Quintus Prolog Release Notes

by the Intelligent Systems Laboratory

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

Release 3.5
December 2003

Swedish Institute of Computer Science
qpsales@sics.se http://www.sics.se/quintus/

This manual corresponds to:

Quintus Prolog Release 3.5

SICS PO Box 1263
SE-164 29 Kista, Sweden
+46 8 633 1500
http://www.sics.se/quintus/

Authorization Codes:
qpadmin@sics.se

Problem Reports and Product Suggestions:
qpsupport@sics.se

Additional Information:
qpsales@sics.se

Mailing List Subscription:
majordomo@sics.se
with '`subscribe quintus-users`' in the message body

Table of Contents

1 Installation Guide

This chapter describes the installation process, mostly from a UNIX perspective. For information specific to the Windows installer, see Section 1.7 [Installing under Windows], page 9.

1.1 Overview

There are three basic steps to installing Quintus Prolog:

1. Downloading the software. Approximately 30 Mb of disk space is required per platform.
2. Running the installation script to install the Quintus software and perform post installation tasks such as building executables.
3. Running the license manager to set up the license file.

1.1.1 Components

Quintus Prolog is organized into a number of components to enable you to easily choose which pieces of the release you wish to install. The installation script lists the available components that can be installed. Under Windows, all available components are installed.

The following components are included in Quintus Prolog Release 3.5 on all platforms:

`pro` The core component that contains the development system, runtime generator and associated tools, plus the Quintus library, tutorial and demo files and the main help system files.

`edit` Support files for the interface between GNU Emacs and Quintus Prolog.

The following components are included on UNIX platforms:

`proxl` A low level interface to the X Window system, built directly on top of the Xlib library.

`proxt` An interface to the Motif toolkit that provides access to Motif functionality for developing graphical user interfaces in Prolog.

`qui` The Quintus User Interface, providing a Motif based development environment that includes a source linked debugger for Prolog.

`sys` This component updates certain operating system files, for example to enable `make(1)` and `file(1)` to understand QOF files.

The `pro` component must be installed in order to run Quintus Prolog. All other components are optional, although we suggest that they all be installed.

1.2 Preparing For Installation

The *QuintusDir* referred to throughout this Installation Guide is the directory in which you will be instaling your Quintus software.

You can choose any directory as *QuintusDir*, so long as there is sufficient space. Under UNIX, a popular location is '`/opt/quintus`'. Under Windows, a popular location is '`C:\Program Files\Quintus Prolog 3.5`'.

If possible, we recommend that you do all installation tasks for Quintus products while logged in as `root` on the file server where the release will be installed. This avoids many potential problems with directory ownership and protection.

However, Quintus Prolog has been organized so that only the optional `sys` component must be installed by `root`; all other components can be installed by any user.

Your *QuintusDir* must be mounted (or mountable) by all machines on which Quintus Prolog is to be used. At the beginning of the installation process you will be asked to confirm your choice of *QuintusDir*.

Although we don't recommend it, if you have an earlier version of Quintus Prolog Release 3 installed, you can use the same directory for installing release 3.5, provided that enough space is available. You do not have to use that directory and can use another one if you prefer. You may wish to delete the earlier version prior to unloading Quintus Prolog Release 3.5 in order to free disk space for your new release. Deleting an earlier version of Quintus Prolog Release 3 does not invalidate saved states or other QOF files built with that release. Older QOF files will work with Release 3.5. However, saved states will no longer be executable from the UNIX shell if you do this. In order to make a saved state executable from the UNIX shell, it can be loaded into the 3.5 system and then saved out again using the `save_program/1` predicate to create a new saved state dependent on the new release.

Warning: Do not rename an old installation directory to make the name available for the new release directory (unless you don't mind the old release ceasing to work). If you rename a Quintus release directory you have to redo the installation process since the directory name is inserted into a number of the executable files.

