TLaunch: a launcher for a TEX Live system

Siep Kroonenberg

February 1, 2017

This manual is for tlaunch, the TEX Live Launcher, version 0.5.1.

Contents

Chapter 1

The launcher

About this document

This document is about the TEX Live launcher. The first chapter describes the launcher in general.

The second chapter describes the launcher-based installation at the RUG, or Rijksuniversiteit Groningen, as an example of what can be done with a launcher-based installation.

Finally, since many TEX developers spend as little time as they can help in a Windows environment, I added an appendix with Windows background information.

1.1 Introduction

I designed the TEX Live launcher for the Windows TEX Live installation on our university network, which contains TEX Live itself plus several other TEX-related applications. The launcher provides a single access point to all this software, and to related local and online resources.

The launcher also takes care of configuration: at first run, TEX Live is added to the searchpath, and relevant filetype associations are set up.

This means that the launcher rather than the classic TEX Live installer

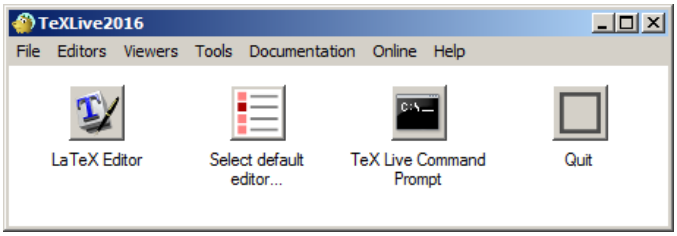

Figure 1.1: A possible Launcher window

takes care of all Windows-specific configuration. Which is a good thing if the TEX Live directory tree is on a shared network drive, or if a TEX Live installation has to be copied to many workstations.

Filetypes, menus and buttons and pre- and post configuration scripts are defined in a standard Windows ini file. The format of this file is described in section 1.4 of this chapter. The supplied ini file provides something more or less equivalent to a classic Windows TEX Live installation.

Although at the moment the launcher contains no support for localization, most of the user interface strings are defined in the ini file and can be replaced with strings in other languages.

1.2 Modes

1.2.1 Normal mode

In a normal run, the launcher displays a window with a menu or a series of buttons or both, see figure 1.1. For anything launched via these controls, TEX Live is prepended to the process searchpath irrespective of the system- or user searchpath; see Appendix A.3.5 on Windows searchpath handling.

1.2.2 Initializing

On first run, tlaunch creates file associations and prepends TEX Live to the user searchpath, see appendix A.3.5. Such configuration allows users to start up their LATEX editor by double-clicking a LATEX file in their file manager, bypassing the launcher altogether.

On first run, the launcher also creates a renamed copy of itself and a renamed modified copy of the configuration file to a directory inside the user's profile. This pair of copies serves as *forgetter*, see below. tlaunch registers this forgetter as an uninstaller under Windows.

1.2.3 Forgetting

Since a network- or multi-user TEX Live installation can be uninstalled by others, it is desirable that the configuration done on first run can be reversed without the presence of the original TEX Live. The forgetter takes care of this.

The launcher detects from its location whether it should run normally or as forgetter.

It is also possible to undo configuration from within a normal run if the ini file defines a button- or menu control for it.

1.3 Using scripts

External scripts may run on demand, as the action associated with a button or a menu control; see section 1.4.6. Scripts may also run automatically, as supplemental initialization or cleanup; see the `pre_config`, `post_config` and `pre_forget` variables in table 1.1. Examples uses:

- Forgetting a previous release of TEX Live before configuring the current one. This only makes sense for a centrally-managed TEX installation, where it is known what TEX installation came before.
- Giving TEXworks some spelling dictionaries
- Writing configuration data for non-TEX Live components
- Generating or updating the XeTEX font cache, assuming that the font cache directory is user-writable
- Adjusting XeTEX font configuration, see Section 2.6

1.4 The ini file

The ini file defines the menu items and buttons of the graphical interface. These controls can start up GUI programs or run utility scripts, or run some predefined functions. The ini file also defines filetype associations, and may define scripts for doing additional configuration and cleanup.

