3

- The (deprecated) predicates `jasper_call_static/6` and `jasper_call_instance/6` have been removed.
- SICStus 4 uses ISO syntax. This may affect Java code that handles Prolog terms.

7.6 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.
- 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*.
- Loading multiple SICStus runtimes has not been very well tested with multi threaded Jasper.

7.7 Java Examples Directory

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

7.8 Resources

There are almost infinitely many Java resources on the Internet. One good starting point is http://java.sun.com/.

8 Berkeley DB Notes

`library(bdb)` is built on top of Berkeley DB. It is an optional component of SICStus and nothing, except `library(bdb)`, will be affected if Berkeley DB is not installed on the machine.

Berkeley DB can be downloaded from:

http://www.oracle.com/database/berkeley-db/

`library(bdb)` is built using version 4.8.24 or some later version of 4.8.x. It may be possible to recompile it to work with other versions as well.

8.1 Berkeley DB on MS Windows

When installing Berkeley DB on Windows you should use the binary installer available from Oracle.

When using Berkeley DB under Windows, you should set the `PATH` environment variable to contain the path to '`libdb48.dll`'. Consult the Berkeley DB documentation for further info.

8.2 Berkeley DB on Mac OS X

The pre-built installation of SICStus for Mac OS X assumes that Berkeley DB is installed in the default location '`/usr/local/BerkeleyDB.4.8/`'.

There is no binary installer available for installing Berkeley DB 4.8 on Mac OS X. Instead you need to build it yourself. To do this you need to download and install the Apple C compiler (XCode) and then download, build and install the Berkeley DB library, using something like the following in the Terminal program:

1. Unpack the downloaded source code archive. At the time of writing, 4.8.30 is the latest version of Berkeley DB 4.8.

   ```
   tar xzf db-4.8.30.tar.gz
   ```

2. Move into the source code folder tree

   ```
   cd db-4.8.30/build_unix/
   ```

3. Configure Berkeley DB for both 32-bit and 64-bit

   ```
   ../dist/configure LDFLAGS='-arch x86_64 -arch i386' CFLAGS='-arch x86_64
   -arch i386'
   ```

4. Build

   ```
   make
   ```

5. Install in the standard location

   ```
   sudo make install
   ```

   The installation step requires that you are running as an administrator and the `sudo` command will require that you provide the password.

6. Verify that Berkeley DB was installed

   `/usr/local/BerkeleyDB.4.8/db_archive -V`

   This should print some version information, e.g. '`Berkeley DB 4.8.30: (April 9, 2010)`'.

9 ODBC Notes

ODBC (Open Database Connectivity) is a standard API for using a DBMS (DataBase Management System). By using ODBC you can access data from a multitude of DBMSs without having to know the details of each DBMS.

`library(odbc)` appeared in release 4.1.0 and we expect it to evolve and improve as we receive feedback from users. While we generally strive for backward compatibility we may have to make incompatible changes to `library(odbc)` in order to accommodate necessary improvements.

`library(odbc)` is currently supported on MS Windows, Mac OS X and Linux.

9.1 ODBC on MS Windows

ODBC is a standard component of MS Windows. You only need to install the DBMS specific ODBC drivers. Please refer to the ODBC documentation for MS Windows, and the ODBC documentation of your DBMS vendor.

9.2 ODBC on Mac OS X

ODBC is a standard component of Mac OS X. You only need to install the DBMS specific ODBC drivers. Please refer to the ODBC documentation for Mac OS X, and the ODBC documentation of your DBMS vendor.

The ODBC library, `library(odbc)`, crashes on Mac OS X 10.5. The underlying cause seems to be bugs in the ODBC software (iODBC 3.52.5) that comes with Mac OS X 10.5. The version of ODBC (iODBC 3.52.6) that comes with Mac OS X 10.6 does not seem to have this problem. It may be possible to obtain a newer version of the ODBC software at `http://www.iodbc.org/`.

9.3 ODBC on Linux

`library(odbc)` was built with unixODBC. unixODBC is an installable package on many Linux distributions, and can also be downloaded from `http://www.unixodbc.org`. You will also need to install the DBMS specific ODBC drivers.

10 The SICStus Prolog IDE (SPIDER)

SICStus Prolog IDE, also known as SPIDER, is an Eclipse-based development environment for SICStus with many powerful features. SPIDER is meant to eventually replace the Emacs interface and the Windows `spwin.exe` program as the main development environment for SICStus Prolog. SPIDER was initially made available with release 4.1.0. See Section "SICStus Prolog IDE" in *the SICStus Prolog Manual* for more information and links.

11 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 21.2. 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/software/emacs/` and `http://www.xemacs.org`, respectively.

11.1 Installation

The Emacs interface is distributed with SICStus and installed by default. The default installation location for the Emacs files is '`<prefix>/lib/sicstus-4.2.0/emacs/`' on UNIX platforms and '`C:\Program Files\SICStus Prolog 4.2.0\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*:

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.

11.1.1 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, Emacs should be able to find the SICStus documentation automatically; see Section "Installation" in *the SICStus Prolog Manual*:

12 Revision History

This chapter summarizes the changes in release 4 wrt. previous SICStus Prolog releases as well as changes introduced by patch releases.

12.1 What Is New In Release 4

12.1.1 Virtual Machine

- The internal representation of Prolog terms and code has been redesigned, resulting in code that runs up to twice as fast as in release 3.
- Certain memory limitations that existed in release 3 have been dropped. All available virtual memory can be used without any limitations imposed by SICStus Prolog.
- The limitations on "temporary" and "permanent" variables for compiled clauses have been dropped. There is no size limit on compiled clauses.
- The number of available atoms is four times larger than in release 3 (1M atoms are available on 32-bit platforms).
- The range of small integers is eight times larger than in release 3. Although the size of integers is unbounded, small integers are handled more efficiently than other numbers.
- Multifile predicates are compiled by default; in release 3, they could not be compiled.
- Native code compilation has been dropped.
- Execution profiling is available for compiled as well as interpreted code. The profiling data accessible by `profile_data/1` and `library(gauge)` is more precise. Some of the choices of release 3 have been dropped.
- Execution profiling has been generalized to support coverage analysis for compiled as well as interpreted code.

12.1.2 Prolog Language

12.1.2.1 Single Language Mode

Release 3 had the notion of multiple language modes: `iso` and `sicstus`. Release 4 does not have this notion. The syntax and semantics of the Prolog language correspond to the previous `iso` language mode.

12.1.2.2 DCG Notation

The exact rules for translating DCG rules to plain Prolog clauses have not been laid down in a standard, but there is a broad consensus in the Prolog community about what they should mean. One of the guiding principles is that the translation should be steadfast, in particular that the translated code should always treat its last argument as an output argument and not use it "too early". In some cases, a non-steadfast translation was produced in release 3. This has been corrected in release 4.

12.1.2.3 Asserting Terms with Attributed Variables

In release 3, terms containing attributed variables and blocked goals could be asserted, copied, gathered as solutions to `findall/3` and friends, and raised as exceptions. The copy would contain new attributed variables with the attributes copied. This operation could be

very expensive, could yield unexpected results and was not always safe e.g. in the context of CLPFD constraints. In release 4, the semantics of this operation has changed: in the copy, an attributed variable is simply replaced by a plain, brand new variable. Of course, if the same attributed variable occurs more than once, the same plain variable will occur in the corresponding places in the copy. If the attributes are relevant, the program can obtain them by using the new built-in predicate `copy_term/3` described below.

12.1.2.4 Arithmetic

The infix operator '`#`' (bitwise exclusive or) has been renamed to '`\`'.

12.1.2.5 Syntax

Atoms can now contain the NUL character, i.e. character code zero. It is classified as white space and must therefore be entered using escapes. As an example `'a\0\a'` is a three character atom containing two `a`s separated by a NUL.

Internally, atom names and other encoded strings, use the non-shortest form '`0xC0 0x80`' to encode NUL. This is similar to how NUL is handled by Tcl/Tk and Java.

12.1.2.6 Prolog Flags

The `language` and `wcx` Prolog flag have been dropped. The `profiledcode` value of the `compiling` Prolog flag has been dropped. Several new Prolog flags have been added. See Section "Prolog Flags" in *the SICStus Prolog Manual*.

12.1.2.7 Stream Properties

The `wcx` property has been dropped. Several new stream properties have been added. See Section "`stream_property/2`" in *the SICStus Prolog Manual*.

12.1.2.8 Statistics Keywords

Several new statistics keywords have been added. See Section "`statistics/[0,1]`" in *the SICStus Prolog Manual*.

12.1.2.9 Built-In Predicates

The set of built-in predicates has changed slightly. The following predicates have been removed:

`'C'/3`
: This was used in the Prolog translation of DCG rules. It could trivially be replaced by unifications and served no other reasonable purpose.

`get0/[1,2]`
`put/[1,2]`
: These used to have an overloaded semantics meaning one thing on binary streams and another thing on text streams. They have been subsumed by their ISO counterparts.