1.3 Running the Installation Script

The next step is to invoke the '`install3.5`' script. The script will verify that you are running on an appropriate platform and print some information:

```
% ./install3.5

        Quintus Prolog Release 3.5 Installation

   Installer            : <your login name>
   Hostname             : <your hostname>
   Machine type         : <your machine type>
   Operating System     : <your OS release version>
   Date of installation : <date you installed Quintus>
```

The installation script then asks for the path of the *QuintusDir*. This must be the same as the directory where the install script is located and is determined automatically by the install script. One reason to modify it would be if the default contained automount directories such as '`/tmp_mnt`'.

```
Enter name of Quintus home directory [<default directory>]:
```

The script will then print the path chosen for *QuintusDir* and the installation log file.

```
   Quintus directory    : QuintusDir
   Installation log file: QuintusDir/<log file>
```

Next, the script lists the available components to install:

```
   Select components that you wish to install.  You may type 'all'
   to select all components or else type the individual components
   on one line (e.g. pro edit qui).  Components marked with '*'
   appear to be already installed.

   The following components are available:

         edit  - Quintus Prolog Emacs support files  (0.3Mb)
         pro   - Quintus Prolog for Sun Sparc Platform  (22.5Mb)
         proxl - Quintus Prolog low level interface to X windows (2.6Mb)
         proxt - Quintus Prolog interface to Motif toolkit  (1.6Mb)
         qui   - Quintus Prolog Motif-based User Interface  (3.4Mb)
         sys   - Quintus Prolog System Files  (0.0Mb)

Select [all]: pro qui edit
```

Some components depend on others—for example, most components depend on the `pro` component. The script will inform you if you select a component that is dependent on another component that is not selected or installed.

The `sys` component will only appear if you are running as `root`. It will appear to be installed already (i.e. marked with '`*`') if the `sys` component has been installed from a previous release of Quintus Prolog. We recommend that you install this again as this release may contain updated versions of certain files.

The script will then print the list of components it is about to install and ask if you wish to continue. If you type *n* to this then the installation script will abort. Otherwise it will proceed to the next stage.

```
The following components will be installed: pro qui edit

Do you wish to continue? (y/n) y
```

The postinstall scripts are then executed for each component that has one. With the exception of the `sys` package, these are not interactive and will just print informational messages about what they are doing. For example:

```
% Executing pro install script
% Setting paths in executables
% Building prolog
% Executing qui install script
% Building qui
% Executing proxt install script
% Building proxt shared objects
```

The `sys` postinstall script will request some information about adding links to Prolog executables from a '`bin`' directory.

1.4 Running the License Manager

Having installed Quintus Prolog, the license file must then be set up with a license code in order to run `prolog`, `qpc`, etc. This is achieved with the license manager program `qplm`, which resides in the Quintus runtime directory (for example '*QuintusDir*`/bin3.5/linux`').

The first step is to initialize the license file with *`qplm -i site name`*. The site name given to `qplm` here should be exactly the same as supplied with the code from SICS since it is used to generate the code. For example:

```
% qplm -i "Hallatrow Designers Inc."
```

The next step is to add the products and codes. This is done with the command *`qplm -a product users code`*. The *product* argument is a string that comprises the product name (`prolog` in this case) the platform and the version number. E.g., the Linux release thus requires a code for `prolog/linux/3.5`.

The *users* argument specifies the number of regular users of Quintus Prolog at this site and *code* is a 16 character string that is supplied by SICS. To add a code for a 5 user license, issue the command:

```
% qplm -a prolog/archdir/3.5 5 thiscodewontwork
```

SICS also provides temporary licenses that expire on a given date. The expiration date can be added with the command *`qplm -a product users expire code`*, where *expire* is an

expiration date with the format YYYY-MM-DD. The following example adds a 2 user license that expires on 17 May 2004:

```
% qplm -a prolog/archdir/3.5 2 2004-05-17 thisalsowontwork
```

You can print the information in the license file, including site name and products licensed with the *qplm -p* command.