1.4.1 Location

One option is to place both the binary and the ini file in the root of the TEX Live installation. Another is to place the binary in TEX Live binary directory, *`tlroot`*`/bin/win32`, and the ini file where `kpsewhich` can find it, *e.g.* in *`tlroot`*`/texmf-config/web2c`.

The binary and the ini file should have the same first name.

1.4.2 Encoding

The launcher tries to guess the encoding used, and accepts ASCII, UTF-8 and Windows' UTF-16, with or without BOM. If all else fails, it tries ANSI with the system default code page.

1.4.3 Syntax

The ini file is a regular Windows ini file with sections, definitions and comment lines.

	DEFAULT	MEANING
`tlroot`	predefined	root of the TEX Live installation
`version`	predefined	release year
`tlperl`	predefined	path of the built-in Perl binary
`tlwperl`	predefined	same for the GUI binary
`java`	predefined	Java binary, if found
`tlconfig`	required	directory for the launcher's user data, see *e.g.* sections 1.4.4 and Appendix A.4.2
`tlname`	TeX Live %version%	used for *e.g.* window title and uninstaller 'DisplayName'
`customed_progid`	`TL.customed`	Filetype for external, user-selected editor, see section 1.4.5
`pre_config`	empty	optional program or script to be run before first-time initialization
`post_config`	empty	optional program or script to be run after first-time initialization
`pre_forget`	empty	optional program or script to be run before undoing initialization
`announce`	empty	optional text to display

Note. All process environment variables, *e.g.* `%appdata%`, are accessible while the launcher parses the ini file. Variable names are case-insensitive.

Table 1.1: Strings with a special meaning in the ini file

- A section starts with a line containing the section name enclosed in square brackets '`[`' and '`]`'. It ends at the start of the next section or at the end of the file.
- A definition line consists of a line *`key=value`*.
- A comment line starts with '`;`'.

The ini file is processed in one go, which means that everything must be defined before it is used. The ordering of the list below of possible sections satisfies this requirement.

However, it is not necessary that everything that is referred to actually exists; if a menu- or button control refers to a non-existent file, the control is quietly left out, and if the `COMMAND` of a filetype refers to a non-existent file the filetype registry key is created but remains empty.

The ini file can contain the following sections:

1.4.4 The Strings section

In this section arbitrary strings can be defined to be used later during parsing. The names of the strings are case-insensitive, their values are not. Various strings have a special meaning, see Table 1.1. Their values may be predefined, *i.e.* they are set by the launcher before processing the ini file, and should not be tampered with; they may be required, or they are allowed to stay empty.

At least `TLCONFIG` should be defined in the ini file. This is the directory for launcher user files such as the forgetter. A few suggestions:

- `%UserProfile%\.texlive%version%\tlaunch`, *i.e.* under the common root of `%TEXMFVAR%` and `%TEXMFCONFIG%`
- `%appdata%\tlaunch\%version%`; see Appendix A.4.2
- or maybe `%localappdata%\tlaunch\%version%` if TEX Live is installed on the local system

1.4.5 Sections for filetype associations (FTAs)

In Windows, an extension is associated with a filetype and a filetype is associated with a program. An extension can also have alternate filetypes associated with it, which may show up if you right-click a file and select 'Open with...'. More on filetypes in appendix A.3.2.

A filetype section has as name the string '`FT:`' followed by the filetype name. An example of a filetype section:

```
[FT:TL.TeXworks.edit.%VERSION%]
COMMAND="%tlroot%\bin\win32\TeXworks.exe"
;SHELL_CMD="%tlroot%\bin\win32\TeXworks.exe" "%1"
;ICON="%tlroot%\bin\win32\TeXworks.exe,0"
;NAME=TeXworks
EXTENSIONS=.tex .cls .sty
;PRIMARY=1
;PATH_PREFIX=0
```

The commented lines are optional and represent default values.

`COMMAND` is the command to start the program.

`SHELL_CMD` is the command to open a file. The default is `COMMAND` with '` "%1"`' appended.

