The l3sys-query script: System queries for LaTeX using Lua

The LaTeX Project*

Release 2024-04-03

Contents

1 Introduction **1**

2 The command line interface **2**
2.1 `ls [⟨args⟩]` 2
2.2 `pwd` 3

3 Spaces in arguments **3**

4 Wildcard expansion handling **4**

5 The LaTeX interface **4**

1 Introduction

TeX engines provide only very limited access to information about the system they are used on: using primitives, one can for example get the size of a single file, but not a list of files in a given location. For most documents, this is not an issue as they are self-contained. However, for cases where "dynamic" construction of parts of a document is needed based on file lists or other system-dependent data, methods to obtain this from (restricted) shell escape are desirable.

Security considerations mean that directly querying the system shell is problematic for general use. Instead, *restricted* shell escape may be used to get many details, provided a suitable tool is available to provide the information in a platform-neutral and security-conscious way. The Java program `texosquery`, written by Nicola Talbot, has been available for a number of years to provide this facility. As well as file system insight, `texosquery` also provides for example locale data and other system information. However, the requirement for Java means that the script is not automatically usable when a TeX system is installed.

The LaTeX team have therefore provided a Lua-based script, `l3sys-query`, which conforms to the security requirements of TeX Live using Lua to obtain the system information. This means that it can be used "out of the box" across platforms. The facilities provided by `l3sys-query` are more limited than `texosquery`, partly as some

*Email: latex-team@latex-project.org

information is available in modern TEX systems using primitives, and partly as the aim of `l3sys-query` is to provide information where there are defined use cases. Requests for additional data interfaces are welcome.

2 The command line interface

The command line interface to

`l3sys-query` ⟨*cmd*⟩ [⟨*option(s)*⟩] [⟨*args*⟩]

where ⟨*cmd*⟩ can be one of the following:

- `ls`
- `ls` ⟨*args*⟩
- `pwd`

The ⟨*cmd*⟩ are described below. The result of the ⟨*cmd*⟩ will be printed to the terminal in an interactive run; in normal usage, this will be piped to the calling TEX process. Results containing path separators *always* use `/`, irrespective of the platform in use.

As well as these targets, the script recognizes the options

- `--exclude` Specification for directory entries to exclude
- `--ignore-case` Ignores case when sorting directory listings
- `--pattern` (`-p`) Treat the ⟨*args*⟩ as Lua patterns rather than converting from wildcards
- `--recursive` (`-r`) Enables recursive searching during directory listings
- `--reverse` Causes sorting to go from highest to lowest rather than lowest to highest
- `--sort` Sets the method used to sort entries returned by `ls`
- `--type` Selects the type of entry returned by `ls`

The action of these options on the appropriate ⟨*cmd(s)*⟩ is detailed below.

2.1 `ls` [⟨*args*⟩]

Lists the contents of one or more directories, in a manner somewhat reminiscent of the Unix command `ls` or the Windows command `dir`. The exact nature of the output will depend on the ⟨*args*⟩, if given, along with the prevailing options. Note that the options names are inspired by ideas from the Unix commands `ls` and `find` as well as the Windows command `dir`: they therefore do not map directly to those of any one of the command line tools that they somewhat mirror.

When no ⟨*args*⟩ are given, all entries in the current directory will be listed, one per line in the output. This will include both files and subdirectories. Each entry will include a path relative to the current directory: for files *in* the current directory, this will be `./`. The order of results will be determined by the underlying operating system process: unless requested *via* an option, no sorting takes place.

As standard, the ⟨*args*⟩ are treated as a file/path name potentially including `?` and `*` as wildcards, for example `*.png` or `file?.txt`.

```
l3sys-query ls '*.png'
```

Some care is needed in preventing expansion of such wildcards by the shell or `texlua` process: these are detailed in Section 4. In this section, `'` is used to indicate a character being used to suppress expansion: this is for example normal on macOS and Linux.

