# A New Approach to the TEX-related Programs: A User-friendly Interface

Sergei V. Znamenskii and Denis E. Leinartas
Department of Mathematics Krasnoyarsk State University, Svobodnyi prospekt 79, 660041 Krasnoyarsk, Russia
znamensk@math.kgu.krasnoyarsk.su, den@math.kgu.krasnoyarsk.su

**Abstract**

Various TEX-related programs such as TEX, METAFONT, BM2FONT, DVISCR, DVIPS, BIBTEX, and others have to use sophisticated command line options and configuration files to work cooperatively. Modifying the configuration of such a complicated system becomes a problem. A good solution is a hypertext-based language based on an intellegent shell which provides an easy interface for TEX-users.

The number of various configuration options to use in a command line or in environment to start a TEX-related program has become somewhat enormous. The appropriate choice of an option set to use depends in a rather sophisticated way on hardware configuration, desirable interface and a lot of installed software-specific features.

Therefore the proper modification of a TEX system configuration becomes a problem to the ordinary user even if he uses some shell to facilitate the work. There exists a wide variety of TEX shells and different users have different opinions on the subject of which shell is better. The TEXShell V2.7.1 by von Jörgen Schlegelmilch (available on CTAN) is currently the most popular non-commercial shell in Russia to run under DOS. It provides a convenient user interface. For example, the user can change printers by just selecting appropriate item in the menu system; change any command line options; save changed configurations completely or partially; use various configurations for different directories; to run TEX, preview or print a file; and other actions – by just hitting a 'hot' key from the internal editor. The user may find and correct errors in a source file(s) easily and get complete hypertext help on LATEX and shell functions. The user can switch the color selection of TEX commands and control symbols, add any custom batch file or program to the user menu from inside the editor, or use his favorite external editor, etc. The shell supports the new emTEX directory system under DOS and OS/2 and uses less then 1k of memory when TEX or the previewer is executing. The 4TEX shell (last release 3.26) provides more-or-less similar features, but TEXShell seems to be a bit more easy to use, especially with a set of different configurations. Does anybody really need more?

The TEX user certainly can be happy working with the TEXShell or any other convenient shell with the same facilities. Problems arised in the case he needs to:

- control, on the screen, the margins to be used on printed paper;
- install another printer;
- use some new emTEX features (such as booklet printing);
- use one of the very nice extra utilities such as BM2FONT or MFPIC which depends on printer font set; and/or
- repeat all the actions above simultaneously.

In such cases, the user will have to look into the documentation and perhaps write new configuration and batch files again and again without any hope of getting a final version. If you only change printers, for example, You usually have to change the path to write BM2FONT and MFPIC `.pk` font files, recalculate the extension set to start GFTOPK execution, defining resolution, font directories, screen margins and scaling options for the previewer. Moreover, in order to determine the correct way configuration, you need to collect information from a set of documentation files with the size of hundreds of kilobytes.

None of the existing shells can do it automatically. It is a task for human intellect. And nobody is able to "teach" any existing shell to perform this kind of job. You should know all of the interrelations between a type of printer, a resolution set and mode for METAFONT to make the whole system work correctly. Moreover, the interrelations mentioned above may also depend on font directories, existence of `.pk` files and other machine-dependent conditions which you should check before you may

achieve success. The new version of the same program usually handles the environment and the command line options in different way. It is really difficult.

The experimental TEX shell was designed as emTEX-based, with the support of RFBR to make the following extremely easy:

- the use of any TEX–METAFONT interface-based packages such as MFPIC;
- the use of other packages such as BM2FONT;
- the change of printer driver or changing to another *svga* mode;
- the ability to see the same margins on screen as those on the printed page;
- different configurations for different kinds of jobs (different journals for example);
- writing all temporary files (`.log, .dvi`, etc.) to a special directory; and
- the support of all features of the current emTEX distribution.

