An Overview to dvitool 2.0

by Jeffrey W. McCarrell
jwm@Berkeley.EDU

This file describes dvitool version 2.0, an interactive previewer for the output files produced by TEX. It does not describe TEX, nor any of the TEX derivatives, nor how to produce hard copy of your TEX files. The person who is responsible for maintaining dvitool, and who should be your first contact point to find out details of your local implementation is:

local-contact: your-name-here

If they are not able to answer your questions, you may direct them to me at the electronic mail address listed above.

Dvitool was designed to be an efficient, productive tool for creating TEX documents; a significant percentage of the development time was spent on the user interface. The resulting system is very flexible but it is also somewhat complex. There are three different approaches to understanding the system:

- this overview help file describes the general mechanisms that comprise dvitool. All of the major interfaces and features are described here. After reading this file you should be able to run any of the 70 odd commands that dvitool has to offer; you should understand how the interface that collects the arguments to the commands works; and you should understand how dvitool "attaches" a command to a sequence of keystrokes and where to look to learn how to change the default attachments.
- The command help-commands describes each of the commands; it can be run from the help item in the main menu which can be brought up by pushing the right mouse button while in the main window of dvitool.
- The help-variables command is similiar to help-commands except that it describes all of dvitool's variables.

Running Commands: Every command has a long, mnemonic name. For example, the command to repaint the DVI image so that more of it can be seen is called scroll-down. Every command can be executed by the command called exec, so the general way to run dvitool commands is to first run exec, whose job it is to run other commands, and then to run the command you wanted. But how does one run exec? Some commands are associated with, or "bound to", sequences of characters. In this particular case, dvitool knows that the pattern <ESC>x means "run the exec command". (The string <ESC> means the escape key on your keyboard, not the five characters '<', 'E', 'S', 'C' and '>'. Characters which are symbols for other characters are surrounded by angle brackets. Control characters will be represented in this document by a C, followed by a dash (-), followed by the character. So control–x is written <C-X>). One way to scroll down is to type:

<ESC>xscroll-down<RETURN>

Fourteen characters is a lot to have to type to run a command used as often as scroll-down, so dvitool provides a simpler way to do the same thing. Any command

can be "bound" to a sequence of 1 or 2 keystrokes. This is how `dvitool` knew that `<ESC>x` meant "run `exec`."

Well, that's all well and good, but how do you learn which commands are bound to which keys? There are two approaches:

- the command `bound-to` describes all of the keystrokes that will run the command given as its argument;
- the command `describe-key` goes the other direction and describes what command will be invoked by the keys given as its argument.

If a keystroke is bound to a command, it is called a "binding." A command can have many bindings, but a key can have only one. For those of you that like to have it all at once, the command `dump-bindings` lists each key combination and the command it invokes. This information is written in a file named `dvitool.commands` in your current directory so you can print it out and keep it handy for easy, if voluminous, reference.

Note that mouse input in this context is considered a keystroke. You can bind commands to mouse inputs just like you can bind commands to ascii characters. For more information on how to actually make your own bindings, see the help entry for `bind-to-key`. One way to do this is to type:

`<ESC>xhelp-c␣<RETURN>bin␣<RETURN>`

There is a yet another way to run a command: invoke it from a menu. The right mouse button is bound to a command that produces a menu of choices of other commands to run. If the command has a key binding, it is listed on the right side of the menu, the idea being that you can get a good idea of the available commands just by popping up a menu and as you become more familiar with the command you can note its key-binding and invoke it directly with a single keystroke (or mouse press) rather than the several actions menus require.

Arguments: Many of the commands take arguments; for example, the command `find-file` takes the name of a `DVI` file as its argument. To show you that it is waiting for you to type an argument, `dvitool` will change the cursor tracked by the mouse into the shape of a mouse. Should you decide that you don't want to run this command after all, you should type `C-G` to abort the command. In general, the abort character (which can be specified by the variable `abort-character`) is a good key to press when you are unsure what is going on and you want to get back to the top level. You'll know you're back at the top level when the cursor changes to it's default shape (usually a circle with a hole in its center).

Arguments have types like integer, string, and so forth. `Dvitool` usually gives a clue as to the type of argument it is expecting by changing the cursor in the message window. The cursor in the message window is not the cursor that tracks the mouse, but a cursor to let you know where the next characters you type will appear. Here's a table of all of the of argument types `dvitool` has, and the character that is displayed "under" the cursor.

type	*char*	*completion*	*type*	*char*	*completion*
command name	'c'	yes	integer	'i'	no
file name	'f'	yes	font name	'F'	yes
string	's'	no	literal string	'S'	no

variable name	'v'	yes	variable help index	'V'	yes
command help index	'C'	yes	overview help index	'O'	yes
others	'␣'	no			

The completion column describes whether or not each type is capable of the time saving capability to "complete" some or all of the argument for you. Completion compares the characters you've already typed to the set of possible choices and fills in any characters that are common to all the choices or, if what you have typed so far uniquely identifies a choice, it completes that choice. The space character invokes completion. `Dvitool` will do what it can and either complete the argument all the way or give you a message about why it failed.

Another special feature of completion is the ability to list all of the choices that match the input string you've typed so far, invoked with `?`. `Dvitool` uses its typeout mechanism to display all of the choices and then it waits for you to type some character to show that you've finished reading the list and you're ready to proceed. A space character or a mouse press will simply be eaten and ignored; other characters are acted on.