`ICON` is the icon to be used in GUI file managers. The default is `COMMAND` with '`,0`' appended, without a space. If there is no such icon then a fall-back icon will be used; see the right-most icon in Figure 1.1.

`NAME` is only used for LaTeX editors, in the editor-selection window, see section 1.5. If not specified, it will be derived from the program filename.

`EXTENSIONS` is the list of extensions that should have the filetype as primary or secondary association.

`PRIMARY` is default 1. If set to 0, then the program only shows up in the Open with... dialog. Mainly of interest for the bitmap2eps utility. See appendix A.3.2 on primary and secondary filetypes.

`PATH_PREFIX` is default 0. If set to 1, then Windows will prepend TeX Live to the program's searchpath. The launcher will only do this if `COMMAND` is a bare or quoted filename, without options or parameters; see Appendix A.3.4.

1.4.6 Sections for utility scripts

A batchfile or command-line program can be declared in a utility-script section. If a button or menu item invokes such a script, standard output is intercepted and displayed in a dialog box when the script has completed. Standard error is also captured, but shows up only in the log file `%TEMP%\TeXLive_Launcher.log`. A splash text is displayed while the script is running. The default value for the splash text is 'Working...'. An example:

```
[SC:XeTeX-fontcache]
command=fc-cache -v
splash=Creating or updating XeTeX font cache
```

This item is included in the default ini file, although it may not work out of the box on a multi-user installation. Section 2.6 describes how to get it to work, and the `tlaunchmode` script, see section 1.6.1, will take care of this automatically.

1.4.7 The built-in functions

The following functions are available:

`FU:quit` Terminate the launcher

`FU:clear` Undo all configuration and terminate

`FU:initialize` Undo all configuration, terminate and restart. This forces re-initialization.

`FU:editor_select` See section 1.5 below

`FU:default_editor` See section 1.5 below

`FU:about` An About box

1.4.8 Menus and buttons

The visible interface of the launcher consists of an optional menu with drop-down submenus and an optional row of buttons. There should be at least one button or one submenu with one entry.

A submenu is defined in a section with name 'MN:' followed by the submenu name, and the row of buttons is defined in a section named 'BUTTONS'. A button- or a menu item has as key the string to be displayed and as value one of the following:

- A filetype. This invokes its COMMAND.
- A utility script. This also invokes its COMMAND.
- 'SO:' (Shell Open) followed by a document or URL. The document or url will be opened in its default program.
- A predefined function, see section 1.4.7.
- An arbitrary command.

In a submenu section, a sole '=' will produce a separator line. In the button section, it will do nothing.

Example buttons- and submenu sections (see section 1.4.6 for the SC:XeTeX-fontcache entry):

```
[MN:Tools]
Editor choice=FU:editor_select
TeX Live Command Prompt=%comspec% /T "TeX Live" /e:on
=
Generate/Refresh xetex font database=SC:XeTeX-fontcache

[MN:Documentation]
All TeX Live documentation by package=SO:%tlroot%\doc.html
TeX and LaTeX Q & A=SO:http://tex.stackexchange.com/

[BUTTONS]
LaTeX Editor=FU:default_editor
PostScript Viewer=FT:TL.PSView.view.%VERSION%
Quit=FU:quit
```

1.4.9 The General section

Here, three keywords are allowed:

`Filetypes` Allowed values are

- `none`: do not set or change filetype associations

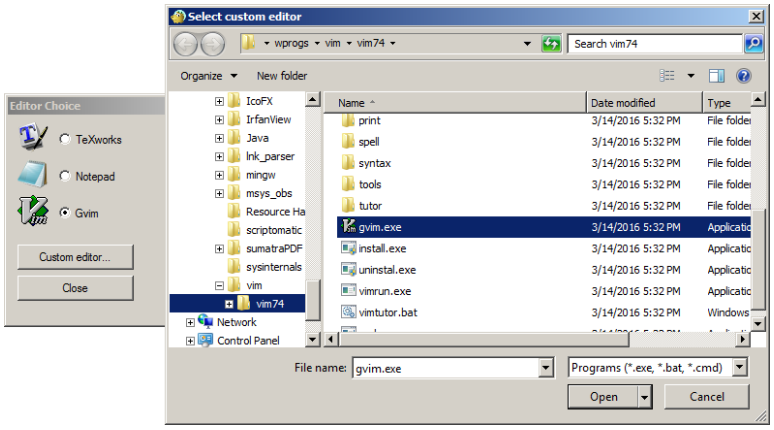

Figure 1.2: Editor selection with a file browser for a custom editor

- `new`: create filetype associations only if they do not override existing ones; default
- `overwrite` create filetype associations regardless of existing ones