You can delete regular users from the license file if they no longer use the product with the command *qplm -d product username*.

1.4.1 The License Directory

The licensing system maintains two files in the directory '*QuintusDir*/license3.5':

'license.qof'
: contains the license code(s)

'users.qof'
: maintains a list of users of Quintus Prolog

The 'users.qof' file is written to each time an "occasional" user invokes prolog or qpc, where an "occasional" user is one who has not used the product more than five times.

If *QuintusDir* resides on a file system that is mounted read-only on any machines where Quintus Prolog is run, then the license directory should be replaced with a symbolic link to a directory on a filesystem that is writable before running qplm. This is illustrated by the following commands, where *QuintusDir* is '/opt/quintus' and the license directory is moved to '/var/quintus/license3.5':

```
% rmdir /opt/quintus/license3.5
% mkdir /var/quintus/license3.5
% ln -s /var/quintus/license3.5 /opt/quintus/license3.5
```

Please note: The 'users.qof' file is intended as an *aid* for you to remain in compliance with your product license. Circumventing it, for example by deleting regular users or making it unwritable, does not bring you in compliance with your license.

1.5 The Sys Component

The sys component performs installation operations that affect global system files and therefore can only be done while logged in as root. These operations are optional, but useful. Specifically, the sys component attempts to do the following (for some platforms only a subset of these operations is performed):

- install '`quintus.h`' in '`/usr/include/quintus/quintus.h`'
- add lines to the default rules file that enables the `make(1)` command to recognize how to make QOF files
- add lines to the '`/etc/magic`' file to recognize QOF files and saved states.
- (optionally) create links to the executables in the Quintus runtime directory from the '`/usr/local/bin`' directory.
- (optionally) create links to the man pages in the Quintus directory from the '`/usr/local/man`' directory.

If, in your computing environment, operating system files reside on local disks then you may wish to install this component on each machine where Quintus Prolog is going to be used. Consult with your system administrator if you have any doubts regarding these procedures.

1.5.1 The '`quintus.h`' Include File

The '`quintus.h`' include file contains definitions and declarations that are useful when using the Quintus Prolog Foreign Interface, embedding Prolog in another language, managing interrupts, etc.

The `sys` component postinstall script attempts to install '`quintus/quintus.h`' (creating that directory if necessary) in the directory given by the environment variable `QPINCDIR`, or '`/usr/include`' if not set. That is, '`/usr/include/quintus/quintus.h`' is created by default.

Once this is done, users writing C code to be used with Quintus Prolog can simply include the file in their C source using:

```
#include <quintus/quintus.h>
```

If you do not install the `sys` component on your system then users who need to #include '`<quintus/quintus.h>`' can use the C compiler option:

`-I`*QuintusDir*`/generic/qplib3.5/include` (UNIX), or
`-I`*QuintusDir*`\include` (Windows)

1.5.2 Updating Default Rules File

The UNIX utility `make(1)` may be configured to ensure that QOF files are up to date by modifying the default make rules to recognize certain kinds of files generated and used by Quintus Prolog and various Quintus utilities.

The `sys` component postinstall script attempts to modify this file, adding the following rules at the end:

```
# Quintus Prolog Compiler (qpc) section
.SUFFIXES: .qof .pl
QPC=qpc
QPCFLAGS=
.pl:
        $(QPC) $(QPCFLAGS) -o $@ $<
.pl.qof:
        $(QPC) $(QPCFLAGS) -cN $<
```

Once in place, `make(1)` will be able to automatically update QOF files. `QPCFLAGS` can be set to pass `qpc` any flags needed when building your application.

If you do not install the `sys` component on your system then the same effect can be produced by including the above rules in any makefiles involving `qpc` and Prolog code.

1.5.3 Updating '/etc/magic'

The UNIX command `file(1)` determines the type of a file by examining its contents. By modifying the file '`/etc/magic`', which records characteristics of files that `file(1)` recognizes, `file(1)` can be used to identify QOF files and QOF files that are saved states.

