GNU Teseq 1.1.1 Manual

by Micah Cowan <micah@addictivecode.org>

This manual is for GNU Teseq, version 1.1.1.

Table of Contents

1 Overview **1**
A Quick Example 1
Stripping out escapes 2

2 Invoking Teseq **3**
Interactive Mode 4

3 Output Format **5**
3.1 Text Lines 5
3.2 Control-Character Lines 6
3.3 Escape-Sequence Lines 7
Recognizing Escape Sequences 9
3.4 Label Lines 10
3.5 Description Lines 11
3.6 Delay Lines and Halt Lines 12

4 Color Mode **13**

5 The Reseq Command **14**
Reserved Line Prefixes 15

6 Standards **17**

7 Future Enhancements **18**
7.1 Localized Messages 18
7.2 Stateful Processing 18
7.3 Terminal Database Awareness 19

8 Contact and Information **20**

Appendix A Copying **21**

Index **28**

1 Overview

GNU Teseq (the author pronounces it: "tea" + "seek") is a tool for analyzing files that contain control characters and terminal control sequences, by printing these control sequences and their meanings in readable English. It is intended to be useful for debugging terminal emulators, and programs that make heavy use of advanced terminal features such as cursor movement, coloring, and other effects.

Teseq is useful for:

- creating animated, interactive demos to run on the terminal (see `https://asciinema.org/a/7445` for a video on how to do this),
- knowing the exact output of a program (Did it have spaces at the end of the line? Or maybe it contains invisible control characters?),
- examining a text file's contents unambiguously, a la `cat -v` or the `ed` program's `l` command (but with much more information),
- stripping control sequences from a text file, e.g. to produce a plain ascii text file from a typescript file generated by the `script` command (see example below),
- examining the invisible control sequences within a text file, that affect graphical formatting or character encoding, in order to understand how they work and where they appear in the file, or
- debugging graphical terminal applications, and terminal emulators (its originally-intended purpose).

GNU Teseq is free software (`http://www.gnu.org/philosophy/free-sw.html`). See Appendix A [Copying], page 21, for copying conditions.

A Quick Example

You can't beat a short example to demonstrate what a program does, so here goes (note: a video version of this example may be seen at the main Teseq project page (`http://www.gnu.org/software/teseq/`), or at `https://asciinema.org/a/7443`. Suppose you've got a program that writes the following output to a terminal.

Hi there, world

A simple text string, using a boldface font to render the first word.

Suppose that, after a moment or two, the program then replaced the final word "world" with the word "earth".

In order to achieve this effect, the program would have to send special controls to the terminal to ask it to start writing in bold text, and then to revert back to normal text for the rest. To replace the final word, it might have to issue a series of backspaces before issuing the replacement word. All of this will be handled transparently by the terminal, and you wouldn't see any of the special c aracters the program sent to the terminal—unless perhaps you convinced the program to write to a text file as if it were writing to a terminal, or ran the program under a terminal capture utility such as the `script` command.

You may be able to produce this effect yourself with the following shell command.

```
$ printf '\033[1mHi\033[m there, world'; sleep 1; \
    printf '\b\b\b\b\bearth\n'
```

If you were to examine the output from such a command with a text editor, it might look something like this.

```
^[[1mHi^[[m there, world^H^H^H^H^Hearth
```

Not very readable, is it? That's where Teseq comes in! When you run that gibberish through the `teseq` command with the default settings, you'll get the following output.

```
: Esc [ 1 m
& SGR: SELECT GRAPHIC RENDITION
" Set bold text.
|Hi|
: Esc [ 0 m
& SGR: SELECT GRAPHIC RENDITION
" Clear graphic rendition to defaults.
| there, world|
. BS/^H BS/^H BS/^H BS/^H BS/^H
|earth|.
```

Note that the special control sequences that tell the terminal to start and stop writing in boldface text are separated out on their own lines (prefixed with a colon '`:`'), and followed by an identification (prefixed with an ampersand '`&`') and a description of what it does (prefixed with a quote character '`"`').

The actual text appears in lines bracketed by pipe '`|`' characters.

The series of single-character backspace controls appear on a line prefixed with a period '`.`', identified by its identifying acronym (BS for BACKSPACE), and its control-key representation (Control-H).

The final word, "earth", is followed by a period just after the closing pipe symbol; this indicates a following linefeed (or "newline") character.

The `reseq` command may be used to reverse the procedure, accepting the above input and printing out the original set of escape sequences that produced it. See Chapter 5 [Reseq], page 14.