`get/[1,2]`
`tab/[1,2]`
`skip/[1,2]`
: Although these do not have ISO counterparts, they have been removed for being in the spirit of `get0/[1,2]` and `put/[1,2]`. We have provided `skip_char/[1,2]`, `skip_code/[1,2]`, and `skip_byte/[1,2]` as an ISO style replacement for `skip/[1,2]`.

`ttyget0/1`
`ttyget/1`
`ttynl/0`
`ttyput/1`
`ttyskip/1`
`ttytab/1`
`ttyflush/0`
: These used to exist as shorthands for the respective predicate with an additional `user` argument. In most cases, the "respective predicate" is one of the non-ISO style predicate mentioned above, so there was no point in keeping the shorthand.

`fileerrors/0`
`nofileerrors/0`
: These used to exist as shorthands for `set_prolog_flag/2` with specific arguments, and so can be trivially replaced.

`call_residue/2`
: Dropped because it was not possible to ensure the correct behavior in all circumstances, it relied heavily on copying terms with attributed variables, and it was not needed by any library module. It has been replaced by a similar predicate, `call_residue_vars/2`, which should suffice in most cases where `call_residue/2` was used; see below.

`undo/1` Dropped because it was not possible to ensure the correct behavior in all circumstances. Users that know what they are doing can still call the unsupported predicate `prolog:undo/1`. The argument should have a module prefix.

`help/0`
`version/0`
`version/1`
: These predicates, managing and displaying messages, can be easily emulated by feaures of the message system.

`fcompile/1`
`load/1` These predicates used to compile Prolog source code into '`.ql`' files, and load such files. '`.ql`' files serve a purpose when boot-strapping the Prolog system, but offer no advantages over '`.po`' files, the Prolog object code format used by other built-in predicates.

`load_foreign_files/2`
: This predicate provided a shorthand for building and loading a temporary foreign resource. Working with foreign resources is straightforward, and so the shorthand was dropped.

`require/1`
: This predicate provided a shorthand for locating and loading library predicates. This was originally introduced for a compatibility reason that is now obsolete. It is straightforward to provide the necessary `:- use_module/2` directives, and so the shorthand was dropped.

`profile_data/4`
`profile_reset/1`
: As of release 4.2, the execution profiling technology has been reengineered, eliminating the need to specially instrument code before it could be profiled. The new scheme also keeps track of the number of calls per caller-callee pair. Execution profiling is available for compiled as well as interpreted code. These two predicates have been replaced by a small number of new ones.

The following predicates have been added:

`call/`*N*
: Generalizes `call/1`. For example, `call(p(1,2), a, b)` is equivalent to `call(p(1,2, a, b))`.

`skip_char/[1,2]`
`skip_code/[1,2]`
`skip_byte/[1,2]`
: ISO style replacements for the non-ISO style `skip/[1,2]`.

`call_residue_vars/2`
: Called as follows:

```
call_residue_vars(:Goal, -Vars)
```

Executes the procedure call *Goal*, unifying *Vars* with the list of residual variables that have blocked goals or attributes attached to them. **Please note:** behaves differently from `call_residue/2` of release 3.

`copy_term/3`
: Called as follows:

```
copy_term(+Term, -Copy, -Body)
```

Unifies *Copy* with a copy of *Term* in which all variables have been replaced by brand new variables, and all mutables by brand new mutables. If *Term* contains variables with goals blocked on them, or variables with attributes that can be interpreted as a goal (see Section "`library(atts)`" in *the SICStus Prolog Manual*), then *Body* is unified with the conjunction of such goals. If no such goals are present, *Body* is unified with the atom `true`. The idea is that executing *Body* will reinstate blocked goals and attributes on the variables in *Copy* equivalent to those on the variables in *Term*.

`profile_reset/0`
`profile_data/1`
`print_profile/[0,1]`
`coverage_data/1`
`print_coverage/[0,1]`
: As of release 4.2, the execution profiling technology has been reengineered, eliminating the need to specially instrument code before it could be profiled.

Execution profiling is available for compiled as well as interpreted code. It has been generalized to support coverage analysis for compiled as well as interpreted code. The new scheme also keeps track of the number of calls per caller-callee pair. These are the relevant new built-in predicates.

Some predicates have been changed slightly; in most cases, this affects predicates that take a list of options:

`[`*F1*`,`*F2*`,...]`
: This is now a short-hand for `load_files([`*F1*`,`*F2*`,...])`.

`is_mutable/1`
: The predicate `is_mutable/1` has been renamed to `mutable/1`, in analogy with `integer/1`, `atom/1` etc.

`module/1`
: The predicate `module/1` has been renamed to `set_module/1`, to avoid possible confusion with the `module/2` declaration.

`format/[2,3]`
: For the predicate `format/[2,3]`, the semantics of the '`~@`' spec has changed slightly: the goal *Arg* is called as if by `\+ \+` *`Arg`*, i.e. any bindings made by the goal are lost.

`close/2`
: Takes new options:

  `direction/1`
  : Specifies which directions to close.

`open/4`
: The `wcx/1` option has been dropped. Takes several new options. See Section "`open/4`" in *the SICStus Prolog Manual*.

`absolute_file_name/[2,3]`
: The `ignore_underscores/1` option has been dropped. The `file_type/1` option value `ql` has been dropped, whereas the option value `executable` is new. The `access/1` option values `execute`, `executable` and `search` are new. The `glob/1` option is new, allowing to match file names against a pattern. If available, the load context directory (`prolog_load_context/2`) will be used as default directory.

`load_files/2`
: The `load_type/1` option value `ql` has been dropped. `encoding_signature/1`, `encoding/1`, subsuming the `wcx/1` option of release 3, and `eol/1`, are new options, corresponding to the respective stream properties.

`write_term/3`
: The `quoted_charset/1` option is new, reflecting the value of the Prolog flag with the same name.

`halt/1`

The predicate `halt/1` now raises an internal exception like `halt/0`. This gives surrounding Prolog and C code an opportunity to perform cleanup.

`append/3`
`member/2`
`memberchk/2`

These are now built-in, they used to reside in `library(lists)`.

12.1.2.10 Hook Predicates

The hook `user:term_expansion/[2,4]` is replaced by the hook:

```
user:term_expansion(Term1, Layout1, Tokens,
                    Term2, Layout2, [Token|Tokens]).
```

The purpose of the new argument *Tokens* is to support multiple, independent expansion rules. The purpose of the arguments *Layout1* and *Layout2* is to support source-linked debugging of term-expanded code. Each expansion rule should have its unique identifying token *Token*.

The hook `user:goal_expansion/3` is replaced by the following per-module hook:

```
M:goal_expansion(Term1, Layout1,
                 Module, Term2, Layout2).
```

Typically, *Module* has imported the predicate *Term1* from module *M*. The purpose of the arguments *Layout1* and *Layout2* is to support source-linked debugging of goal-expanded code.

12.1.3 Library Modules

There is no consensus for a core library, portable across Prolog systems, let alone a standard for such a library. Since release 3, SICS has acquired Quintus Prolog, which has a rather rich library. For release 4, we have decided to make this asset be available to the SICStus community by providing a library that is a merger of the previous SICStus and Quintus libraries, which already overlap significantly.

The User's Manual documents the library of release 4. For the purposes of aiding code transition to release 4, the following is a list of the release 3 library modules, and their fate in release 4. See also Section 12.2 [Guide to Porting Code from Release 3], page 34.

`atts`
`comclient`
`fdbg`
`gauge`
`heaps`
`linda/client`
`linda/server`
`pillow`
`prologbeans`
`tcltk`
`timeout`
`trees`
`wgraphs`
`xml` As in release 3.

`arrays` The native release 4 counterpart is called `library(logarr)`. Also available is a deprecated compatibility module `library(arrays3)`.

`assoc` The native release 4 counterpart is called `library(avl)`, reflecting the abstract data type, AVL trees, and with a modified, richer API. Also available is a deprecated compatibility module `library(assoc3)`.

`bdb` As in release 3, but uses the default Berkeley DB hash function, so all of the standard Berkeley DB utilites should now work.

`charsio` Called `library(codesio)` in release 4. Likewise, the syllable '`chars`' has been renamed to '`codes`' in predicate names.

`clpb`
`clpq`
`clpr` As in release 3, unsupported.

`clpfd` As in release 3, plus the following additions and changes:

`automaton/8`
: is a new constraint capturing any constraint whose checker of ground instances can be expressed as a finite automaton.

`minimum/2`
`maximum/2`
: are new constraints, constraining a value to be the minimum (maximum) of a list of values.

`nvalue/2` is a new constraint, constraining the number of distinct values taken by a list of values.

`cumulative/[1,2]`
: provides a unified interface, subsuming `serialized/[2,3]` and `cumulative/[4,5]`.

`table/[2,3]`
: defines an n-ary constraint by extension, subsuming `relation/3`.