The test version of the shell was created based on the TX sources by Ricard Torres (`torres@upf.es`) and which are available on CTAN (a very simple and effective shell for work with TEX under DOS) and successfully tested in the Krasnoyarsk State University, several institutions at Krasnoyarsk and in the Information and Publication Centre of Steklov Mathematical Institute (Moscow). After we replace the temporary Russian fonts with the standard set and probably some other changes as proposed by the project, the shell will be freely available from the RFBR Russian TEX server `ftp.tex.math.ru` via *anonymous ftp* as an add-on package to the current emTEX distribution.

Though it is not the purpose of this paper to describe our shell completely, we need to give some information on it. It contains an extremely small menu, which saves screen space allowing the user to see a maximum of the program finished before output. The menu size depends on the number of files to work with and the number of currently available commands. One available command is Set (hit the "S" key) to call a set configuration menu in which you can select (change) a printer resolution, *svga* mode, portrait or landscape mode, output directions, or printer behavior (for two-side printing, etc.). The format is normally selected automatically, but you can change it if Hit the `<E>` key (Edit) to call the default editor; hit `<T>` to TEX a file; hit `<V>` to preview the file; and so on. Usually the shell gives a prompt for the next command, and you can just press `<SPACE>` to approve the command. If you work with LATEX files, the default sequence for the first time to run will be `<T>_<T>_<V>_<E>`. With MAKEINDEX the sequence will be `<T>_<M>_<T>_<V>_<E>` and with MFPIC `<T>_<F>_<T>_<V>_<E>` where `<F>` starts METAFONT in the appropriate mode, starts GF2PK and generates the correct output files. Unfortunately there is no on-line help available in our shell at this time.

As soon as we announced that we were about to place this shell into the base of a non-commercial "Russian TEX" distribution, the biggest disagreement became what set of packages was to be supported. Almost everybody wanted to restrict the number or packages to use as much as possible, but insisted upon the inclusion of the packages he know best. How to resolve this discussion? The only way we can see is to make a TEX shell to use a widely variable set of packages which are easy to install and use – "unpack an play". The TDS (TEX directory structure) standard gives us a chance to make the packages extremely independent upon operating systems. The more we try to configure the shell for a new package, which is complex yet still easy-to-use, the more time we spend checking and correcting all of the places in the configuration files which affect the package behavior.

Thus we need a new approach to the shell and user interface. It should be a shell with some features of the human mind, or an "intelligent" shell.

What does that mean? Are we to develop a system of artificial intellect only to start TEX-related programs? Of course not! But we want our shell to be able to check logical conditions and change the behavior according to the current situation. This may be useful not only in the configuration but in the interface too. It isn't necessary to have a menu item "Print" if there is no `.dvi` file, as well as there is no need to run METAFONT if an `.mf` file does not exist. It would also be better if the shell would lead us through the menus, choosing the next reasonable item; i.e., after running TEX, the next action is previewing and after that it is another edition in the most cases. The user can agree with the shell by pressing `<ENTER>` or choosing something else if he wants. We believe such advantages makes work a bit more convenient.

In addition to this, we consider a fully detailed, context-dependent HELP as a very important part of the shell. Being concerned with TEX for about three years we have come to the conclusion that users sometimes want to have a more detailed explanation of every menu item than the existing shells support.

How can we improve this service? Nowadays hypertext structures are very much appreciated in

user guides on computer systems. Many shells provide reference information organized as hypertext. TEXShell has only the HELP mentioned above. But this is not context-dependent, so sometimes it is quite difficult to find the information you want. Besides, this is a separate part of the shell and the user would need to interrupt work to study what he needed to do next.

So we can say that the help is laid down into the base of the future shell described here. This is a hypertext-based shell. Each hypertext page contains all reference information about marked words. Every such word means an action. It may be running a program or changing its configuration as a new hypertext page which allows one to take the next step. To avoid this information, you are able to enter an "expert" mode and see nothing except for hypertext references. This seems to be quite easy to use such shell.

The other task this shell will have to solve is installation of any additional package which needs to be integrated into the current conglomeration of programs. There are many such packages for TEX now and there'll be many more in future. Each of them presupposes its own way of installation and configuration. Thus, any package which can use the common directory structure may be installed just by unpacking the appropriate archive containing a configuration file for our shell. This file should include all information about behavior of this package and have some machinery for configuring the package. The shell definitely cannot know all packages' names. That is why the configuration files should have a special extension, say TXC (for TEX Config).