Stripping out escapes

The `teseq` command can be instructed to recognize, but not output, escapes encountered in the input. This is useful for stripping out the formatting strings that are invisible on display, but which get in the way during editing, or make the content inappropriate for inclusion in documentation, etc.

To sanitize a typescript file or other text file containing control sequences, try the following command:

```
$ teseq -EDLC your-file.txt | reseq - - | col -b > new-file.txt
```

The `-EDLC` options tell teseq not to output recognized escape sequences or identify them, and otherwise to abbreviate its output. Then, `reseq` takes the output from `teseq` and turns it back into raw text (minus the now elided escape sequences). Finally, `col -b` (a fairly standard Unix tool) interprets common control characters such as backspaces and carriage returns, resolving them into a text file that displays the same, but doesn't contain those controls.

2 Invoking Teseq

The format for running the `teseq` program is:

```
teseq options [input-file [output-file]]
teseq -h | --help
teseq -V | --version
```

If *input-file* or *output-file* is unspecified, or specified as '`-`', standard input/output is used. Output is written to standard output by default, but see the `-o` option.

`-h`
`--help` Print usage information on standard output and exit successfully.

`-V`
`--version`
Print the version number and licensing information for `teseq` on standard output and then exit successfully.

`-C` For control characters from the C0 set of Ecma-48 / ISO/IEC 6429, don't write the control-key representation, only the identifying acronym. For example, write the carriage-return/line-feed combination as

```
. CR LF
```

rather than

```
. CR/^M LF/^J
```

`-D` Don't print description lines (those beginning with '`"`').

`-E` Don't print escape-sequence lines (beginning with '`:`'). **Warning:** this results in loss of information, and in particular means that running the output through the `reseq` command won't reproduce the input.

Still, this option can be useful (in combination with `-L`) for those that don't care about the exact sequence of characters, or what their function is called, but just what their effects in the terminal are (those that Teseq understands). The output produced will describe what happened, but not how.

`-L` Don't print identifying labels (lines beginning with '`&`') for escape sequences.

`--color[=`*when*`]`
`--colour[=`*when*`]`
Colorize the output. WHEN defaults to 'always' or can be 'never' or 'auto'.

`-I`
`--no-interactive`
Don't put the terminal into non-canonical or no-echo mode, and don't try to ensure output lines are finished when a signal is received (see below).

`-b`
`--buffered`
Force `teseq` to use buffered I/O (see below).

`-t` *timings*
`--timings` *timings*
: Read timing information from *timings* and emit delay lines. This file must be formatted as if generated by '`script -t`' (for the `script` command from util-linux).

`-x`
: No effect. Accepted for backwards compatibility.

Note that there are no options for suppressing text lines ('`|`') or control-character lines ('`.`'), as there are for description or escape-sequence lines.

The `-L`, `-D` and `-E` options also have mnemonic equivalents of `-&`, `-"` and `-:` respectively, corresponding to the character prefixes for the lines they suppress; and `-C` has an `-^` equivalent, for the '`^X`'-style control representations it suppresses. However, while they may be more practical to remember, they will be less practical to type, since both `-&` and `-"` are apt to be interpreted as special by the shell, and must be quoted with a backslash (`-\&` and `-\"`) in order to pass them to the `teseq` command.

Interactive Mode

When `teseq` is started with a terminal as its input, it sets the terminal to non-canonical mode. That way, you can see real-time translation of input, as you type. If both input and output are terminals, then `teseq` will also turn local echo off, so that the characters you type will not interfere with the output you see. You can try it out by simply running `teseq` without any arguments. **Note**, this means that the control for indicating "end-of-file" (usually '`C-d`') will not be processed specially, but will be passed through to `teseq` like any other character. Use the interrupt character (usually '`C-c`') instead, or specify `--no-interactive` to disable this behavior.

When run in this way, characters typed as input are immediately translated and written out. The exception is that when an ESCAPE character is encountered, `teseq` must wait for the next few characters before writing anything, so it can decide whether to start an escape line or a control-character line.

When `teseq` has a terminal as its output, it is careful to ensure that it finishes output lines when it is stopped by a signal. If it was in the middle of writing a text line, it will write the closing pipe character '`|`', followed by a newline. If it was in the middle of trying to determine whether it's in a valid escape sequence or just an escape character followed by other characters, it will assume the latter case, and translate all the characters it has seen so far.

To force `teseq` to behave as if it's not connected to a terminal (that is, to refrain from ensuring lines are finished, or setting non-canonical/no-echo mode), use the `--no-interactive` (`-I`) option.

The `teseq` program does not take care to finish lines when the output is not a terminal.