Removal of entries from the listing can be achieved using the `--exclude` option, which should be given with a ⟨*xarg*⟩, for example

```
l3sys-query ls --exclude '*.bak' 'graphics/*'
```

Directory entries starting `.` are traditionally hidden on Linux and macOS systems: these "dot" entries are excluded from the output of `l3sys-query`. The entries `.` and `..` for the current and parent directory are also excluded from the results returned by `l3sys-query` as they can always be assumed.

For more complex matching, the ⟨*args*⟩ can be treated as a Lua pattern using the `--pattern` (`-p`) option; this also applies to the ⟨*xarg*⟩ argument to the `--exclude` option. For example, the equivalent to wildcard `*.png` could be obtained using

```
l3sys-query ls --pattern '^.*%.png$'
```

The results returned by `ls` can be sorted using the `--sort` option. This can be set to `none` (use the order from the file system: the default), `name` (sort by file name) or `date` (sort by date last modified). The sorting order can be reversed using `--reverse`. Sorting normally takes account of case: this can be suppressed with the `--ignore-case` option.

The listing can be filtered based on the type of entry using the `--type` option. This takes a string argument, one of `d` (directory) or `f` (file).

As standard, only the path specified as part of the ⟨*args*⟩ is queried. However, if the `--recursive` (`-r`) option is set, the query is applied within all subdirectories. Subdirectories starting with `.` (macOS and Linux hidden) are excluded from recursion.

For security reasons, only paths within the current working directory can be queried, this for example `graphics/*.png` will list all `png` files in the `graphics` subdirectory, but `../graphics/*.png` will yield no output.

2.2 `pwd`

Returns the absolute path to the current working directory from which `l3sys-query` is run. From within a TEX run, this will (usually) be the directory containing the main file, assuming a command such as

```
pdflatex main.tex
```

The `pwd` command is unaffected by any options.

3 Spaces in arguments

Since `l3sys-query` is intended primarily for use with restricted shell escape calls from TEX processes, handling of spaces is unusual. It is not possible to quote spaces in such a call, so for example whilst

```
l3sys-query ls "foo *"
```

does work from the command prompt to find all files with names starting `foo␣`, it would not work *via* restricted shell escape. To circumvent this, `l3sys-query` will collect all command line arguments after any ⟨*options*⟩, and combine these as a space-separated ⟨*args*⟩, for example allowing

```
l3sys-query ls foo '*'
```

to achieve the same result as the first example. The result is that the ⟨*args*⟩ will only every be interpreted by `l3sys-query` as a single argument. It also means that spaces cannot be used at the start or end of the argument, nor can multiple spaces appear between non-space arguments.

4 Wildcard expansion handling

The handling of wildcards needs some further comment for those using `l3sys-query` from the command line: the expl3 interface described in Section 5 handles this aspect automatically for the user.

On macOS and Linux, the shell normally expands globs, which include the wildcards `*` and `?`, before passing arguments to the appropriate command. This can be suppressed by surrounding the argument with `'` characters, hence the formulation

```
l3sys-query ls '*.png'
```

earlier.

On Windows, the shell does no expansion, and thus arguments are passed as-is to the relevant command. As such, `'` has no special meaning here. However, to allow quoting of wildcards from the shell in a platform-neutral manner, `l3sys-query` will strip exactly one set of `'` characters around each argument before further processing.

It is not possible to use `"` quotes at all in the argument passed to `l3sys-query` from TEX, as the TEX system removes all `"` in `\input` while handling space quoting.

Restricted shell escape prevents shell expansion of wildcards entirely. On non-Windows systems, it does this by ensuring that each argument is `'` quoted to ensure further expansion. Thus a TEX call such as

```
\input|"l3sys-query ls '*.png'"
```

will work if `--shell-escape` is used as the argument is passed directly to the shell, but in restricted shell escape will give an error such as:

```
I can't find file `"|l3sys-query ls '*.png'"'.
```

The LATEX interfaces described below adust the quoting used depending on the `shell-escape` status.

5 The LATEX interface

Using `l3sys-query` is not tied to access *via* expl3, but this is the preferred approach for the LATEX Team. Details of how to use `l3sys-query` as an expl3 programmer will covered in `interface3.pdf` once the macro code is finalized. A document level interface will also be provided via a l3sys-query package which is based on the expl3 interface and will be described here.