`all_different/[1,2]`
`all_distinct/[1,2]`
: Arguments can have unbounded domains.

`scalar_product/[4,5]`
: can optionally be told to maintain arc-consistency. This functionality subsumes `knapsack/3`.

`global_cardinality/[2,3]`
: can optionally be told to use a simple algorithm. This functionality subsumes `count/4`.

`fd_copy_term/3`
: is gone. Subsumed by built-in `copy_term/3`.

`jasper` The Jasper module is available in the current release. An alternative for Java users is PrologBeans. The latter is the recommended method for interfacing Java with SICStus. Jasper should only be used when PrologBeans is insufficient.

`lists` The native release 4 counterpart has a modified, richer API. Also available is a deprecated compatibility module `library(lists3)`.

`ordsets` As in release 3, plus several new predicates.

`queues` The native release 4 counterpart has a modified, richer API. Also available is a deprecated compatibility module `library(queues3)`.

`random` The native release 4 counterpart has a modified, richer API. Also available is a deprecated compatibility module `library(random3)`. **Please note:** The random number generator state is slightly different from the one in release 3.

`sockets` The new predicate `socket_client_open/3` subsumes `socket/2` and `socket_connect/3`.

`socket_server_open/[2,3]` subsumes `socket/2`, `socket_bind/2` and `socket_listen/2`.

`socket_select/7` can wait for any kind of stream, not just socket streams. `socket_select/7` waits until one *unit* (character for text streams, byte for binary streams) can be transferred.

`socket_select/7` can wait for streams ready to write.

`socket_select/7` does not create streams, you need to explicitly use `socket_server_accept/4`.

Socket streams are binary by default.

Blocking socket operations can be interrupted on both UNIX and Windows.

`library(sockets)` should work with IPv6 (in addition to IPv4 and `AF_UNIX`).

`system` Operations on files and directories have been moved to its own module, `library(file_systems)`. Process primitives have been redesigned and moved to a new module, `library(process)`. The predicates for creating temporary files, `mktemp/2` and `tmpnam/1`, have been removed. They used C library functionality that is broken by design and insecure. Instead, to create and open a temporary file use something like `open(temp('foo'), write,`

`S, [if_exists(generate_unique_name)])`, possibly together with `stream_property(S, file_name(Path))` if you need to know the path to the generated file name.

The (little) remaining functionality is largely as in release 3. Also available is a deprecated compatibility module `library(system3)`.

`terms` As in release 3, plus several new predicates. `term_hash/2` is not guaranteed to compute the same hash values as in release 3.

`ugraphs` As in release 3, plus a couple of deletions.

`objects` Replaced by the Quintus Prolog flavor of `library(objects)`.

`chr` A reimplementation of `library(chr)`, based on the Leuven implementation.

`flinkage`
`spaceout` Not present in release 4.

`vbsp` Not available in the current release. Visual Basic .NET and other .NET languages can use PrologBeans .NET.

The following is a list of library modules that are new in release 4.

`aggregate`
provides an aggregation operator for data-base-style queries.

`assoc` uses unbalanced binary trees to implement "association lists", i.e. extendible finite mappings from terms to terms.

`bags` defines operations on bags, or multisets

`between` provides some means of generating integers.

`file_systems`
accesses files and directories.

`objects` provides a package for object-oriented programming, and can be regarded as a high-level alternative to `library(structs)`.

`plunit` Unit test harness.

`process` Creating, killing, releasing, and waiting on processes.

`rem` provides Rem's algorithm for maintaining equivalence classes.

`samsort` provides generic sorting.

`sets` defines operations on sets represented as lists with the elements unordered.

`structs` provides access to C data structures, and can be regarded as a low-level alternative to `library(objects)`.

`types` Provides type checking.

`varnumbers`
An inverse of `numbervars/3`.

12.1.4 Input-Output System

The internals of the I/O subsystem have been completely redesigned. The new version should be faster while at the same time providing more functionality and more consistent behavior between operating systems and between stream types.

The semantics of character codes has been fixed as (a superset of) Unicode. Redefining the meaning of character codes is no longer supported.

New features and changes to the SICStus streams (`SP_stream`) include:

- Streams are binary or text also at the lowest level, e.g. in the C API, and there are separate operations for performing I/O of bytes and characters.
- Streams have a layered design. This makes it possible to add character set translation and other transformations (compression, encryption, automatic character set detection, ...) to any stream.
- All streams provide non-blocking operations and are interruptible, e.g. with `^C` ('`SIGINT`'). This is also true for file streams and under Windows.
- Subject to OS limitations, file names can use Unicode and be of arbitrary length. In particular, under Windows, the Unicode API is used for all operations.
- Limits on file size, file time stamps etc have been removed.
- Error handling has been simplified and made more consistent. In the C API all I/O operations return an error code from a rich set of error codes. Errors during write and close operations are no longer ignored.
- It is possible to wait for I/O ready (both for read and write) on any type of stream. This works for all platforms, including Windows. Select operations wait for the appropriate item type, e.g. until a whole (possibly multi-byte) character can be transferred on a text stream.

Other minor changes:

- Now `byte_count/2` can be called only on binary streams.
- `at_end_of_stream/[0,1]` never blocks. Instead it will fail, i.e. behave as if the stream is not at its end, if the operation would otherwise block. See Section "`at_end_of_stream/[0,1]`" in *the SICStus Prolog Manual*.

12.1.5 Foreign Language APIs

12.1.5.1 Foreign Language Interface

The conversion specifier (in `foreign/[2,3]` facts) `string(N)` has been dropped.

The conversion specifier `chars` has been renamed to `codes`, in analogy with the built-in predicate `atom_codes/2`, the second argument of which is a list of character codes.

The C header generated by `splfr` from the `foreign/[2,3]` facts now uses the `const` attribute where appopriate.

Foreign resources are no longer unloaded by `save_program/[1,2]`. For this reason, the deinit function of a foreign resource is no longer called when saving a program so `SP_WHEN_SAVE` has been removed.

12.1.5.2 C API Functions

Many functions in the C API has been changed or removed, especially those related to OS and I/O operations. There are also a number of new C API functions.

Old API	**Replaced by**
`SP_make_stream`, `SP_make_stream_context`	`SP_create_stream`
`SP_set_tty`	`SP_CREATE_STREAM_OPTION_INTERACTIVE`
`SP_fgetc`	`SP_get_byte`, `SP_get_code`
`SP_fputc`	`SP_put_byte`, `SP_put_code`
`SP_fputs`	`SP_put_codes`, `SP_put_encoded_string`
`SP_fflush`	`SP_flush_output`
`SP_chdir`	`SP_set_current_dir`
`SP_getcwd`	`SP_get_current_dir`
`SP_set_wcx_hooks`	Gone
`SP_wcx_getc`, `SP_wcx_putc`	Gone
`SP_to_os`, `SP_from_os`	Gone
`SP_put_number_chars`	`SP_put_number_codes`
`SP_get_number_chars`	`SP_get_number_codes`

Other new functions include:

```
SP_get_stream_user_data
SP_get_stream_counts
SP_put_bytes
SP_fopen
```

`SP_unget_code`
`SP_unget_byte`

Also, many functions take new or changed parameters.

12.1.5.3 Java API

- The PrologBeans API has been extensively revised. See the PrologBeans HTML (javadoc) documentation.
- PrologBeans was built with Java 1.5.

12.2 Guide to Porting Code from Release 3

Release 4 does not provide a mode in which it is 100% compatible with earlier releases. However, in addition to what is said in Section 12.1 [What Is New In Release 4], page 23 (read that first!), this section provides further guidelines for migrating Prolog code from release 3 to release 4.

1. First of all, make sure that your code runs in ISO execution mode. In release 3, the command line option '`--iso`' can be used.
2. A number of built-in predicates have been dropped. They are listed in the table below, along with their approximate substitutes. Refer to the documentation for each case.

Dropped built-in	**Replaced by**
`get0/[1,2]`, `get/[1,2]`	`get_code/[1,2]`, `get_byte/[1,2]`
`ttyget0/1`, `ttyget/1`	`get_code/2`, `get_byte/2`
`put/[1,2]`, `tab/[1,2]`	`put_code/[1,2]`, `put_byte/[1,2]`
`ttyput/1`, `ttytab/1`	`put_code/2`, `put_byte/2`
`skip/[1,2]`	`skip_code/[1,2]`, `skip_byte/[1,2]`
`ttyskip/1`	`skip_code/2`, `skip_byte/2`
`ttynl/0`	`nl/1`
`ttyflush/0`	`flush_output/1`
`fileerrors/0`, `nofileerrors/0`	`set_prolog_flag/2`
`'C'/3`	unification
`call_residue/2`	`call_residue_vars/2`
`undo/1`	`prolog:undo/1`

`help/0`	the message system
`version/0`	the message system
`version/1`	the message system
`fcompile/1`	`save_files/2`
`load/1`	`load_files/2`
`load_foreign_files/2`	`splfr` + `load_foreign_resource/1`
`require/1`	`use_module/2`
`is_mutable/1`	`mutable/1`
`module/1`	`set_module/1`

3. The hook predicates `user:term_expansion/[2,4]` and `user:term_expansion/3` are now called `user:term_expansion/6` and *`Module`*`:term_expansion/5` and have a modified API; see Section "Term and Goal Expansion" in *the SICStus Prolog Manual*.
4. The set of library modules has been enriched by incorporating a subset of the Quintus Prolog library modules that we have deemed useful.