Whether or not `teseq` is connected to a terminal, it uses unbuffered I/O by default, unless the input is an ordinary file. This is so that each character may be processed as soon as it's seen. However, this can result in much longer processing time overall. To force `teseq` to buffer its input and output, use the `--buffered` (`-b`) option.

3 Output Format

Teseq produces output that is primarily intended to be read by humans (from input that is usually not). For this reason, Teseq output follows the following principles:

- Displays for different sorts of characters and sequences are displayed on distinct lines, and are easily distinguished by the initial character.
- Every input character should be unambiguously visible to the user in the output (except, of course, when the user explicitly asks not to see them, as with the `-E` option). In particular, trailing space should be visible.
- Line lengths are limited to a maximum of 78 characters, so as to fit comfortably on most terminal displays.
- The output makes clear distinction between line breaks that occur due to line limits or intermixed character types, and those that actually occurred in the input.

See Chapter 1 [Overview], page 1, for an output example.

3.1 Text Lines

Lines of plain text input, are output between a pair of pipe characters '`|`', with a final period to indicate the linefeed character. Input:

```
Hello there
```

Output:

```
|Hello there|.
```

Trailing whitespace is thus made plainly visible to the reader.

```
|Trailing space:    |.
```

A blank line is represented as empty content between the pipes.

```
||.
```

If the input line is too long to display as one line of output, it will be displayed as follows, with dash characters, '`-`', to mark continuity at the beginnings and ends of the lines (outside the pipe characters).

```
|This input line |-
-|was too long to|-
-| fit|.
```

You might wonder whether pipe characters themselves might have to be escaped, so as to avoid confusion with the surrounding pipes. After all, in a C string literal, denoted by double-quotes, '`"`', one must escape literal double-quote characters by preceding them with a backslash (and must then also escape literal backslashes): `"She said \"no\""`. However, no such escaping mechanism is needed in Teseq. Input:

```
A line with a pipe | in it, and a line with just a pipe
|
```

Output:

```
|A line with a pipe | in it, and a line with just a pipe|.
|||.
```

No special treatment whatsoever!

The reason Teseq can get away with this is that the pipe character only has its special "mark the enclosed as normal text" meaning, when it is the first and the last character on a line of output (aside from a possible prefacing dash to indicate a wrapped line, or a trailing dash or period).

A lot of the output lines you've seen so far have all ended with a period, denoting the linefeed character. However, this isn't always the case. For instance, if the input line contains a control character, Teseq will close the text line (with just a pipe, no period), print a control-character line with the control characters, and then finish up the rest of the text line. The following sample output represents a single line which contains a delete character and a null character:

```
|Fee fi|
. DEL/^?
|fo|
. NUL/^@
|fum|.
```

And here's a line that's terminated with a carriage-return/line-feed combination, rather than just a linefeed.

```
|The promised line|
. CR/^M LF/^J
```

The linefeed character is special: when it's preceded by control characters or escape sequence, it's printed as a control character; when it's preceded by text characters (or by nothing at all), it's printed as that final dot thing. This behavior is designed to make a newline look like the end of a line or a blank line when it should, and to look like a control character when it should.

And of course, if the final line in the file is missing a newline, the dot won't appear there either.

Currently, the only characters included in text lines, are those from the printable range of characters from US ASCII (Ecma-6 / ISO 646)—and sometimes linefeed, represented by the special final-dot notation. This means that if the terminal was using a character set that high-value code points, all high-valued bytes will be represented in hexadecimal on control-character lines, and not displayed on a text line. Future versions of Teseq may provide options allowing for these characters to be represented properly in text lines, but for now, the output format is ASCII characters only. See Chapter 7 [Future Enhancements], page 18.

Note that even characters falling in the normal range of ASCII printable characters may not necessarily be represented correctly: for instance, if escape codes are present in the input that would switch the terminal to a different national variant of Ecma-6 / ISO 646, then the real terminal might display (say, for ISO-646ES) '`Ñ`' instead of '`[`'; but the output from Teseq will not change based on this (even though it will recognize and identify the control sequences that invoke that character set for use).

3.2 Control-Character Lines

Control-character lines are used to display characters whose code values fall outside the range of printable characters from US ASCII (Ecma-6 / ISO 646). That is, those characters

whose code values fall below below 32 decimal (those from the C0 set of control characters from Ecma-48 / ISO/IEC 6429); and those whose values are at or above 127 decimal (the "delete" character, and character byte values with the high bit set).