`searchpath` Allowed values are 0 (leave searchpath alone) and 1 (add TEX Live to the searchpath; default). See appendix A.3.5. In any case, when a program is started from the launcher it will have TEX Live prepended to the process searchpath.

`keeptemps` Allowed values are 0 (delete temporary files; default) and 1 (keep them). This is a debug option for running external scripts. In most cases however, the contents of the temporary files are copied to the log file `%TEMP%\TeXLive_Launcher.log` anyway.

An example general section:

```
[General]
FILETYPES=new
SEARCHPATH=1
KEEPTEMPS=0
```

Since these are all default settings, one may as well omit this section.

1.5 Editor choice

If a filetype has `.tex` among its supported extensions, the launcher considers it an editor. On initialization, the first one becomes the default, unless `PRIMARY` is set to 0 and there is another candidate with `PRIMARY` 1. If in the `General` section `FILETYPES` is set to `none` or `new`, then an existing file association for `.tex` files will not be overwritten.

The function `FU:default_editor` invokes the default editor if there is one. The function `FU:editor_select` invokes a dialog for selecting a default editor; the options are the ones defined in the ini file, the current default and selecting one via a file browser dialog, see figure 1.2.

If the new editor is selected via the file browser, it will be assigned to the filetype `TL.customed` and this filetype will become the default for `.tex` files. It is possible to configure another filetype string in the ini file, *e.g.* one which includes the `%VERSION%` string.

Appendix A.3.3 explains why a `.tex` file might still get opened in the previous editor.

1.6 Launcher-based installations

It seems possible to do away with much of the Windows-specific code of the current installer. To this end, I added install and uninstall options to the launcher. Installation merely means creating a Start menu entry for itself and to register itself as uninstaller.

In installation mode, it is assumed that the launcher and its ini file are already in place as part of the regular TEX Live installation.

Installation and uninstallation are triggered by command-line parameters:

`user_inst` Install the launcher for a single user

`admin_inst` Install the launcher for all users

`uninst` Undo installation but leave the TEX Live directory tree alone

`uninst_all` Undo installation and remove the TEX Live directory tree. This is the only option of these four which touches the TEX Live installation itself.

If there is a forgetter for the current user, both uninst options will run it. A command-line option `silent` will ensure that the forgetter will run without user interaction.

Within the launcher, there are corresponding functions `FU:uninst_all` and `FU:uninst` which can be assigned to a menu- or button control. If necessary, the launcher will pop up a UAC prompt and restart in elevated mode.

1.6.1 The tlaunchmode script

The included Perl script tlaunchmode can convert a local TEX Live installation between classic and launcher-based. Run with a parameter 'on', the script turns launcher mode on; with 'off' it reverts the installation to classic, and anything else prints a brief help message.

It aborts if admin permissions are required but missing.

It can be installed in the usual way: put it under the scripts subdirectory of some texmf tree, run `mktexlsr` on that tree, and copy in the *`tlroot`*`\bin\win32` directory `runscript.exe` to `tlaunchmode.exe`.

Although at the moment there is no mechanism to make the path- and file association settings in the ini file conform to those set during a classic installation, those orginal settings are restored when converting back to classic mode.

1.6.2 TEX Live Manager

Nothing special has been done for the TEX Live Manager. It can be assigned to a menu- or button control, although this makes little sense for a centrally managed network installation. If necessary it will automatically pop up a UAC prompt.

Chapter 2

The launcher at the Rijksuniversiteit Groningen

This chapter is about the launcher-based TEX Live installation at the RUG (Rijksuniversiteit Groningen) and the environment in which it operates.