   `library(clpb)`, `library(clpq)` and `library(clpr)` are provided but not supported. `library(flinkage)` and `library(spaceout)` are not included in release 4. `library(objects)` has been replaced by its Quintus counterpart, with a completely different API.

   The following table lists the affected SICStus 3 library modules.

Affected module	**Closest equivalent**	**Comment**
`arrays`	`arrays3`	a
`assoc`	`assoc3`	b
`charsio`	`codesio`	c
`clpfd`	`clpfd`	d
`lists`	`lists3`	e
`queues`	`queues3`	f
`random`	`random3`	g
`sockets`	`sockets`	d

`system`	`system3`	h
`terms`	`terms`	d

Comments to the table:

a. `library(arrays3)` is a code migration library module; the long-term solution is to use `library(logarrs)` instead.

b. `library(assoc3)` is a code migration library module; the long-term solution is to use `library(avl)` instead.

c. The syllable '`chars`' has been changed to '`codes`' throughout.

d. Several API changes; see the documentation.

e. `library(lists3)` is a code migration library module; the long-term solution is to use `library(lists)` instead.

f. `library(queues3)` is a code migration library module; the long-term solution is to use `library(queues)` instead.

g. `library(random3)` is a code migration library module; the long-term solution is to use `library(random)` instead.

h. `library(system3)` is a code migration library module; the long-term solution is to use `library(system)`, `library(file_systems)` and `library(process)` instead.

   One difference between `library(system3)` and the original release 3 version is that `exec/3` returns a process reference, a compound term, instead of an integer process identifier.

12.3 Limitations in the Current Release

This section lists features that are missing or incompletely implemented in the current release of SICStus Prolog (SICStus Prolog 4.2.0) but that may appear in future releases. Please let us know what features are important to you!

`library(tcltk)`: There is no way to pass non-Latin 1 characters from Tcl/Tk to Prolog. The Tcl/Tk Terminal is not supported.

`library(spaceout)`: not supported; see Section 12.1.3 [Library Modules], page 28.

The Visual Basic 6 module (`vbsp`) is not supported; see Section 12.1.3 [Library Modules], page 28.

The Windows GUI `spwin.exe` does not support full Unicode. The console version `sicstus.exe` fully supports Unicode when run from a console window or from within SPIDER or Emacs.

The Emacs mode may not work reliably when passing Prolog code between Emacs and SICStus if the code is not written using Latin 1.

12.4 Changes Introduced in Version 4.0.1

12.4.1 New Features

12.4.2 Bugs Fixed

- Spurious `SPIO_E_ERROR` exceptions when interrupting Prolog. Most often seen when using `library(timeout)` or when using `^C` at the top-level prompt.
- Inconsistent error messages if the license information was missing or incomplete.
- `library(fdbg)`: inconsistent trace messages for labeling steps.
- `library(clpfd)`: error handling for user-defined global constraint actions.
- Source info of interpreted clauses.
- Memory management issue with garbage collection + pending unblocked goals
- CHR debugging and tracing did not work.

12.4.3 Other Changes

- **Compatibility issue**: The two Latin 1 character codes `0x00AA` (`FEMININE ORDINAL INDICATOR`) and `0x00BA` (`MASCULINE ORDINAL INDICATOR`) are now classified as lower case letters by the Prolog parser. They used to be (incorrectly) classified as symbol chars. This may affect code that used any of these characters in unquoted atoms or functors.

  This change was made to align their classification with the Unicode standard.
- Quoted atoms and strings can now contain any character sequence from Unicode 5.0 when reading, with some restrictions; see Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.
- Quoted atoms and strings are now by default written using a larger subset of Unicode than before. See the documentation for the Prolog flag `quoted_charset` (see Section "Prolog Flags" in *the SICStus Prolog Manual*).
- Windows: All code is built with the security options '`/GS`', '`/SAFESEH`', '`/NXCOMPAT`'.
- Corrected the documentation for `SP_put_list_n_codes()`.
- Now UTF-8 is used when communicating with the SICStus Prolog sub-process in versions of Emacs and XEmacs that supports it.

12.4.4 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will allow some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

12.5 Changes Introduced in Version 4.0.2

12.5.1 New Features

- Added support for ISO-8859-2, a.k.a. Latin 2.
- `absolute_file_name/3`: new option `file_type(executable)` expands to `extensions(['', '.exe'])` on Windows and to `extensions([''])` on other systems.

12.5.2 Bugs Fixed

- Memory manager: efficiency bug.
- `library(structs)`: unsigned types, 64-bit issues.
- PrologBeans: Lists of integers with element values above 255 broke the communication between Java and SICStus.
- Closing a stream would sometimes hang due to a race condition on UNIX-like platforms. This was most likely to happen on MacOS X.
- `set_stream_position/2` and `seek/4` did not work on output streams.
- Multiple issues with `absolute_file_name/3`.
  - Option `file_errors(fail)` would sometimes report permission errors (`SPIO_E_PERMISSION_ERROR`) instead of silently failing.
  - Option `file_errors(fail)` now fails instead of raising an exception for file name domain errors like malformed file names and too many symbolic links (`SPIO_E_INVALID_NAME`).
  - Options `access(execute)` and `access(search)` now imply `access(exist)`. This is similar to how `access(read)` works.
  - The undocumented internal option `access(directory)` was allowed. Use `file_type(directory)` instead.
- `library(process)`: `process_create/[2,3]` now skips non-executable file and non-files if the *File*-argument can expand to more than one file. This is especially useful when using the symbolic name `path/1` to specify a file.
- `library(avl)`: Bug in `avl_delete/4`.
- `library(random)`: Document and check validity of the random number generator state. Bug in `random_numlist/4`.
- `get_atts/2`: Could fail incorrectly.
- `library(clpfd)`: A memory management problem. An integer overflow problem. Propagation bug in `case/[3,4]`, affecting `automaton/8` too.
- A problem with shared subterms in copying, asserting, collecting and throwing terms.
- The Prolog flag `title` was truncated by `spwin.exe` under Windows.
- The `spdet` utility did not automatically add '`.pl`' and '`.pro`' extensions to file name arguments.

12.5.3 Other Changes

- `library(clpfd)`: minor efficiency issues.
- The `user_error` stream is always unbuffered, even when not attached to a terminal.

- Improved detection of the '`executable`' file property under Windows, e.g. in `absolute_file_name/3` and `process_create/[2,3]`.
- The Prolog flag `title` is now saved by `set_prolog_flag(title, ...)` on all platforms. It used to be ignored except under Windows.

12.5.4 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will allow some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

12.6 Changes Introduced in Version 4.0.3

12.6.1 New Features

- The new Prolog flag `legacy_char_classification` makes it possible to use full Unicode, e.g. Chinese characters, in unquoted atoms and variable names. See Section "Prolog Flags" in *the SICStus Prolog Manual*.
- The Prolog flag `redefine_warnings` can take new values, and is no longer ignored in runtime systems. See Section "Prolog Flags" in *the SICStus Prolog Manual*.
- `SP_load_sicstus_run_time()`, and related functionality for loading multiple SICStus runtimes into a process, is now available.
- Jasper Java interface (`library(jasper)`) is now available. Jasper is mainly for legacy code; PrologBeans is still the preferred method of calling Prolog from Java.
- `library(sockets)` now supports UNIX domain (`AF_UNIX`) sockets on UNIX-like platforms. The new predicate `socket_server_open/3` allows some options when opening a server socket.
- `SP_set_argv()`, a new C API function for setting the values returned by the `argv` Prolog flag. Similar to the `argv` argument to `SP_initialize()`, but can report failure and can use locale information.
- `spld` and `splfr`: new command line options. The new (POSIX) option '`--`' is treated the same as the older '`-LD`'. New option '`--conf` *`VAR=VAL`*' to override variable *VAR* in the configuration file. Option processing has been rewritten to be more robust and consistent. See Section "The Application Builder" in *the SICStus Prolog Manual* and Section "The Foreign Resource Linker" in *the SICStus Prolog Manual*.
- `sicstus` The new (POSIX) option '`--`' is a synonym for the old '`-a`'.

12.6.2 Bugs Fixed

- `trimcore/0` could lead to memory corruption.