Control-character lines begin with an initial dot, '`.`', followed by the control characters or high-value bytes being represented.

```
. BEL/^G NUL/^ CR/^M LF/^J DEL/^? xA0 xFF
```

Control characters (whose numeric codes fall below decimal 32, plus the delete character at decimal 127) are represented by a mnemonic acronym identifying the character's function. Unless the `-C` was given, this acronym is followed by a slash, and the control-key combination that would produce the corresponding character (control characters are usually much more recognizable from one or the other of their name or their control-character representation, than they are by their hexadecimal code value). The "control-key combination" representation consists of a "hat" or "circumflex accent" character, followed by a character with a value in the range of 63 through 95 decimal.

Note that the delete character, designated as '`DEL/^?`', is a special case, in that one can not generally reproduce that key by holding down the control key and typing a question mark; it is simply used as an identification of the key.

Other values (high-value bytes) are represented by the lowercase letter '`x`' followed by the two-digit hexadecimal code value for the character.

For reference, here's a table of the control characters (plus DEL). It is based on the information from Table 1 of Ecma-48 / ISO/IEC 6429 (the control-key representation has been added).

Hex	**Key**	**Name**	**Hex**	**Key**	**Name**
x00	^@	NUL	x10	^P	DLE
x01	^A	SOH	x11	^Q	DC1
x02	^B	STX	x12	^R	DC2
x03	^C	ETX	x13	^S	DC3
x04	^D	EOT	x14	^T	DC4
x05	^E	ENQ	x15	^U	NAK
x06	^F	ACK	x16	^V	SYN
x07	^G	BEL	x17	^W	ETB
x08	^H	BS	x18	^X	CAN
x09	^I	TAB	x19	^Y	EM
x0A	^J	LF	x1A	^Z	SUB
x0B	^K	VT	x1B	^[	ESC
x0C	^L	FF	x1C	^\	IS4
x0D	^M	CR	x1D	^]	IS3
x0E	^N	SO	x1E	^^	IS2
x0F	^O	SI	x1F	^_	IS1
x7F	^?	DEL			

3.3 Escape-Sequence Lines

Escape-sequence lines, which begin with the colon, '`:`', don't add any new semantics—any characters in an escape-sequence line could be represented on control-character and text

lines (and, with just a one-character change in the escape sequence, would be). But they serve to set escape sequences apart from normal control-character or text lines, making it easier to see on one line all the characters that contribute to a single control function, rather than splitting them between control-character and text lines. Here's an example with some intermixed escape-sequence and text lines (without the usual label and description lines, which are described in later sections):

```
|Well |
: Esc [ 3 ; 31 m
|Daniel|
: Esc [ 23 ; 39 m
| didn't do it...|.
```

The two escape-sequence lines represent, respectively, controls that set text rendering to use italics in the color red, and to set normal font rendering in the default color (the actual interpretation of these controls may vary by application).

Note that the escape-sequence lines include control characters (well, character, namely '`Esc`') intermixed with normal text characters. So the above could have been written like:

```
|Well |
. ESC
|[3;31mDaniel|
. ESC
|[23;39m didn't do it...|.
```

But this loses the separation between characters that, yes, happen to be text characters, but really just contribute to some terminal control function invocation, and characters that are, really and truly, text.

```
: Esc [ 3 ; 31 m
```

Some things to note. First, the escape key is noted as '`Esc`', and not '`ESC`' as it would be in a control-character line. Don't ask me why; maybe I just felt it was one more thing to dilineate between escape-sequence lines and control-character lines.

Also, each character is separated from its neighbors by a single space, except that strings of digits are lumped all together.

Each character is represented by itself (including colon, as long as it's not at the start of the line), except the escape character, and the space character (represented as '`Spc`'). Control characters and high-value bytes are not currently represented on escape-sequence lines (they are not part of any escape sequences Teseq recognizes), but if they are in the future (say, as part of non-standardized escape sequences), they will most likely be represented as '`xXX`' hexadecimal strings (as high-value bytes are represented in control-character lines.

If an escape sequence requires more than one output line, the continuing lines will also begin with a colon, followed by two spaces (instead of one). Lines will not be split in the middle of a number.

```
: Esc [ 1 ; 2 ; 3 ; 4
:   ; 5 ; 6 ; 7 m
```

Recognizing Escape Sequences

Okay, great, so escape-sequence lines help distinguish control-characters and text characters that make up an escape sequence from those that don't. But what exactly makes up an escape sequence, anyway?