The `sys` component postinstall script attempts to modify this file, adding the following lines to the end:

```
0       string      QOF             Quintus Prolog Object File,
>10     short       x               Ver %d
>8      short       x               Rev %d
>12     long        0x04030201      (byte-swapped)
256     string      QOF             Quintus Prolog Saved-state,
>266    short       x               Ver %d
>264    short       x               Rev %d
>268    long        0x04030201      (byte-swapped)
```

If '`/etc/magic`' has been updated, and the file '`camerton.qof`' is a QOF file that has been built using Quintus Prolog Release 3.5, you should see:

```
% file camerton.qof
camerton.qof: Quintus Prolog Object File; Ver 1 Rev 71
```

If you do not install the `sys` component on your system then the same effect can be achieved by creating your own copy of the system's '`/etc/magic`' file and adding the above lines to it.

Your version of '`/etc/magic`' can then be used with the '`-m`' option to `file(1)` to recognize QOF files. If you have made your own copy of '`/etc/magic`' in '`~/mymagic`' and modified that copy as suggested, you can type:

```
% file -m ~/mymagic camerton.qof
camerton.qof: Quintus Prolog Object File; Ver 1 Rev 71
```

1.5.4 Links for Executable Files and Man Pages

The `sys` postinstall script will ask whether you wish to install links to Quintus executables in the directory given by the environment variable `QPBINDIR`, or '`/usr/local/bin`' if this is not set. If you answer no to this question then no links are installed.

Quintus Prolog users will need to add the Quintus runtime directory to their `PATH` in order to find the executables `prolog`, `qpc`, etc. For example:

```
% PATH=$PATH:QuintusDir/bin3.5/archdir; export PATH
```

The `sys` postinstall script will then ask whether you wish to install links to Quintus man pages in the directory given by the environment variable `QPMANDIR`, or '`/usr/local/man`' if not set. If you answer no to this question then no links are installed.

Quintus Prolog users may need to add the the Quintus man directory to their `MANPATH` in order to find the man pages for `prolog`, `qpc`, etc. For example:

```
% MANPATH=$MANPATH:QuintusDir/generic/q3.5/man; export MANPATH
```

1.6 The GNU Emacs Interface

The `edit` component provides support files for an an interface between Quintus Prolog and GNU Emacs and XEmacs. For information on obtaining these editors, see http://www.gnu.org and http://www.xemacs.org. Under Windows, we recommend XEmacs since its Windows installer is much easier to use.

In addition, to use the GNU Emacs interface, you must set an environment variable which tells Prolog where GNU Emacs is located:

```
% setenv QUINTUS_EDITOR_PATH /opt/gnu/bin/emacs (csh or tcsh)
% export QUINTUS_EDITOR_PATH=/opt/gnu/bin/emacs (sh, bash or ksh)
> SET QUINTUS_EDITOR_PATH=c:\Program Files\XEmacs\XEmacs-21.4.13\i586-
pc-win32\xemacs.exe (Windows)
```

Now to invoke `prolog` with GNU Emacs:

```
% prolog +
```

To exit, type *C-x C-c*.

1.7 Installing under Windows

Under Windows, you install Quintus Prolog by running '`InstallQuintus3.5.exe`'. It will prompt for an installation location (by default '`C:\Program Files\Quintus Prolog 3.5`') and finally give you the option to enter the license information. If you need to re-enter the license information at a later time you can run '`qplm.exe`', as described in Section 1.4 [Running the License Manager], page 4.

There is currently no uninstaller for Windows. If, for some reason, you would like to uninstall Quintus Prolog you can simply remove the installation folder, e.g. '`C:\Program Files\Quintus Prolog 3.5`'.

