TEXMFSTART

Hans Hagen — 2003/2005

This manual is about a small (RUBY) script that can be used to run a script or open a document which is located somewhere in the `texmf` tree. There is not so much to tell about `texmfstart`, so we stick to demonstrating how to launch the program.

Start the `texexec` PERL script:

```
texmfstart texexec.pl --pdf somefile
```

Start the `pstopdf` RUBY script:

```
texmfstart pstopdf.rb --method=3 cow.eps
```

Or without suffix:

```
texmfstart texexec --pdf somefile
texmfstart pstopdf --method=3 cow.eps
```

The suffixless method is slower unless the scripts are known. For familiar CONTEXT scripts it's best not to use the suffix since this permits us to change the scripting language.

You can also say:

```
texmfstart --file=pstopdf --method=3 cow.eps
```

When locating a file to run, several methods are applied, one being `kpsewhich`. You can control the path searching by providing a program space, which by default happens to be `context`.

```
texmfstart --program=context --file=pstopdf --method=3 cow.eps
```

The general pattern is:

```
texmfstart switches filename arguments
```

where `switches` control `texmfstart`'s behaviour, and `arguments` are passed to the program identified by `filename`.

```
texmfstart showcase.pdf
```

This will open the document `showcase.pdf`, when found. The chance is minimal that such a document can be located by `kpsewhich`. In that case, `texmfstart` will search the tree itself.

Given that it is supported on your platform, you can also open a PDF file on a given page.

```
texmfstart --page=2 showcase.pdf
```

On MS WINDOWS the following command will open the PDF file in a web browser. This is needed when you want support for form submission.

```
texmfstart --browser examplap.pdf
```

When `kpsewhich` cannot locate the file, the following environment variables will be used:

`RUBYINPUTS`	ruby scripts with suffix `rb`
`PERLINPUTS`	perl scripts with suffix `pl`
`PYTHONINPUTS`	python scripts with suffix `py`

`JAVAINPUTS`	java archives with suffix `jar`
`PDFINPUTS`	pdf documents with suffix `pdf`

The script accepts a few directives:

`--program`	the program space where `kpsewhich` will search
`--verbose`	report some status and progress information
`--report`	don't run, only report the full location of the file
`--browser`	start the document in a web browser
`--page`	open the document at this page
`--file`	an alternative for providing the file
`--arguments`	an alternative for providing the arguments to be passed
`--direct`	run a program without searching for it's location
`--execute`	use RUBY's 'exec' instead of 'system'

This scripts evolved out of earlier experiments and is related to scripts and programs like `runperl`, `runruby` and `irun`.

You can create startup scripts by providing one of the following switches in combination with a filename.

`--make`	create a start script or batch file for the given program
`--windows`	when making a startup file, create a windows batch file
`--linux`	when making a startup file, create a unix script
`--stubpath`	destination of the startup file
`--indirect`	always use `texmfstart` in a stub file

The performance of the indirect call is of course less than a direct call. You can gain some time by setting the environment variables or by using a small TEX tree.

The script tries to be clever. First it tries to honor a given path, and if that fails it will strip the path part and look on the current path. When in bad luck, it will consult the environment variables. Then it will use `kpsewhich` and when that fails as well, it will start searching the TEX trees. This may take a while, especially when you have a complete tree, like the one on TEX Live.[1]

You can provide a path where the stub will be written. This permits tricks like the following. Say that on a CDROM we have the following structure:

```
tex/texmf-mswin/bin/texexec.bat
tex/texmf-linux/bin/texexec
tex/texmf-local/scripts/context/perl/texexec.pl
```

If we are on the main `tex` path, we can run `texmfstart` as follows:

[1] On my computer I use multiple trees parallel to the latest TEX Live tree. This results in a not that intuitively and predictable search process. The cover of this manual reflects state of those trees.

```
texmfstart --make --windows --stubpath=tex/texmf-mswin/bin \
    ../../texmf-local/scripts/context/perl/texexec.pl
texmfstart --make --unix    --stubpath=tex/texmf-linux/bin \
    ../../texmf-local/scripts/context/perl/texexec.pl
```

This will generate start up scripts that point directly to the PERL script. Such a link may fail when files get relocated. In that case you can use the `--indirect` directive, which will force the `texmfstart` into the stub file.

```
texmfstart --make --windows --indirect --stubpath=tex/texmf-mswin/bin \
    ../../texmf-local/scripts/context/perl/texexec.pl
texmfstart --make --unix    --indirect --stubpath=tex/texmf-linux/bin \
    ../../texmf-local/scripts/context/perl/texexec.pl
```

You can also use `texmfstart` to launch other programs that need files in one of the TEX trees:

```
texmfstart --direct xsltproc kpse:somescript.xsl somefile.xml
```

or shorter:

```
texmfstart bin:xsltproc kpse:somescript.xsl somefile.xml
```

In both cases `somescript.xsl` will be resolved and in the second case `bin:` will be stripped. The `--direct` switch and `bin:` prefix tell `texmfstart` not to search for the program, but to assume that it is a binary. The `kpse:` prefix also works for previously mentioned usage.

A convenient way to edit your local context system setup file is the following; we don't need to go to the path where the file resides.

```
texmfstart bin:scite kpse:cont-sys.tex
```

Because editing is happenign a lot, you can also say:

```
texmfstart --edit kpse:cont-sys.tex
```

You can set the environment variable `TEXMFSTART_EDITOR` to your favourite editor.

One of the reasons for writing `texmfstart` is that it permits me to write upward compatible scripts (batch files), so instead of

```
texexec --pdf somefile
texexec --pdf anotherfile
```

I prefer to use:

```
texmfstart texexec --pdf somefile
texmfstart texexec --pdf anotherfile
```

A bit obscure feature is triggered with `--iftouched`, for instance:

```
texmfstart --iftouched=normal.pdf,lowres.pdf \
    downsample.rb --verylow normal.pdf lowres.pdf
```

Here, `downsample.rb` is only executed when `normal.pdf` and `lowres.pdf` have a different modification time. After execution, the times are synchronized. This feature is rather handy when you want to minimize runtime.

There are a few more handy features built in. The reason for putting those into this launching program is that the sooner they are executed, the less runtime is needed later in the process.

Imagine that you have installed your tree on a network attached storage device. In that case you can say:

```
texmfstart --tree=//nas-1/tex texexec --pdf yourfile
```

There should be a file `setuptex.tmf` in the root of the tree. An example of such a file is part of the CONTEXT distribution (minimal trees). This feature permits you to have several trees alongside and run specific ones.

Another feature is conditional running. We need this for instance when we do runtime graphic conversions. By checking beforehand if the original file has changed, we can prevent redundant runs. This feature is used in the resource library tools.

```
texmfstart --iftouched=foo.bar,bar.foo convert_foo_to_bar.rb
```

There are a few more (experimental) features which we will describe in due time.