Our TEX Live installation resides on the network. The TEX Live Launcher is available via the start menu, and anybody on the university network can click the launcher shortcut and start using TEX.

The files in the `tlrug.zip` zipfile are tidied up versions of those of the RUG installation: compared to the files actually in use, they benefit from hindsight and omit ugly workarounds for specific local problems.

2.1 Historical

An earlier solution for TEX Live was implemented with a patchwork of scripts in various languages. An initialization script created filetype associations, Start Menu shortcuts and modified the searchpath. It made use of the built-in Perl and its TEX Live modules.

However, some people got confused that the generated TEX Live menu

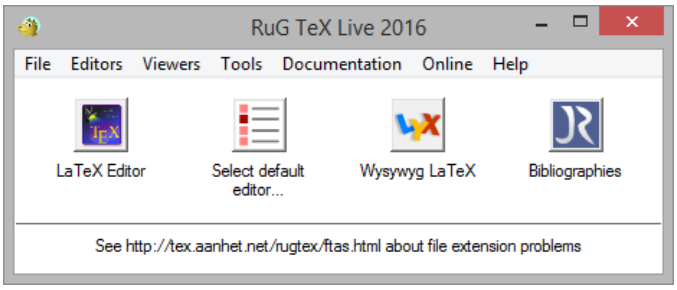

Figure 2.1: The launcher at the Rijksuniversiteit Groningen

was in one place, *viz.* in the conventional Start / Programs menu, and the initializer script in the centrally maintained 'RUG menu'.

When between 2013 to 2014 the university transitioned to centrally managed desktops, using the RES workspace management system[1], I created the launcher. For speed of development, a first version was written in the AutoIT[2] scripting language. AutoIT has good access to Windows' internals, and comes with a utility which packages a script with a AutoIT runtime into a self-contained executable. The AutoIT versions did not use an ini file; everything was coded into the script itself.

For the 2015 TEX Live release, I finally had a usable C version, in which all configuration was read from an ini file.

2.2 RES desktops

The RES system synthesizes a desktop or workspace for the user on logon. Personal settings are selectively captured, stored in a database and restored on re-logon; the settings part of RES can be considered an alternative implementation of the roaming profiles described in Appendix A.2.

This desktop is also available remotely, which works reasonably well most of the time.

For TEX Live, I submit a wish list of settings to the workspace management people, and they enter everything into the RES system. Unfortunately, the RES system has its quirks, and what I expect to happen is not always what actually does happen. But this is not the place to expand on my trials and tribulations with RES.

2.3 Components of the RUG TEX installation

Various third-party programs are incorporated into our TEX Live installation. Below are some details.

Most TEX-related software does not have deep hooks into the system. Even if applications require elevated permissions to install, usuallly they can simply be copied to another location and work fine from there. This is the case with *e.g.* TEX Live itself and with TeXstudio.

The add-ons are:

- Additional editors: TeXnicCenter and TeXstudio
- The pdf viewer SumatraPDF
- The Java-based bibliography manager JabRef

[1] Real Enterprise Solutions, `https://res.com/`

[2] `https://www.autoitscript.com/`

- The epspdf GUI with bundled single-file Tcl/Tk runtime
- The pseudo-wysywyg LyX LaTeX editor

There are also controls for:

- browsing the TeX Live installation
- a command-prompt with TeX Live as the first directory on the search-path
- generating or updating a XeTeX font cache; see Section 2.6
- some manuals from the TeX Live installation
- links to TeX-related websites

Such menu items are simply created by single lines in the ini file.

There are no controls for the TeX Live manager or for uninstalling.

Figure 2.1 shows buttons for some of the additional software. Along the bottom of the window is an announcement text, which is an optional string item in the ini file. The Help menu item opens a help text in the configured default text editor – probably Notepad.

2.4 Directory organization

There is a directory under the TeX Live root containing all the extras. There is another subdirectory for the various scripts. All paths in both the ini file and the scripts are relative to the root of the installation.

I did not put anything into the existing subdirectories of the TeX Live installation, and put both the binary and the ini file in the root of the installation.