The Ecma-35 / ISO/IEC 2022 standard defines an escape sequence to be a sequence of characters beginning with ESC, with a final byte in the range `x30`–`x7E`, and any number (including zero) of intermediate bytes in the range `x20-x2F`. Table 3.1 has been provided as a reference for finding which characters match which codes.

	x2X	**x3X**	**x4X**	**x5X**	**x6X**	**x7X**
xX0	SPC	0	@	P	‘	p
xX1	!	1	A	Q	a	q
xX2	"	2	B	R	b	r
xX3	#	3	C	S	c	s
xX4	$	4	D	T	d	t
xX5	%	5	E	U	e	u
xX6	&	6	F	V	f	v
xX7	’	7	G	W	g	w
xX8	(	8	H	X	h	x
xX9	)	9	I	Y	i	y
xXA	*	:	J	Z	j	z
xXB	+	;	K	[	k	{
xXC	,	<	L	\	l	|
xXD	-	=	M	]	m	}
xXE	.	>	N	^	n	~
xXF	/	?	O	_	o	DEL

Table 3.1

So, for instance, the following is a valid escape sequence.

```
: Esc $ ( C
```

‘`$`’ and ‘`(`’ have code values `x24` and `x28`, and so are valid intermediate bytes; ‘`C`’ has the value `x43`, and so terminates the escape sequence.

You may have noticed that a lot of the examples of escape sequences in this document don't actually follow this format. For instance,

```
: Esc [ 3 ; 31 m
```

According to the definition we just gave, ‘`[`’ should be the final byte of an escape sequence. So why does Teseq keep going until it reaches the ‘`m`’?

The answer is that the escape sequence *does* end with the ‘`[`’; but the combination ‘`Esc [`’ invokes a control named `CONTROL SEQUENCE INTRODUCER` (CSI). The CSI control marks the beginning of a different kind of sequence, called a “control sequence”. Control sequences are described by the Ecma-48 / ISO/IEC 6429 standard, which considers it to be a distinct concept from escape sequences; however, Teseq treats both types of sequences as “escape sequences”.

A control sequence starts with the two-character CSI escape sequence '`Esc [`', followed by an optional sequence of parameter bytes in the range `x30`–`x3F`, an optional sequence of intermediate bytes in the range `x20`–`x2F` (the same range as intermediate bytes in a regular escape sequence), and a final byte in the range `x40`–`x7e`. The set of standard control sequence functions are defined in Ecma-48 / ISO/IEC 6429.

When used in accordance with the standard, the parameter bytes are used to provide a semicolon-separated list of numeric parameters to the control function being invoked. These affect the details of the control function's behavior; but not which control function is being invoked:

```
: Esc [ 1 m
& SGR: SELECT GRAPHIC RENDITION
" Set bold text.
: Esc [ 0 m
& SGR: SELECT GRAPHIC RENDITION
" Clear graphic rendition to defaults.
```

Both sequences end with the same sequence of intermediate bytes (none) and final byte; both invoke the SGR control function. But the first one indicates that following text should be rendered boldface, while the second indicates that text rendering should be restored to its default settings.

Intermediate bytes, however, together with the final byte, *do* affect the meaning of the function invoked. Currently, Ecma-48 / ISO/IEC 6429 only defines functions for either no intermediate bytes, or a single space character (`x20`) as the intermediate byte.

```
: Esc [ A
& CUU: CURSOR UP
: Esc [ Spc A
& SR: SCROLL RIGHT
```

Ecma-48 / ISO/IEC 6429 describes an alternate representation for CSI; the 8-bit byte value `x9B`. Teseq does not currently treat that value specially, nor any of the other high-value bytes from the C1 set of control functions. This is because whether or not those bytes indicate control functions is dependent upon what character encoding is in use. Future versions of Teseq may support an option to interpret these forms as well, at which time control sequences using the single-byte CSI control will probably be rendered like:

```
: CSI [ 1 m
```

Ecma-48 / ISO/IEC 6429 also describes another kind of sequence called "control strings". These are not interpreted by Teseq; the control characters involved (for example, '`SOS/^X`' and '`ST/^\`') will be printed on control-character lines, and any text characters will be displayed on text lines.

Future versions of Teseq will probably not depart from this display behavior; however, support for some common interpretations for control strings may be added, in which case a label line and/or description line might follow the control string, describing its usual interpretation.

3.4 Label Lines

Label lines begin with the ampersand, '`&`':

```
& SGR: SELECT GRAPHIC RENDITION
```

Label lines always describe a control function, and are always preceded by the escape sequence that invokes that function (unless the `-E` option was given, suppressing output for escape-sequence lines).

The format of a label line's content is always *acronym*: *name*; both of these are defined for each control function by Ecma-48 / ISO/IEC 6429. In some cases, they may also name private functions defined by popular terminals and terminal emulators.