- `append/3` "optimization" could cause garbage collector crash.
- `spld` and `splfr`: multiple '`--cflag`' options accumulate, as documented.
- `sockets:current_host/1` would fail on Windows 2000 with some network configurations.
- `process:process_release/1` did not work.
- All process creation routines in `library(system3)` now work when there are command line options in the command argument, as was intended.
- `file_systems:current_directory/2` was sensitive to load context when passed a relative path as its second argument.
- The Windows GUI `spwin.exe` command '`Save Transcript`' now works and uses UTF-16 with BOM which can be read by most Windows programs and by recent Emacs and XEmacs.
- The menu commands of the Windows GUI `spwin.exe` no longer load foreign resources. This prevents extra foreign resources from being recorded by `save_program/[1,2]`.
- `library(chr)`
  - Multiple occurrences of the same answer constraint are no longer suppressed.
  - Error in compile-time error message.
- `library(clpfd)`
  - `element/3` and `cumulatives/[2,3]` could crash.
  - Bug in `dom(X)+dom(Y)` in indexicals.
  - Structure sharing issues with `fd_set/2` and `in_set/2` in the global constraint API.
  - `mod` and `rem` are now available with the intended semantics.
  - Incorrect reification of arithmetic relations involving division, `mod` and `rem`.
- Variables not transferred correctly in the PrologBeans process communication protocol.

12.6.3 Other Changes

- Output to different interactive output streams, like `user_output` and `user_error`, are now properly ordered.
- If the standard OS streams cannot be used, the SICStus runtime will use null streams instead of failing initialization. Happened when started from recent Linux `nohup` command.
- Under UNIX, `sicstus` now interprets command line arguments using locale information (the Windows version already did this).
- Saved states invoked as shell scripts will now use a version specific name for the sicstus executable, e.g., `exec sicstus-4.0.3 ...` instead of `exec sicstus ...`.
- The `spld` tool now ignores the `--more-memory` option and no longer attempts to use a modified linker script on x86 Linux.
- The `splfr` tool no longer uses a fixed name for some temporary files, which prevented parallel make.

12.6.4 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will allow some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

12.7 Changes Introduced in Version 4.0.4

12.7.1 New Features

12.7.2 Bugs Fixed

- On Windows the result of `absolute_file_name/[2,3]` would contain backslash instead of forward slash if the absolute file name contained certain non-ASCII characters. This bug also broke all directory listing functions in `library(file_systems)`, e.g. `file_systems:file_member_of_directory/[2,3,4]`.
- A change in 4.0.3 caused `system3:popen/3`, `system3:shell/[1,2]` and `system3:system/[1,2]` to no longer work when the command string contains redirection and other special constructs. These predicates now always invoke the system shell.
- A change in 4.0.3 caused `library(sockets)` to not accept a lone port number as an address. A port number *`Port`* is now treated the same as `inet('', Port)`, as in earlier releases. This also broke `prologbeans:start/[0,1]` when no port was specified.
- A few operators had non-ISO mode operator declarations. This has been corrected to match the documentation, the ISO Prolog standard and the ISO language mode in SICStus Prolog 3. See Section "Built-in Operators" in *the SICStus Prolog Manual*.

  Please note: This is an incompatible change that may cause a Prolog program or data to be parsed differently (or not at all). However, in practice we expect this to affect little or no code. Data written using `write_canonical/[1,2]` or similar will not be affected and will be read back correctly regardless of operator declarations.

  To preserve the old, incorrect, operator declarations, insert the following at the top of your Prolog files:

  ```
  :- op( 500,  fx,[+,-]).
  :- op( 300, xfx,[mod,rem]).
  ```

  To ensure that the new, correct, operator declarations are in effect also in releases predating 4.0.4, insert the following at the top of your Prolog files (**please note**: this documentation was updated after 4.0.4 to correct the associativity of `+`, `-`):

  ```
  :- op( 200,  fy,[+,-]).
  :- op( 400, yfx,[mod,rem]).
  ```

12.7.3 Other Changes

12.7.4 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will allow some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

12.8 Changes Introduced in Version 4.0.5

12.8.1 New Features

- `library(zinc)`: Interpreters for the MiniZinc and FlatZinc combinatorial problem modeling languages being developed in the G12 project; see `http://www.g12.csse.unimelb.edu.au`.
- `library(clpfd)`: Revived deprecated constraints `count/4` and `relation/3`.

12.8.2 Bugs Fixed

- `open/[3,4]` error handling.
- Critical virtual machine bugs in floating point arithmetic.
- Garbage collection now runs in constant space.
- Opening a UNIX fifo, or other non-seekable file, in text mode would not terminate until the other end closed the connection and then it would report a seek error. The problem was with the read-ahead needed to detect character encoding. Non-seekable files are now opened as if `encoding_signature(false)` were passed to `open/4`.
- Runtime systems generated by `spld` did not propagate exit code from `halt/1`.
- `statistics/0` would sometimes report incorrect, including negative, "program space breakdown" for the "miscellaneous" and "interpreted code" categories.
- `SP_event()` handlers are no longer allowed to run during `SP_exception_term()` or `SP_deinitialize()`.

  `SP_exception_term()` calls Prolog code which could allow `SP_event()` handlers to run, e.g. for `library(timeout)`. In this case exceptions and failures from an `SP_event()` handler would be ignored and possibly confuse `SP_exception_term()`.

  `SP_deinitialize()` does some cleanup by calling Prolog code. This can no longer cause `SP_event()` handlers to run.
- Fixed a memory corruption issue that happened during exception handling.
- Prologbeans:
  - Lists of one character atoms were incorrectly transferred from Java to SICStus.

  - The example `sessionsum` was missing the line:

    ```
    pSession.connect();
    ```

    after the declaration of `pSession`.
  - Session listeners were not notified when a client closed the stream.
- Jasper: A memory leak in multithread mode.
- Debugger:

  A `file/[1,2]` breakpoint test or action would raise an exception when used with uninstantiated first (file name) argument.
  - Some conditional breakpoints could not be handled by `SU_messages` message processing. This sometimes caused a raw message term to be presented in the debugger.
  - Sometimes breakpoint tests were evaluated with the wrong value for the `bid/1` breakpoint condition. The `bid/1` breakpoint condition was not always reset to `bid(off)` when no breakpoint was selected. The documentation was updated to correctly say `bid(off)` instead of `bid(none)`.
- Sometimes, SICStus would enter an infinite loop if the error stream was closed in the other read-end. This could happen, e.g. when SICStus was invoked as a subprocess and the parent process exited ungracefully.
- Some Prolog code would not compile in `profiledcode` mode.
- Spurious type errors in several library modules.
- `library(objects)`, `library(structs)`: fixed a 64-bit issue, and putting integers now checks for overflows.
- `library(avl)`: bug in `avl_max/3`.
- `library(clpfd)`: bug fixes for `circuit/1`, `table/[2,3]`, `lex_chain/[1,2]`, `#\=`.
- `library(bdb)`:
  - `db_open/5` could crash if the option `cache_size/1` was passed.
  - Very long filenames could cause crashes.
  - Did not work reliably with non-ASCII file names.
  - `db_enumerate/3`, `db_sync/1`, `db_make_iterator/2`, `db_iterator_next/3` and `db_iterator_done/1` crashed if called after the database had been closed.

12.8.3 Other Changes

- The windowed executable (`spwin.exe`) on Windows now saves and reads the command history (see Section 4.4 [Command Line Editing], page 7).
- `write/[1,2]` is now much faster when writing atomic terms.
- `assertz/1` and friends are now faster when asserting facts, i.e. clauses without bodies.
- `library(terms)`: the new predicate `term_hash/3` allows more control over the hashing behavior and hash algorithm used.

  Notable new features: a new, better, default hash algorithm and several other algorithms, including the 4.0.4 version, are available; it is possible to obtain a full 32-bit hash value; it is possible to get an instantiation error or hash value when the term being hashed is nonground.

`term_hash/[2,4]` has been changed to use a better hash function by default. The new hash function gives fewer collisions in general, and gives the same value on all platforms.

Please note: The change of hash function is an incompatible change that may affect programs or data that depend on the old hash algorithm. The old behavior can be obtained as follows:

```
%% Pre 4.0.5 version
term_hash_4_0_4(Term, Hash) :-
   term_hash(Term, [algorithm('sicstus-4.0.4')], Hash).

term_hash_4_0_4(Term, Depth, Range, Value) :-
   term_hash(Term, [algorithm('sicstus-4.0.4'), depth(Depth), range(Range)], Hash)
```

- `library(debugger_examples)` updated.
- Extended Runtime systems (a separate product, adding the compiler to runtime systems) now require a license at runtime. By default `spld` will embed the license into the executable.
- The hook `user:error_exception/1` is now called with the exception term specified by ISO Prolog, i.e. the same term that is seen by `catch/3` and `on_exception/3`. It used to be called with an internal representation of the exception. This affects error exceptions, i.e. those with functor `error/2`. The old (pre 4.0.5) value passed to `user:error_exception/1` is the second argument of the `error/2` structure.