2.5 Fixes for add-ons

Some of the add-ons needed a bit more work than just copying the installed program directory to its place under the TeX Live root:

2.5.1 TeXnicCenter

The original university installation was based on MikTeX, with TeXnicCenter as editor. In 2008 I replaced MikTeX with TeX Live, and TeXnicCenter with the more up to date TeXstudio editor. Since I did not want to force anyone to switch editors, I also kept TeXnicCenter around.

While TeXnicCenter can autoconfigure itself nicely for MikTeX, it asks TeX Live users a series of questions about what is where. Since many users

are only vaguely aware of directories beyond their home directory, I wrote a vbscript which emulates the MiKTeX autoconfiguration for TEX Live and avoids awkward questions.

2.5.2 TeXstudio

TeXstudio autoconfigures itself fine, but there were still two problems:

1. By default, it checks whether there is a more recent version, while users are not in a position to do an update.

2. With our current desktop management software, programs do not get the user searchpath appended to their process searchpath.

The first problem is solved during first-time initialization of the launcher. If there is no TeXstudio configuration file, then one is created with just the setting not to check for updates. If there is one, the update check option is edited to be off. Either way, there is no impact on other settings.

A solution for the second problem is described in Appendix A.3.4.[3]

Note that TeXworks, Dviout and PSV[iew] are invoked via the TEX Live runscript wrapper, which takes care of the searchpath, among other things.

2.5.3 SumatraPDF

In the absence of any registry settings, SumatraPDF assumes that it is a portable setup, and tries to write user configuration to its own directory. Its developers informed me what registry setting would convince SumatraPDF otherwise, so that it would write its configuration data to the user's profile.

Checks for updates are disabled in a similar way as for TeXstudio.

2.5.4 LyX

On first start, LyX can take several minutes to take stock of its environment and figure out what it can use. During this time, it is not even clear that anything is happening.

To prevent this delay, the post-config script copies a pre-generated configuration directory to the user's profile.

In the LyX installation itself, in the file *lyxroot*`\Resources\lyxrc.dist`, the setting `\path_prefix` has been rephrased to only contain paths relative to `$LyXDir`, which is the root of the LyX installation.

[3] In the current RUG installation, TeXstudio is started instead via a small wrapper program.

2.6 Moving the XeTEX font cache

At the time of writing, the default location of the XeTEX font cache in TEX Live is `$TEXMFYSVAR/fonts/cache`, *i.e.* *tlroot*`/texmf-var/fonts/cache`. In a multi-user or network install, this location is not user-writable. Since this was a problem, a line

```
FC_CACHEDIR = $TEXMFVAR/fonts/cache
```

in the file *tlroot*`/texmf.cnf` moved the cache to a user-writable location.

Since I generate the TEX Live installation on a Linux system, the configured TEX Live font paths in `$TEXMFSYSVAR/fonts/conf/fonts.conf` do not match the target installation. Of course this generated file can be hand-edited, but a less error-prone solution is to leave this file alone and move the font configuration files also to a user-writable location with another line:

```
FONTCONFIG_PATH = $TEXMFVAR/fonts/conf
```

in *tlroot*`/texmf.cnf`. A per-user font configuration is then created by a line

```
"%TLROOT%\tlpkg\tlperl\bin\perl.exe"
  "%TLROOT%\tlpkg\tlpostcode\xetex.pl"
  install "%TLROOT%" skip_gen 2>NUL
```

(one line) in a post-config script, see section 1.3.

The last parameter , *viz.* `skip_gen`, suppresses actual cache generation, since that might take quite some time. Any non-null value would have had the same effect.

2.7 Fixing non-roaming filetype associations

Before RES (see Section 2.2), and before the launcher, the TEX Live-related filetype associations would not follow the user from desktop to desktop; see also Appendix A.3.2. This was solved with a batchfile which restored missing file associations. By placing a shortcut to this batchfile in the Start / All Programs / Startup menu, this batchfile would automatically run at logon. However, RES made this workaround unnecessary.