Label lines currently are never wrapped; however, future versions of Teseq may wrap label lines. At that time, continuation label lines will probably consist of an extra space after the ampersand (similar to how escape-sequence lines are wrapped). Lines will never be split in the middle of a word.

Future versions of Teseq may use label lines to describe things besides escape sequences; for instance, control characters or control strings, or other strings that may be interpreted specially by some devices or applications.

3.5 Description Lines

Description lines begin with the double-quote, '`"`':

```
" Move the cursor up 2 lines.
```

Sequences of description lines are generally preceded by a label line (unless the `-L` option was supplied), and describe the same control function labeled by that line. More than one description line may be used to describe a function, whereas only one label line is ever used for a control function.

It is important to understand that the descriptions provided are only approximations and guesses, and suffer from various limitations. The behavior of many control functions are dependent on terminal state that Teseq does not track. Teseq chooses a common default setting for applicable terminal modes, and issues a description based on that. For instance, the INSERT LINE function is described as follows:

```
: Esc [ 2 L
& IL: INSERT LINE
" Shift lines after the cursor to make room for 2 new lines.
```

However, depending on the current setting of the LINE EDITING MODE, the actual behavior might be to shift the lines *before* the cursor, rather than the ones after. Future versions of Teseq may track enough terminal state to improve the accuracy of these descriptions (see Chapter 7 [Future Enhancements], page 18), but they would still need to guess at the initial state of the terminal, for any modes that had not been explicitly set or reset.

Descriptions are also often inaccurate. For instance, the description for INSERT LINE should really read "shift the current line and the lines after the cursor. . .". In addition, no mention is made of the fact that the extent of the shifted part is dependent on previous invocations of SELECT EDITING EXTENT. A conscious decision has been made to value brevity over accuracy.

Also, the descriptions are based (loosely) on the semantics defined by Ecma-48 / ISO/IEC 6429. There is no guarantee that this corresponds to the semantics defined for the actual terminal on which these functions were invoked. The terminal may have different behavior, or may not even accept the function. The descriptions are intended as

a rough aid in remembering what a given function does; to really understand the actual semantics of a function, you should read the terminal device's documentation, and/or Ecma-48 / ISO/IEC 6429.

Future versions of Teseq may use description lines to describe things besides escape sequences; for instance, control characters or control strings, or other strings that may be interpreted specially by some devices or applications.

Some description lines may appear without a preceding label line (even when `-L` was not specified), in the event that no standard designation for the function is known.

3.6 Delay Lines and Halt Lines

A delay line begins with an "at" sign '`@`', and contains a single numeric value; a number of seconds to pause before continuing on to process further lines. The `reseq` program will obey these instructions only if it is given the `--replay` option.

```
@ 3.0051
```

Delay lines are only issued by `teseq` when it has been given the `-t` option, which uses a timing file from '`script -t`' to determine where to insert delays. Aside from that, delay lines can be useful for manual insertion into Teseq output, to introduce a delay at a particular point when using '`reseq --replay`', which can aid in giving the user time to more easily observe terminal behavior.

If a line begins with three "at" signs together '`@@@`', and `reseq` is invoked with both '`--replay`' *and* '`--halts`', then `reseq` will pause at this location until the user presses a key to indicate continuation. The remainder of the line will be ignored, so can be used for comments.

Such a line will never be output by `teseq`; it must be inserted by the user. Since `reseq` does not give any indication to the user that it is awaiting input (as opposed to simply waiting on a length delay line), it is highly advisable to insert this "halt" line only after a spot in the recorded terminal script that already provides such an indication.

The halt line can be useful for creating animated demonstrations of program usage, and similar sorts of presentation-oriented scripts.

4 Color Mode

Teseq can provide color formatting to its output. This feature is disabled by default, but can be activated via `--color=auto` or `--color=always`. If '`auto`' is used, then color will be added to the output only if the output is to a terminal; otherwise not. If '`always`' is specified, then color formatting codes will be provided for the output unconditionally. Note that such output may not be fed to `reseq`, which will not be able to handle these codes.

The default colors used are suitable for use against a dark/black background, but may be less suitable for use on terminals with light-colored backgrounds. The colors used may be altered by setting the environment variable '`TESEQ_COLORS`'. It consists of comma-separated key=value pairs, where the key is usually the prefix character of the line to be colored, and the value consists of parameter values for the terminal SGR control. Any prefixes not specified in '`TESEQ_COLORS`' retain their default values.