  Please note: This is an incompatible change. Old code that uses `user:error_exception/1` may need to be updated. If the old code looked like:

  ```
  %% Pre 4.0.5 version
  user:error_exception(Old) :- do_something(Old).
  ```

  it can be rewritten as follows (which will also work in older versions of SICStus Prolog):

  ```
  %% >= 4.0.5 version
  user:error_exception(New) :-
     ( New = error(_, Old) -> true; Old = New),
     do_something(Old).
  ```

- `trimcore/0` is now more thorough when releasing memory back to the operating system. This also affects the `trimcore`-variant used by the top-level.
- It is now possible to tell SICStus to use `malloc()` et al. as memory manager instead of the default custom allocator.

  `malloc()` is selected when starting `sicstus` with the new option '`-m`'; when initializing the SICStus runtime with the environment variable `SP_USE_MALLOC` set to `yes`; for SICStus runtimes built with the new `spld` option '`--memhook=malloc`'; and when calling `SP_set_memalloc_hooks()` with the new option `SP_SET_MEMALLOC_HOOKS_HINT_USE_MALLOC`. See Section "SP_set_memalloc_hooks" in *the SICStus Prolog Manual*.
- `library(clpfd)`: unification with domain variables as well as propositional combinations of arithmetic constraints have been accelerated.

12.8.4 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- Exceptions in interpreted code will not get accurate source info in the source linked debugger.
- Saved-states and '`.po`' files are not portable across architectures that have the same word size, which they should be. This will be fixed in release 4.1.0.
- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will accept some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

12.9 Changes Introduced in Version 4.0.6

This section is intentionally left empty. Version 4.0.6 was skipped in the release series.

12.10 Changes Introduced in Version 4.0.7

12.10.1 New Features

- Most text streams can now be opened with `reposition(true)`, allowing `set_stream_position/2` and `seek/4`. This works for fixed-width, single-byte encodings. This includes Latin 1 and similar encodings provided *`LFD`* is used for end-of-line. See Section "open" in *the SICStus Prolog Manual*.
- `library(clpb)` is revived from SICStus 3, unsupported.

12.10.2 Bugs Fixed

- `SP_event()` handlers are no longer allowed to run during `SP_fclose()`, which sometimes needs to perform some cleanup by calling Prolog code. This can no longer cause `SP_event()` handlers to run.
- Exceptions during exception handling would cause the top-level to exit.
- `SP_event()` handlers were not always called during event handling. One symptom was that, at least on Windows, `timeout:time_out/3` could not always interrupt a goal called from an event handler.
- Bug in redefining multifile predicates.
- `sockets:socket_select/7` leaked memory on Windows.
- `library(queues)`: bug in `portray_queue/1`.
- `library(clpfd)`: Incorrect reification and efficiency bugs in arithmetic relations involving division, `mod` and `rem`; incorrect handling of `inf` and `sup` in `table/[2,3]`.

12.10.3 Other Changes

- Foreign resources compiled with releases predating 4.0.5 will not load into newer releases. This change was already in release 4.0.5 but was not documented in the release notes.
- The `eol/1` stream property is now available also when not explicitly specified when opening a file with `open/[3,4]`.
- Decreased overhead for reclaiming dead dynamic clauses.
- Decreased garbage collection overhead in some cases.

12.10.4 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- Exceptions in interpreted code will not get accurate source info in the source linked debugger.
- Saved-states and '`.po`' files are not portable across architectures that have the same word size, which they should be.
- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will accept some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

12.11 Changes Introduced in Version 4.0.8

12.11.1 New Features

`GROWTHFACTOR`
: A new environment variable that controls the rate at which the Prolog stacks grow when they are expanded. See Section "`sicstus` — SICStus Prolog Development System" in *the SICStus Prolog Manual*.

12.11.2 Bugs Fixed

- Compiler: shallow backtracking bug.
- Virtual machine bugs (accesses to uninitialized, freed or dead data, spurious memory corruption, recovery from memory resource error).

12.11.3 Other Changes

- Stack memory is maintained separately from other memory, which can sharply reduce memory fragmentation.
- Decreased garbage collection overhead in some cases.

12.11.4 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- Exceptions in interpreted code will not get accurate source info in the source linked debugger.
- Saved-states and '`.po`' files are not portable across architectures that have the same word size, which they should be.
- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will accept some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

12.12 Changes Introduced in Version 4.1.0

12.12.1 New Features

- SPIDER, the SICStus Prolog IDE, is a new Eclipse-based development environment for SICStus with many powerful features. See Section "SICStus Prolog IDE" in *the SICStus Prolog Manual* for more information.
- Do-loops, a new control structure and built-in predicate `do/2` for performing simple iterations. See Section "Do-Loops" in *the SICStus Prolog Manual*.
- Conditional compilation, a pre-processing step that selectively discards parts of a file at compile time. This is especially useful for writing code that needs to be compatible with more than one Prolog implementation. See Section "Conditional Compilation" in *the SICStus Prolog Manual* for more information.
- New Prolog flags:

  `dialect`
  `version_data`
  : make it easier to determine which version of SICStus is running. Especially useful with conditional compilation.

  `platform_data`
  `os_data`
  : make it easier to determine on which operating system SICStus is running.

  `min_tagged_integer`
  `max_tagged_integer`
  : The range of small integers.

  `argv`
  : Not a new flag but is no longer read-only. Setting it can be useful, e.g. in test cases.

- Added stream property `interactive` for interactive streams, like the standard input and output streams when invoking SICStus on a terminal or with the '`-i`' command line option.

- Meta-predicate declarations now allow an integer instead of : (colon) in order to help analysis tools follow code references. All documentation and libraries have been updated to reflect this. (This was always allowed in SICStus as a substitute for : (colon) but was never documented).
- Compound terms denoting references to dynamic clauses are recognized by the new built-in predicate `db_reference/1`.
- The previously reserved argument to `SP_initialize()` can now be used to pass initialization options.
- Saved-states and '`.po`' files are portable across architectures that have the same word size. Pre-4.1 '`.sav`' and '`.po`' files are not compatible with this and future releases.
- Several new statistics keywords are available. Also, `statistics/0` now resets the "time spent since the latest call" counters.
- `library(odbc)` is a new ODBC library for interfacing with databases. ODBC (Open Database Connectivity) is a standard API for using a DBMS (DataBase Management System). By using ODBC you can access data from a multitude of DBMSs without having to know the details of each DBMS.
- It is now possible to pass environment variables to the sub-process using the new `environment/1` option to `process:process_create/3`.
- `random:setrand/1` can now be passed an arbitrary integer for initializing the state of the random number generators. This is easier than constructing a valid random state like those returned by `getrand/1`.
- `library(clpfd)`:
  - `geost/[2,3,4]` is a new powerful constraint that constrains the location in space of non-overlapping multi-dimensional objects.
  - `table/[2,3]` is more scalable and has several new options for controlling its DAG construction.
  - `automaton/3` is a shorthand for the most common use of `automaton/8`, and `automaton/9` extends `automaton/8` with options.
  - Unary minus (`-`) is allowed in arithmetic expressions.
  - Several new demo examples.
- `library(zinc)`: upgraded to FlatZinc version 1.0.
- `library(system)`: a new predicate, `environ/3`, for reading system properties, environment variables or a merged view of both. See below for the new concept "System Properties" that has replaced most uses of environment variables.
- `library(sockets)`: The predicates that create socket streams now take options `encoding/1` and `eol/1` with the same meaning as for `open/4`.

12.12.2 Bugs Fixed

- Compiler: pathological case bug.
- `SP_raise_exception()` and `SP_fail()` would sometimes not be handled correctly when foreign code called Prolog recursively. Now, exceptions are preserved in callbacks from foreign functions.

- `format/[2,3]` et al. used to treat all non-reserved exceptions as consistency errors. Now non-error exceptions, i.e. not `error/2`, are passed on to the caller.

  Reserved exceptions from the goal invoked for the '`~@`' spec are now passed on to the caller. This ensures that `timeout:time_out/3` and other interrupts will be able to terminate such a goal.
- Source-linked debugging could sometimes indicate the wrong line of code.
- `see/1` and `tell/1` would not accept stream objects.
- Multiple issues when changing one of the standard streams (`user_input`, `user_output`, and `user_error`) with `set_prolog_flag/2`. These issues affected `stream_property/2`, `current_stream/3` and could lead to access to freed memory during `close/[1,2]`.
- `stream_property/2` no longer returns an `eol/1` property for binary streams.
- SICStus no longer sets any environment variables. Setting environment variables has undefined behavior in multi-threaded processes, especially on UNIX-like operating systems. Symptoms included segmentation fault in `getenv()` if several SICStus runtimes were initialized at the same time in different threads of the same process. See below for the new concept "System Properties", which has replaced most uses of environment variables.
- `SP_event()` handlers are no longer allowed to run when the SICStus runtime calls Prolog code in contexts where their result, e.g. failure or exception, cannot be propagated to the caller. This could potentially lead to timeouts and other asynchronous events being ignored.
- If `open/4` fails to open a file for writing, it will now generate a `permission_error/3` error, as prescribied by the ISO Prolog standard. It used to raise a system error.
- The `open/4` option `if_exists(generate_unique_name)` would sometimes access and use freed memory when generating a new file name.
- `SP_get_list_n_codes()` would report more bytes written than what was actually written.
- Goals run as part of `initialization/1` now have access to the load context (`prolog_load_context/2`), similarly to how other goals appearing in directives are treated. This also means that `absolute_file_name/2` will use the location saved in the load context as default directory.