When using Quintus Prolog from a command prompt you may wish to modify certain environment variables so that Quintus Prolog components are found. This can be done by running the '`qpvars.bat`' script located in the Quintus Prolog binaries folder, by default '`C:\Program Files\Quintus Prolog 3.5\bin\ix86`'. Alternatively you can look at the script to figure out what environment variables it changes and then perform the same changes using the Windows XP '`System`' control panel.

In addition you may wish to add a shortcut to the '`qpwin.exe`' to the Start menu or desktop. Note that, under Windows, source linked debugging is only available when running Quintus Prolog with Emacs.

1.8 Platform Specific Notes

Please refer to the table in section "Using Shared Object Files and Archive Files" in *the Quintus Prolog Manual* for platform specific instructions for building shared object files and archive files.

Digital Alpha

Quintus Prolog was revived for this platform in Release 3.4:

- 32 bit integers and pointers are used internally. However, the functions `QP_get_integer()` and `QP_put_integer()` take long integer arguments, and `integer` in the foreign language interface corresponds to long integers in foreign code. In the Structs package, the `long` type corresponds to long integers whereas the `integer` type corresponds to integers. Long integers are however truncated to 32 bits (sign-extended) by the Prolog system.
- In foreign code interfacing Prolog via the foreign language interface, pointers and long integers are assumed to occupy 64 bits. Hence, such code must be compiled in native mode, i.e. not using the `cc` option '`-xtaso_short`'. Furthermore, `qld` will link executables with the '`-taso`' option, so as to ensure that all pointers fit in 32 bits internally.

Silicon Graphics
: Up to and including release 3.3, Quintus Prolog ran under IRIX 6.2 or later and used the O32 ABI. Later releases run under IRIX 6.2 or later and use the N32 ABI.

Sparc Solaris
: Up to and including release 3.3, Quintus Prolog did not comply with the SPARC ABI; later releases do.

IBM RS/6000
: Quintus Prolog was revived for this platform in Release 3.4, under AIX 4.3, and complies with the AIX 4.3 32-bit ABI.

HP-UX
: Up to and including release 3.3, Quintus Prolog did not comply with the HP-PA ABI; later releases do.

Linux
: Quintus Prolog was ported to this platform in Release 3.5, under Red Hat Linux 9, i.e. it requires `glibc 2.3`.

2 Revision History

This chapter highlights the changes between this and Release 3.1 of Quintus Prolog.

2.1 News in Release 3.2

- Atom garbage collection, whereby memory for atoms no longer referenced by the Prolog execution can be reclaimed. With the introduction of this feature, the maximum length of atoms has also been increased from 1024 bytes to 65532 bytes. `library(strings)` has been modified to support the long atoms, only allocating space for long atoms as needed.
- The TCP library has been significantly enhanced to provide more efficient passing of terms between Prolog processes, more functionality for server processes and support for input callbacks, where predicates can be called either when a specified file descriptor becomes ready or at a specified time.
- A new licensing system has been introduced with this release. A license manager program is provided to setup the licenses and provide information on licenses installed.
- The cross-referencer tool has been replaced by a new version that is tuned to finding unreachable code in large programs.
- **Windows note:** *Windowed* executables can be built using the '`-W`' option to `qld`. Such executables direct the standard I/O streams to a dedicated window.
- **Windows note:** Some minor changes in the directory structure have been done, e.g. the `host_type` and `system` directory name has changed from `i386` to `ix86`.

2.2 Bugs Fixed in Release 3.2

- `avl:avl_fetch_1/5 is` missing
- `avl:ord_list_to_avl/2`: creates avl tree with incorrect functor
- `format/2`: segmentation faults with large float exponents
- `set_input(user_output)` raises exception
- float unification in QOF files fails
- `qpc` syntax error message differs from development system
- `lists:nth1/3` description incorrect
- `use_module/2` information missing from QOF files
- `library(tcp)`: connection requests on callbacks functionality
- `date:datime/7` format inconsistent with `time_stamp/3`
- `library(structs)`: use of incomplete enum type preceedes complete declaration of the enum.