In the case of text lines, in which literal text is bracketed by pipe characters ('`|`'), color may be specified separately for the inner text, and for the bracketing pipes, including the '`|-`' and '`-|`' around formatting (non-literal) line breaks, and any final '`.`' signifying a literal line break. The inner text is specified by the key '`|>`', while the decorations are specified by the key '`|`'. Currently, no default color settings are made for the decorations; only the inner text.

Here is an example setting for TESEQ_COLORS:

```
TESEQ_COLORS='|>=36;7,|=1,.=31,:=33,&=35,"=32,̄34'
```

This sets the literal text content of text lines to a cyan foreground, and then standout (reverse) mode, so that the result is a cyan background with a foreground of whatever the usual background would be (i.e., if the background is normally black, then the foreground will be black). The color of the text line decorations is not changed, but the decorations are displayed in bold. Control lines ('`.`') are set to red, escape-sequence lines ('`:`') are set to amber, label lines ('`&`') to violet, descriptions ('`"`') to green, and delay lines ('`@`') to blue.

Note that halt lines (lines consisting of exactly '`@@@`'), though recognized by reseq, are never emitted by teseq, and thus have no means of specification in '`TESEQ_COLORS`'.

5 The Reseq Command

Synopsis:

```
reseq [-t|--timings=timings] input output
reseq --replay [-d divisor] input [output]
reseq -h | --help
reseq -V | --version
```

The *input* and *output* arguments are mandatory, but may be specified as '`-`' for standard input or output. Reseq doesn't let output default to standard output because, since it generates raw terminal codes, it is uncommon (and potentially unsafe) to send this directly to the terminal. The exception is when the `--replay` argument has been specified, which is only useful when output is going to the terminal; in that event, the *output* argument is optional.

`-h`
`--help` Print usage information on standard output and exit successfully.

`-V`
`--version`
Print the version number and licensing information of `hello` on standard output and then exit successfully.

`--replay` Honor delay lines in the input, pausing the specified amount of time before continuing to process the next line. This is useful for producing behavior equivalent to that of the `scriptreplay` command (from util-linux (`http://userweb.kernel.org/~kzak/util-linux-ng/`)), but using a Teseq output file as input, rather than a raw typescript file.

`--halts` Only takes effect if '`--replay`' is also specified. In addition to honoring delay lines, also honors user-inserted "halt" lines (those beginning with '`@@@`') in the input. These lines cause `reseq` to halt processing until the user presses a key.

This halting action is not accompanied by any sort of indication that `reseq` is awaiting user action; it is up to the user to ensure that the input script to `reseq` has included some sort of appropriate indication.

When '`--halts`' is in effect, `reseq` will turn off terminal echoing (so the user's keypress to continue the script is not shown), and switches the terminal over to unbuffered I/O, so that keypresses can be read as soon as they are typed.

`-t` *timings*
`--timings` *timings*
Produce timing information from delay lines, in the format generated by `script -t`. This can be used to regenerate `script` typescript and timing files that were fed as the input to `teseq -t timings`. Note that the result will differ slightly from the output from `script -t`, in that the first delay will be zeroed out (`teseq` always throws out the first delay value, whose value from script is an arbitrary value between 0 and 1), and the last delay line will include all the remaining characters (`script`'s timings don't count the final timestamp line).

The `reseq` command essentially does the reverse of `teseq`. If you feed it the output from `teseq`, it will generate the corresponding escape sequences—that is, it will generate the same content that was fed to `teseq` to produce that output. The shell command

```
$ teseq foo | reseq - -
```

is roughly equivalent to

```
$ cat foo
```

The `reseq` command is written in Perl, unlike `teseq` which is compiled from C-language sources, and so requires a Perl interpreter to be present in order to function.

Of the various types of lines output by the `teseq` command, `reseq` only understands four; text lines:

```
|Hello, there|.
|Here are|-
-|some wrapped|-
-|lines|.
```

control-character lines:

```
. CR/^M LF/^J
. CR LF
```

escape-sequence lines:

```
: Esc [ 31 ; 3 m
```

And, of course, delay lines:

```
@ 3.14159
```

Reserved Line Prefixes

It's important for `reseq` to be able to process its input correctly, even if that output came from a different version of `teseq` than `reseq` is familiar with. So, it's important that `reseq` should refuse to continue processing input if it encounters a line that it doesn't recognize, but which might contain important semantic information that effects the output `reseq` should produce.

At the same time, it'd be a shame for `reseq` to refuse to process a line it doesn't understand, if that line contains non-critical information. For example, consider delay lines (beginning with '`@`'). The delay line holds semantic information, to be sure; but not information that would affect `reseq`'s normal operation (it only has meaning when one of the `--timings` or `--replay` options has been specified). So, if an older version of `reseq` had existed that did not recognize them, it would have been a shame if its introduction in the newer release had caused the older version to refuse to process it.