  Please note: The change in default directory for `absolute_file_name/2`, and thus `open/[3,4]` et al., is an incompatible change that may affect some programs. Old code that depends on the current directive may need to be updated to explicitly call `file_systems:current_directory/1`.

  If the old code looked like:

  ```
  %% Pre 4.1.0 version
  :- initialization read_some_file('myfile').
  ```

  it can be rewritten as follows (which will also work in older versions of SICStus Prolog):

```
%% >= 4.1.0 version
:- use_module(library(file_systems), [current_directory/1]).
:- initialization  current_directory(CWD),
                   absolute_file_name('myfile', Absfile,
                                      [relative_to(CWD)]),
                   read_some_file(Absfile).
```

- `prolog_load_context(stream,S)` will now only succeed when compiling or consulting the code. It used to return a closed stream instead of failing.
- `library(clpfd)`:
  - `nvalue/2` would miss solutions.
  - `element/3` did not maintain arc-consistency in its first argument.
  - Undefined behavior when combining CLPFD with frozen goals, now made consistent.
  - Strength reduction problem for some propositional constraints.
  - Missing meta-predicate declaration for `fd_global/[3,4]`.
  - Some data was not protected from garbage collection.
  - Output of `copy_term/3` was sometimes incomplete or not correct.
- Exported, non-existing predicates: `file_systems:file_must_exist/[1,2]`, `lists3:nextto/3`, `lists3:nth/4`.
- The directory listing predicates in `library(file_systems)`, e.g. `directory_member_of_directory/2`, no longer fail if they encounter a broken symbolic link.
- `library(process)`: Process creation would leak small amounts of memory.
- `library(random)`: `maybe/0` would always fail the first time.
- A typo prevented `library(detcheck)` from working.
- The `spdet` tool now tries the extension '`.pro`' in addition to '`.pl`'. Other minor improvements.
- `library(xref)`: slighly more precise.
- `sockets:socket_client_open/3` would give system error with `SPIO_E_HOST_NOT_FOUND` when connecting to localhost on some platforms.
- `system:environ/2` would leak memory if called with a variable as first argument.
- The Emacs mode did not work in recent Emacsen.

12.12.3 Other Changes

- The atom length restriction has been lifted.
- The Emacs command `run-prolog` now prompts for a (Lisp) list of extra command line arguments, when invoked with a prefix argument, i.e. as *`C-U M-x run-prolog`*.
- While loading clauses from a PO file, if clauses for an existing multifile predicate are encountered, but in a precompiled format different from the existing clauses, the existing clauses remain untouched, the multifile clauses from the PO file are simply ignored, the load continues, and a permission error is raised at the end. Previously, the existing clauses would silently be replaced by the loaded ones. This feature is mainly relevant for hook predicates such as `user:term_expansion/6`.

- "System Properties" has been introduced as an abstraction to replace the direct use of environment variables. See Section "System Properties and Environment Variables" in *the SICStus Prolog Manual* for more information.

  The change is largely backwards compatible with the following notable exceptions:

  - The environment variables `SP_APP_DIR`, `SP_RT_DIR`, etc. are no longer set in the environment. This means that their value can no longer be obtained, e.g. from C code, by using `getenv()` or similar functions. Instead, `SP_getenv()` can be used for a similar effect.
  - For the same reason, sub-processes created with `process:process_create/[2,3]` will no longer see `SP_APP_DIR` et al. in their inherited environment. Instead it is now possible to explicitly pass environment variables to the sub-process using the new `environment/1` option to `process:process_create/3`.
- New automatically set system properties, `SP_APP_PATH`, the path to the executable, `SP_RT_DIR`, the path to the SICStus runtime, and `SP_STARTUP_DIR` the initial working directory. See Section "System Properties and Environment Variables" in *the SICStus Prolog Manual* for more information.
- The initial working directory can be set with the system property `SP_STARTUP_DIR`, independently from the process's working directory. By setting the system property `SP_ALLOW_CHDIR` to '`no`', SICStus can be told to never change the process's working directory. These features are especially useful when embedding SICStus.
- The buffer argument to `spio_t_simple_device_write` is now a `void const*` instead of a plain `void *`. This affects code that use `SP_create_stream()` to create user-defined streams.
- `SP_get_list_n_bytes()` and `SP_get_list_n_codes()` now use stricter input validation.
- SICStus will no longer flush open streams on exit. This change is to prevent SICStus from hanging on exit due to some blocking I/O operation. All streams should be explicitly closed (`close/[1,2]`) or flushed (`flush_output/1`) if their contents is precious.
- `clpfd:case/4`: the `leaves/2` option has been dropped, and the variable order must be the same on every path.
- `library(system)`: `library(system)` no longer depends on any foreign code so the `system` foreign resource is gone.
- `library(jasper)`: The SICStus (Java) working directory is now passed to Java (SICStus) when Java (SICStus) is started from SICStus (Java). Also, SICStus will not change the process's working directory when started from Java.
- The Berkeley DB library, `library(bdb)`, is now built using Berkeley DB 4.8.24.

12.12.4 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will accept some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

12.13 Changes Introduced in Version 4.1.1

Version 4.1.1 is a bugfix release only, no new features have been added. See Section 12.12 [4.1.0 Changes], page 47 for changes introduced in SICStus Prolog 4.1.

12.13.1 Bugs Fixed

- A compiler bug affecting disjunctions, introduced in released 4.1.0, fixed.

12.13.2 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will accept some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

12.14 Changes Introduced in Version 4.1.2

Version 4.1.2 is a bugfix release only, no new features have been added. See Section 12.12 [4.1.0 Changes], page 47 for changes introduced in SICStus Prolog 4.1.

12.14.1 Bugs Fixed

- Term comparison now runs in constant C stack space.
- Virtual machine bugs.
- PO file compatibility bug.
- Memory management bugs.
- Memory and stack corruption on UNIX-like systems with large file number limit and many open files (systems with `ulimit -n` larger than `FD_SETSIZE`, i.e. larger than 1024).
- *pred_spec_tree* parse error.
- Module expansion of args shadowed by do iterators.
- Command line arguments with certain non-ASCII characters would prevent SICStus from initializing.

- `ceiling/1`, `floor/1`, `round/1`, `truncate/1` now accept integers in addition to floats.
- `statistics(garbage_collection,_)` did not report the correct byte count.
- PrologBeans: one hook predicate was incorrectly declared dynamic.
- `library(bdb)`: determinacy bugs.
- CLPFD:
  - `table/[2,3]`: bugs with `inf/sup`, `copy_term/2`.
  - `geost/[2,3,4]`: polymorphism with rules; `volume/1` option.
- `library(linda)`: The Linda server is now more robust against misbehaving clients. Especially on Windows the server would get a connection reset error if the client crashed.
- `library(sockets)`: `socket_select/7` would sometimes return with nothing selected even though it was called with infinite timeout (UNIX-like platforms only).
- Prolog Beans clients (both Java and .NET) would sometimes get an array index out of bounds error when more than ten concurrent sessions were active.

12.14.2 Other Changes

- Added a warning to the PrologBeans.NET ASPX example that it is not secure.

12.14.3 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will accept some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

12.15 Changes Introduced in Version 4.1.3

See Section 12.12 [4.1.0 Changes], page 47 for changes introduced in SICStus Prolog 4.1.

12.15.1 New Features

- `library(plunit)` provides a Prolog unit-test framework.
- CLPFD:
  - `automaton/9` takes several new options that allow capturing properties of the input string, such as the number of occurrences of given patterns, into domain variables.
  - A much more general syntax is allowed for clauses of the form *`Head +: Body`*, which define dedicated propagators as indexicals. In particular, propositional combinations of arithmetic constraints are allowed.