Startup File: One of the first things that `dvitool` does is look for a file of commands to execute whose name is "`.dvitoolrc`" *. It looks for this file in two places: in your home directory, and in the current directory. If a startup file exists in both places, `dvitool` first reads the one in your home directory, then the one in your current directory, so the commands in the startup file in the current directory have precedence over the file in your home directory.

The file should contain commands that you want `dvitool` to run every time it is invoked. Common uses of the startup files are to load key bindings, personal cursors, icons, etc. Comment lines begin with the sharp character ('`#`') and end with a newline. Blank lines are ignored. Every other line is expected to contain a `dvitool` command and arguments. It turns out that there are very few commands that make sense when run inside a startup file, so `dvitool` disallows most of them. Here is the complete list of commands allowed inside a startup file:

command	*description*
`bind-to-key`	change the command invoked by a key sequence
`cd`	change directory
`print`	show the value of a variable
`set`	change the value of a variable
`version`	show the version number

Since key bindings don't make a great deal of sense inside a startup file, the correct way to refer to a command is by its long mnemonic name. Any arguments the command expects should follow on the same line, separated by whitespace. The arguments should appear exactly as you would type them to `dvitool` interactively, with the exception an

* This isn't strictly true. `Dvitool` actually takes the name it was invoked with, prepends a '`.`' and appends `"rc"` to create the filename. So if your system wizard installed `dvitool` under a different name, or you have symbolic links to it, the name of your startup file will differ.

argument that has a space in it (or a tab) should be surrounded by double quotes. There are actually several translations that `dvitool` does just when it is reading arguments in a startup file and here is the complete list:

from	*to*	*comment*
`\\`	`\`	
`\"`	`"`	
`"`		double quotes are elided
`\n`	`<RETURN>`	ASCII 13, control-M

Now that all of the rules have been presented, it's time to look at a real example. These 4 lines are syntactically correct; were you to put them into a startup file in your home directory, `dvitool` would process them without error.*

```
set          init-cursor-file      ~/lib/cursors/$DVICURSOR
set          init-cursor-xhot      0
bind-to-key  cd                    \e\^D
bind-to-key  next-page-positioned  "\e␣"
```

Some points to note here:

- Every filename is always processed for `~` and `$` characters, so line 1 expands `$DVICURSOR` into the value of the environment variable named `DVICURSOR`. It is an error to reference an undefined environment variable.
- The `next-page-positioned` binding requires double quotes so the space character will be passed as an argument and not elided as an argument separator.

Comand Line Switches: There is only one:

- `-E` tells `dvitool` to try to use an existing `dvitool` to preview the first file name. If there are no `dvitool`'s running, then `-E` is a no-op. If there are other `dvitool`'s running, the `-E dvitool` will start up, send a message to a running `dvitool`, and then exit. This mechanism is most often used inside other programs to provide a simple way to invoke `dvitool`.
- The generic *Suntools* flags (they all begin with `-W`) are ignored by `dvitool`. Their effect is undefined, since `dvitool` has variables to perform nearly all of the same functions. The use of the `-W` flags is discouraged.

Help Summary: Here is a complete list of the commands which provide information and or help.

command	*description*
`ascii-of-selection`	show the ASCII representation of some `DVI` characters
`bound-to`	show all the key sequences which invoke a command
`describe-key`	displays the command which will be run by a key sequence
`dump-bindings`	writes a file of key bindings sorted by keys
`dump-commands`	writes a file of all the commands and their arguments

* Of course, you'd have to define the environment variable `DVICURSOR`, and create the cursor file or `dvitool` will complain.

`help-commands`	provides interactive help for each of the commands
`help-overview`	displays this file
`help-variables`	provides interactive help for each of the variables
`list-all-commands`	displays all of the command names interactively
`list-all-variables`	displays all of the variable names interactively
`print`	display the value of a single variable
`version`	displays the version number
`which-char`	shows the font position of the selection
`which-font`	shows the font of the selection

Bugs: Yes Virginia, there is such a thing as a free bug or two in `dvitool`. I personally don't think of them as bugs, but as facts of life. All of the bugs that I could correct without super-human effort have been squashed. Nonetheless, `dvitool` does, at times, behave in unexpected ways. Here is the known list:

- Syntax errors in generic *Suntools* command line arguments (they all begin with `-W`) are silently ignored. The use of these arguments is discouraged.
- If the mouse cursor is positioned over the message subwindow, all input (both keyboard and mouse input) is ignored. Always position the mouse cursor inside the large image window.
- When `dvitool` is waiting for you to type some input (when the mouse cursor has changed into an image of the mouse), any input to the namestripe (like to close or hide the window) is ignored until the local input is completed. Always make sure `dvitool` is back to the top level by typing your abort character (usually control-g) before attempting any operations in the namestripe.
- Due to resolution rounding, two identical hrules may be displayed differently at different places on the page. The difference will be at most 2 pixels.

Complex tools such as TEX and to a lesser extent `dvitool` take time to master; I have tried to ease that transition with consistent interfaces and with several levels of help, but I am always willing to hear suggestions, and even criticism. Good luck!

Berkeley, California — J. W. M.
December, 1986

`jwm@Berkeley.EDU`
`...!ucbvax!jwm`