2.3 Bugs Fixed in Release 3.3

- Pascal Foreign Function Interface bug for `-string`
- bug in `library(putfile)`
- memory leak in `garbage_collect_atoms/0`
- stack shifter bug
- `atom_chars/2` with 256 element list bug
- `qpc` permanent variables limit detection
- Sun C calling convention bug
- QUI interrupt dialog box has no buttons
- ProXL runtimes don't work due to module qualification

2.4 News in Release 3.4

- An Emacs-based source-linked debugger has been added, with approximately the same functionality as the QUI-based debugger.
- The GNU Emacs interface was thoroughly revised and supports GNU Emacs 19 and 20.

2.5 Bugs Fixed in Release 3.4

- `current_predicate/2` failed after loading QOF-file saved by `save_predicates/2`
- re-exported predicates could fail if the `unknown` flag was `fail`
- dynamic code generation bug for imported predicates in runtimes
- bug in finding the correct module of an already loaded QOF file
- `user:message_hook/3` not obeyed on restore
- undefined profiler predicate
- `qpc` bugs: input file suffix, `all_dynamic(true)` in embedded loads, handling '`-N`' and '`-o`' flags
- runtimes did not accept `(Module:Head :- Body)` clauses
- arithmetic bugs
- `retract/1` could leave a dangling pointer
- `ensure_loaded/1` did not work on QOF-files saved by `save_predicates/2`
- advice checking and debugging did not work when invoking a saved state
- DCG expansion of `{Goal}` bug
- C calling Prolog error handling bug
- re-exportation and meta-predicate interaction bug
- `save_modules/2` failed to save a dependency corresponding to `:- use_module(file,[])`.
- garbage collection bugs

- indexing bug upon loading QOF-files
- `QP_seek()` bug
- SIGPIPE signals cause abort instead of exit
- library bugs: `strings:span_*/*`, `strings:midstring/*`, `strings:concat_atom/*`, `strings:concat_chars/*`, `directory:file_member_of_directory/2`, `heaps:get_from_heap/4`, `date:time_stamp/*`
- `proxl:current_window/1`, `proxl:current_display/1` could crash
- `library(tcp)` bugs: floating point byte swapping, `tcp_shutdown()`, non-lingering `tcp_listen()`, file name size limitation
- A large number of documentation bugs.

  Windows notes:
  - binary gets and puts broken
  - `functor/3` bug
  - runtime statistics wrap-around bug
  - handling of asynchronous input, EOF and interrupts in windowed executables such as `qpwin.exe`

2.6 News in Release 3.5

- **For Windows:** It is now possible to build DLLs containing Prolog code to be linked dynamically into applications.
- **For Windows:** Quintus Prolog can be called from Visual Basic.
- The Objects Package is now bundled with the distribution.
- New functionality have been added to the cross-referencer (`qpxref`) and determinacy checker (`qpdet`) tools:
  - they are now fully compatible with the module system, meta-predicates, and exceptions
  - predicates can be declared (non)determinate
  - determinacy checking can be integrated with normal compilation
  - `qpdet` makes a better analysis of predicates calling nondeterminate predicates, including built-ins
  - `qpdet` can optionally find all nondeterminacy through global analysis
- The documentation has been ported to the Texinfo markup language, affecting details of the Help system and Emacs interfaces.
- Where a library module had one name with an underscore and another name without it, only the name without an underscore has been retained.
- `library(prologbeans)` is a package for issuing Prolog queries from a separate Java process. **Please note:** Feedback is solicited on this first release. Future API changes are likely.

2.7 Bugs Fixed in Release 3.5

- critical garbage collection bugs
- a critical bug in interpreted arithmetic
- prelinked QOF files were not considered up to date in binaries
- bug in `ordsets:ord_nonmember/3`
- **For Windows:** CR/LF handling bug in the DOS console