12.15.2 Bugs Fixed

- Critical virtual machine bugs.
- Improvements to interrupt handling (`SP_event()`, `SP_signal()` and related functionality).
- When running under the SPIDER IDE, I/O operations could fail with `SPIO_E_INTERRUPTED`.
- When running under the SPIDER IDE, `restore/1` would disrupt the debugging session.
- Linking with the SICStus runtime on Linux no longer marks the stack as executable. This would prevent SICStus from starting on some versions of SE Linux.
- Work around OS bugs on Mac OS X 10.5, Mac OS X 10.6 and Linux that would sometimes cause SICStus to hang when closing streams.
- Source_info bug for huge interpreted clauses.
- Profiling: `profile_reset/1` was broken; problems with multifile.
- `library(timeout)`: `time_out/3` would sometimes interrupt I/O or miss a timeout under Windows.
- `library(sockets)`: Some operations would raise an exception when an interrupt occurred, e.g. at `^C`.
- CLPQ/CLPR: constants π and e were inaccurate.
- CLPFD:
  - `relation/3, table/[2,3]`: bugs with empty and infinite sets.
  - Missing propagation in the context of unification.
- SPRM 11909 Prologbeans: passing deeply nested terms to and from Prolog could lead to stack overflow in Prologbeans client code (Java, .NET).

12.15.3 Other Changes

- The following I/O predicates are now several times faster: `put_code/[1,2]`, `put_byte/[1,2]`, `get_code/[1,2]`, `peek_code/[1,2]`, `get_byte/[1,2]`, `peek_byte/[1,2]`, and `read_line/[1,2]`.
- JASPER: When creating a `se.sics.sicstus.SICStus` instance any (Java) system property named `se.sics.sicstus.property.`*NAME* will be passed to the created SICStus instance as the (Prolog) system property *NAME*.
- CLPFD: `scalar_product/[4,5]` is less prone to integer overflows, and faster in the most common cases.
- Linux: The installer script would sometimes fail to configure support for Java.

12.15.4 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will accept some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

12.16 Changes Introduced in Version 4.2.0

12.16.1 New Features

- If the Prolog flag `source_info` is on at compile time, more information from the source code is kept around at runtime, with improved functionality including:
  - The debugger can show variable bindings for the current clause and its ancestors.
  - The Emacs interface offers a variable bindings window that is updated on every debugger port interaction.
  - `listing/[0,1]` displays the clauses with their source code variable names.
  - PO files and saved-states retain information about variable names and line numbers.
  - The SICStus Prolog IDE (SPIDER) also takes advantage of the improved debugging information, e.g. the variable bindings view in SPIDER now has information for more variables than in earlier releases.
  - Generic runtime systems are now available on all platforms, not only on Windows. See Section "Generic Runtime Systems" in *the SICStus Prolog Manual*, for details.
- Execution profiling has been reengineered:
  - The execution profile can be printed in a format similar to `gprof(1)`.
  - There is no longer any need to specially instrument code for profiling.
  - Execution profiling is available for compiled as well as interpreted code.
  - Execution profiling is either globally *on* or globally *off*, reflected by the value of the new `profiling` Prolog flag.
  - The new scheme keeps track of the number of calls per caller-callee pair.
  - The new scheme detects calls that succeed nondeterminately.
  - No clause level data is maintained, all data is per predicate.
  - The `profiledcode` value of the `compiling` Prolog flag has been dropped.
  - PO files do not store any profiling data.
  - The built-in predicates `profile_data/4` and `profile_reset/1` have been replaced by `profile_reset/0`, `profile_data/1`, `print_profile/[0,1]`, `coverage_data/1`, and `print_coverage/[0,1]`.
  - SPIDER can present the profile information.
- Improved performance on Linux Intel, 32 and 64 bits.
- Coverage analysis is now available, for compiled as well as interpreted code:
  - It uses the same infrastructure as execution profiling.

  - Code coverage can be reported textually in a hierarchical format, or alternatively by highlighting the relevant lines of code in the relevant Emacs buffers and in SPIDER.
- It is now possible to debug runtime systems, e.g. when SICStus is embedded in some other applications, such as Java. It is also possible to attach to a runtime system from SPIDER. See Section "Debugging Runtime Systems" in *the SICStus Prolog Manual*.
- The `meta_predicate/1` predicate property will retrieve the specifications used in the original meta-declaration, which can be integers or the atoms `:`, `+`, `-`, or `?`. Previously, only the atoms `:` or `?` would be retrieved. **Please note**: This is an incompatible change. Code that inspects this predicate property may need to be updated.
- A new stream property, `id`, has been added. This property provides a unique identity that is never re-used, even after the stream has been closed. See Section "`stream_property/2`" in *the SICStus Prolog Manual*.
- CLPFD:
  - The `case/[3,4]` constraint has been extended to take linear inequalities into account in addition to the DAG.
  - The new constraint `smt/1` provides a front-end to the extended `case/[3,4]` constraint.
  - Reified constraints can be used as terms in arithmetic expressions.
- `library(zinc)`: upgraded to FlatZinc version 1.2.
- Changes to `library(odbc)`:
  - New predicates:

    `odbc_current_table/[2,3]`
    : Enumerate tables and their attributes.

    `odbc_table_column/[3,4]`
    : Enumerate table columns and their attributes.
  - `odbc:odbc_query_close/1` can now close both result sets and statement handles.
  - The format has changed for some odbc exceptions. Now all odbc-related exceptions have the same basic structure.
- When SICStus is started from Emacs, using *`M-x run-prolog`*, or from the launcher script in the '`Applications`' folder, the system property `SP_ULIMIT_DATA_SEGMENT_SIZE` is set to `unlimited`. This ensures that overly restrictive default limits on process memory usage do not affect SICStus. This is primarily an issue on Mac OS X. Previously, this setting was only applied when SICStus was invoked from the SICStus Prolog IDE (SPIDER).

12.16.2 Bugs Fixed

- Interrupt latency problem fixed.
- Fixed exception handling bug introduced in 4.1.3.
- Misencoded and null strings from C are handled gracefully.
- `write_term(X,[max_depth(D)])` did not always respect the depth limit.
- `seeing/1` and `telling/1` now return `user` for the current input resp. output stream, which was always intended.

- If the standard input stream encounters an invalid character, it will be silently replaced with the Unicode replacement character `0xFFFD`. This was already the case for the standard output streams.

  The same behavior, for the standard input and output streams, is now also in effect when running SICStus in the SPIDER IDE.
- When `open/[3,4]` cannot expand a system property, e.g. `open('$FOO/bar.txt', read, S)` when the system property `FOO` is undefined or empty, an exception is raised. Previosly, `open/[3,4]` silently failed in this case.
- The `spld` tool would ignore the '`--namebase`' argument for some generated files.
- CLPFD:
  - Dangling pointer hazard fixed for domain variables with frozen goals.
  - Missing propagation problems fixed.
  - `sorting/3` could fall into infinite loop.
  - Indexical compilation problems fixed.
- CLPQ/CLPR: called the undefined predicate `'C'/3`.
- `library(terms)`: bugs in `sub_term/2`, `term_variables/2`, `subsumeschk/2` and friends.
-

  Improvements in how `library('linda/server')` and `library('linda/client')` handle server shutdown.

  The server will stop listening for new connections as soon as it receives a shutdown request from a client. Among other things this makes the socket port available for re-use on the same machine.

  A call to `linda_client:shutdown_server/0`, in the client, will not return until the server has acknowledged the command. This removes a race condition when the client attempts to re-connect to the server.
- A number of problems in `library(odbc)` has been fixed. There are also some new features and other changes; see above.

12.16.3 Other Changes

- Windows 2000 is no longer supported, for it is no longer supported by Microsoft.
- The limitations on "temporary" and "permanent" variables for compiled clauses have been dropped. There is no size limit on compiled clauses.
- PO files and saved states are now much smaller than in earlier releases.
- The new features required changes to the PO file format. PO files and saved states created by previous versions are not compatible with this version, and vice versa.
- The `spld` tool now defaults to '`--moveable`' on Linux, Mac OS X and Solaris. This can be turned off with the new option '`--no-moveable`'.

12.16.4 Known Issues

The following are known issues with this release of SICStus. See Section 12.3 [Limitations in the Current Release], page 36 for more information about missing or incomplete features in this release.

- SICStus Prolog does not verify that Prolog text is in Unicode NFC format.

  When reading terms, SICStus Prolog currently does not verify that the input text contains valid Unicode 5.0 characters in Normal Form C. See Section "Syntax of Tokens as Character Strings" in *the SICStus Prolog Manual*.

  This is not a problem as long as the input is in the proper format but it will accept some input that may be rejected or interpreted differently in a future version of SICStus Prolog.

13 Generic Limitations

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

The number of atoms created may not exceed 1048575 (33554431) on 32-bit (64-bit) architectures.

Saved-states and '`.po`' files are not portable between 32-bit and 64-bit architectures.

Indexing on large integers or floats is coarse, i.e. there is essentially no indexing between different large integers or floats. This can have a **huge** negative impact on performance, e.g. when using hash codes or some such to represent a hash-table as clauses. The hash predicates in `library(terms)` avoid this, by default, but it has been known to cause hard to track down performance problems when the number is created by some other means.

14 Contact Information

Current support status for the various platforms as well as a web interface for reporting bugs can be found at the SICStus Prolog 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.

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

```
subscribe sicstus-users
```