Now we want to consider the system of configuration files for our shell in more detail. Besides the packages' configuration files mentioned above, the system must comprise the main *read-only* file.

It should be emphasized here that this file is the shell itself. We mean that the shell is only a program written for an interpreter of a new language which allows creation of hypertext with the functions of a shell.

The next standard file is a file which contains information about the current state of a shell. This includes some variables and technical data. The other file is user-defined, where one can put the name of the work directory, necessary format, type of printer and so on. One more file of the same type is the job configurations file. Here are the common options for a group of users which do similar work. Another important file describes various devices used by printers and some programs. And last, but not least, the station-independent file where common commands for all supported platforms options are collected. Having these files, one can be comfortable working on any station where this shell is installed. You can prepare your text on one station, compose it on the server, view on another station and print somewhere else.

We are not yet ready to describe the programming language which allows creation of such shells; we can just say that it was developed for purposes like this and we are still working on it. We only show one example of a very simple configuration file. It isn't a complete shell;however, it gives a glimpse of the new language under creation: As you can see from this example, this language is quite similar to TEX but digits and other special TEX signs are referred to as strings. One-line remarks preceded by % are allowed. The command `\newpage` defines the highlighted words in hypertext, followed by page and the hot-key which one can press as well as the 'Enter' key to achieve the next page. Language's devices provides various detail levels of reference information which one may use in the shell. It may be whole pages for a novice or simply highlightings for an expert. `\def` looks like TEX's and is very easy to understand. It defines new hypertext pages as are used in `\newpage`. There are some logical primitives for `\if` in addition to `\exists`. They give one an opportunity to compare dimensions like textttpt, in, `cm` and so on as well as numerals and strings. Standard environment variables defined above are available as `$name$`. For example `$mfmode$` means `canonbj` if you use Cannon Bubble Jet printer in your system. The command `\addnewpack` finds the configuration file of new package and reads it contents. This file should include everything about the package itself and all interrelations with other packages too.

The package configuration file can contain special commands describing what must be done *before* execution of the main commands of some other package(s) and *after* it. This looks like a possibility to avoid user problems of installation. Only unpacking will be required, even for a rather complicated package such as MFPIC.

We did not have the intention to describe a completed thing. The aim of this paper was to discuss a new approach to this problem. In a year we plan to put the first version of a new shell with a sources on a Russian TEX server to be freely available at least for non-commercial usage by *anonymous ftp*.

```
%%% A simple configuration file for the shell
\def{\TheHomePage} Now you can compose your file using
\newpage{TeX}{\TeXpage}{T}
\if \exists{$name$.dvi}
or view composed file by  \newpage{DviScr}{\dviscrpage}{V}
and print it on the printer \newpage{Print}{\printpage}{P}
\fi
\if \exists{$name$.aux}
\if \infile{$name$.aux}{\bibdata}
And now you can call \newpage{BibTeX}{\bibtexpage}{B} to
complete the bibliography.
\fi
\fi
%%% Below the approximate scheme of adding new packages is shown
%%% the command \addnewpack  adds found in the directory ../texmf/txc/
%%% files <package_name>.txc to the page \packages

\addnewpack

From the moment of the latest upgrade of our shell the
\newpage{New Packages}{\packages}{}
is appeared.
}

%%% The  Home Page  definition is completed.
%%% Now we should define the pages mentioned in it
\def{\TeXpage}{By pressing the word \newpage{\TeX}{\runshellscript}{}
you run TeX and provided there are no mistakes new file $name$.dvi will
appear.
\if \exists{$name$.dvi}
And you can \newpage{View}{\dviscrpage}{V} it.
\fi
You can configure  TeX on the page
\newpage{configuration}{\texconf}{}}
...
```

**Figure 1**: Sample configuration shell

**Acknowledgements**

This work was supported by Russian Foundation of
Basic Research grant 95-07-19400v.