For situations where non-roaming file associations are still a problem, I intend to add a 'reassoc' mode to the forgetter. The forgetter is already in place and knows how to parse the ini file. With the reassoc option, the forgetter will silently recreate missing filetypes. If TEX Live or the launcher is missing it will do nothing. There will be an option in the ini file to enable or disable creation of such a shortcut to the forgetter, in its role as rememberer.

Appendix A

Windows issues

A.1 User and system

Microsoft has wised up a lot security-wise. When Windows XP appeared, the line of Windows 9*x* windowses made place for slightly crippled versions of the NT-based professional editions. Even for Home editions there is now a separation between per-user and system-wide settings and files. Since Windows Vista, this separation is much more strictly enforced, and even administrators have to take an extra step, such as clicking yes to a UAC[1] prompt, before they can do anything deemed risky.

A.2 Roaming

On a Windows domain network, it can be arranged that users have more or less the same desktop, whatever computer within the network they happen to be working on. This is accomplished by either 'folder redirection', *i.e.* defining network locations for certain dedicated directories (see 'Known Folders' in Appendix A.4), or by copying user data back and forth between workstation and network on logon and logoff. There may also be a dedicated network share for user documents which is accessible from any computer, and which may do double duty as home to redirected folders.

A.3 Windows configuration

Some Windows configuration can only be stored in the registry, in particular file associations, see section A.3.2 below, and environment variables, including the searchpath.

Other configuration can be stored either in the registry or in configuration files, at the discretion of the developer.

[1] User Account Control

A.3.1 Registry locations

We have to deal with three locations or hives within the registry:

- `HKEY_CURRENT_USER`, or HKCU in short, for user-specific settings
- `HKEY_LOCAL_MACHINE`, or HKLM in short, for system-level settings
- A third hive, `HKEY_CLASSES_ROOT`, or HKCR, will be described in the next subsection.

A.3.2 Filetype associations

The basic idea is that an extension has a default filetype or ProgID, and this filetype in its turn can define commands such as open, edit, view or print. For example, the extension `.jar` has as its filetype or ProgID `jarfile`, and for the ProgID `jarfile` the open command is defined as *e.g.*

```
"C:\Program Files (x86)\Java\jre1.8.0_45\bin\javaw.exe" -jar "%1" %*
```

Extensions can also have secondary file associations. These are the ones showing up when right-clicking a file and selecting 'Open with…'.

Filetype associations exist in per-user and system-wide flavors:

- User-specific filetype associations are stored in HKCU\`Software\Classes`
- System-level filetype associations are stored in HKLM\`Software\Classes`
- The effective filetype associations are stored in `HKEY_CLASSES_ROOT` or HKCR, which is a runtime merge of the above two branches. Settings in HKCU override corresponding settings in HKLM.

For example, the link from `.jar` to `jarfile` is can be read from the HKCR\`.jar` key and the link from `jarfile` to actual commands from the HKCR\`jarfile` key.[2]

Secondary filetype associations can be stored in a subkey of the extension subkey, either the `OpenWithList` (obsolete) or the `OpenWithProgIds` subkey.

Unfortunately, these user-level filetype associations do not roam. With the RES desktop management system, this is no longer a problem for us. But a future version of the launcher will contain a workaround for other environments; see section 2.7.

[2] In some situations, reading from HKCR proved not entirely reliable. Therefore, TeX Live always explicitly checks first HKCU and if necessary HKLM.

A.3.3 UserChoice

Apart from this scheme, choices made by the user in the 'Open with...' dialog are stored under subkeys of HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts*.ext*, the exact subkey(s) depending on the Windows version. Such entries should have priority over the ones described above, and do roam.

Windows 8 and later may pop up a dialog asking with what program to open a file even if there is already an answer in a Software\Classes key. The answer will be a preference stored under the FileExts key mentioned above. This should alleviate the problem of non-roaming filetype associations.

The launcher will not touch these registry entries.

A.3.4 Application registration

An application which is registered may have a better chance to be listed as an alternative in the 'Open with...' dialog.

The currently recommended way to register an application is under the SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths*basename* key (basename including .exe file extension). The default entry of such a key is the full path of the file. Therefore, only one file with a given basename can have such a registration entry.

The other entry of interest under this key is 'Path', which is a searchpath fragment that should be prepended to the regular searchpath, just for this program. This prefixing happens if the program is opened by Windows Explorer. The path prefix is ignored if the program is started from the command-line or from the launcher.

For filetypes defined in the launcher ini file, the launcher creates an application registration key for the associated program, but only if the COMMAND field is just a filename or -path, with or without quotes.

Applications can also be registered under HKCR\Applications*basename*, but here is no option to set a searchpath prefix. Microsoft considers this location obsolete.

A.3.5 The searchpath and other environment variables

Environment variables are also stored in the registry: per-user variables in HKCU\software\environment, system environment variables in HKLM\System\CurrentControlSet\Control\Session Manager\Environment.

For the searchpath, the effective searchpath consists of the system %Path% variable, with the user %Path% variable *appended* if it exists, with an intervening ';' of course. Therefore, *directories on the system searchpath have priority over the user searchpath.*

Other environment variables from these registry locations behave as expected: the user version has priority if both exist. Note that the names of environment variables are case-insensitive.

Various pieces of system information, such as `COMPUTERNAME` and `CommonProgramFiles`, are not explicitly stored as environment variables, but are nevertheless available as such.

A.4 Windows Known Folders

Windows has an elaborate and ever expanding system of 'Known Folders': *e.g.* Program Files, ProgramData, User Profile (HOME directory), Start Menu, Desktop, Documents, Downloads, History, SendTo, Templates, Administrative Tools, Pictures, Music, Videos, Account Pictures, Cookies, Favorites, and dozens more, many of them both in a system- and a user version.

Sometimes a known folder is not a real folder but a virtual one, and for some known folders, 64-bits applications and 32-bits applications view things differently.

There are APIs which associate known folder identifiers to actual directory paths.

With Windows Vista a new API has been introduced, and the old one declared obsolete. One change for the better: the directory name 'Documents and Settings' has been replaced with simply 'Users'.

This API also gives access to other properties, such as the localized names shown in Windows Explorer, such as 'Programme' instead of 'Program Files', or 'Gebruikers' instead of 'Users'.

Microsoft recommends to avoid hard-coded paths, and to place everything under known folders instead. One benefit is at least that files in known folders have a better chance of surviving system upgrades.

A.4.1 Programs

The directory reserved for programs is normally `c:\Program Files`. On 64-bit systems, this directory is reserved for 64-bits programs, and `c:\Program Files (x86)` for 32-bits programs. Directories under one of the Program Files folders are automatically write-protected.

A.4.2 Configuration- and data files

There are also preferred locations for per-user- and system-wide program data. For Windows Vista and later these are:

- For user-specific settings, typical locations are `%appdata%`, which is usually `%userprofile%\%appdata%\roaming`, and `%localappdata%`,

which is usually %userprofile%\%appdata%\local. For a networked workstation, %appdata% may be relocated or backed up to a location on the network, and be available to the user on any workstation.

- For system-wide settings there is %ProgramData%, usually C:\ProgramData.

On Windows Vista and later, these directories normally do *not* have spaces in their path. They are by default also hidden.

A.4.3 TEX Live choices

It is clear that TEX Live does not try very hard to conform to all Microsoft's recommendations, and I do not think that it should. A couple of obvious advantages of the existing default path C:\texlive*yyyy*[3] are the absence of spaces, and the locale-independence of the path seen in Windows Explorer.

In this default location, TEX Live will not be write-protected automatically. For a system-wide installation, the TEX Live installer itself will take care of this.

As to user files, it should be ok to put $TEXMFCONFIG, $TEXMFVAR and tlaunch's %TLCONFIG% under %appdata% or %localappdata%. For the first two, this could also be done after the fact with a couple of lines in *tlroot*/texmf.cnf, *e.g.*:

```
TEXMFVAR = $APPDATA/texlive2016/texmf-var
TEXMFCONFIG = $APPDATA/texlive2016/texmf-config
```

There they will be hidden, but most users will never interact directly with files in these directories anyway.

[3] Regular users can create directories in the root directory of the C:-drive, even though they cannot create regular files there.