To address both of these concerns, `reseq` has taken the approach of reserving certain prefixes for use as "semantically significant" line prefixes, and others for use in lines that `reseq` can safely ignore. A line that begins with any of the following characters, will cause `reseq` to halt processing and exit with an error.

```
!$+/=[\^{~
```

The idea is that these prefixes are reserved for future use in lines that `reseq` must understand in order to produce correct results.

This leaves all remaining characters free for use in specifying future `teseq` output lines that do not affect processing for older `reseq`s. Note that they are still reserved for `teseq`, and are not intended for users to insert commentary or such. However, `teseq` will never use a line beginning with the space character; and `reseq` will always ignore such lines, so the space character may be used to indicate user comments.

6 Standards

The most authoritative source of information on control functions, their representations in bytes, and their intended meaning, is the Ecma-48 / ISO/IEC 6429 standard. Also related is the Ecma-35 / ISO/IEC 2022 standard, which describes control functions for switching the character encoding, and defines the possible forms of escape sequence (note that, in Teseq, "escape sequence" refers to both the Ecma-48 concept of "control sequences", and the official Ecma-35 meaning of "escape sequence". These standards are available, "free of charge and copyright", at:

`http://www.ecma-international.org/.`

Teseq also recognizes some additional sequences that were not defined by international standard, but are in fairly wide use. In such cases, if descriptions are enabled, the description text for these sequences will be prefixed by a tag in parentheses that identifies where this functionality is found. For instance, `(DEC)` for sequences used in DEC equipment such as the VT100, or `(Xterm)` for squences supported by xterm. Sources of information that were used for identifying and describing these sequences, include:

`http://invisible-island.net/xterm/ctlseqs/ctlseqs.html`
`http://www.vt100.net/.`

7 Future Enhancements

Here are some potential additions that may appear in future versions of Teseq.

7.1 Localized Messages

It is expected that a near release of Teseq will include support for native-language translations of control label names and descriptions. Note that the acronym portions of label lines will never be translated: they represent official designations for their respective controls.

An important requirement of such translations is that they must not interfere with processing of the document; if they use a shift-based encoding, they will shift to the translation encoding only after the line prefix, which must be a plain ASCII character, and shift back to normal ASCII prior to the end of the line. Bytes with values corresponding to the ASCII control characters NUL, CR, or LF (hexadecimal '`x00`', '`x0D`', and '`0x0A`') will not appear, except when they represent their ASCII functions (in no case will NULs be present in the document).

These requirements rule out the use of wide-character encodings such as UCS-2 or UTF-16, which could not be intermixed with ascii characters and which have character representations that would enclude byte values corresponding to the forbidden characters noted above, but should be compatible with UTF-8, ISO 2022-based encodings such as the ISO 8859 series, ISO 2022-JP, or EUC-JP and other encodings that meet these requirements such as GBK, Big5, or Shift JIS.

7.2 Stateful Processing

Future versions of Teseq may support options to remember state information about the terminal. For instance if Teseq has seen the invocation of SELECT GRAPHIC RENDITION to set underlined text, Teseq might render any further text that appears in with underlining. In addition, the meaning of some standardized commands defined by Ecma-48 / ISO/IEC 6429 depend on the current setting of various terminal modes, and Teseq currently makes assumptions about those modes. A state-remembering Teseq might remember the last time a mode was set, and no longer need to make assumptions.

If Teseq is given information about the size of the target terminal, it could also provide information about the cursor's present location after every cursor-moving command.

Currently, all output from `teseq` is in US ASCII; but future versions of Teseq might support output in other encodings. If that happens, Teseq might also add support for the handling of Ecma-35 / ISO/IEC 2022 character code-switching sequences, such that the characters given in text lines would reflect the actual characters as they would actually appear in the terminal device, depending on the current encoding state.

Note that features which alter the bytes found between the pipe characters of a text line, would most likely break reverse-translation with `reseq`, as it would be much harder to tell what the original byte values had been.

7.3 Terminal Database Awareness

Future versions of Teseq may allow the user to specify the name of the terminal for which the input was intended, which Teseq will then use to detect when a feature from that terminal's entry in the terminal database (for example, the terminfo database) has been invoked.

Since features from terminal database entries often involve multiple Ecma-48 / ISO/IEC 6429 controls, Teseq would probably need to indicate out-of-band "start" and "end" markers for the feature. For instance, if the definition for the '`clear`' feature for the specified terminal is '`\E[H\E[J`', then an input string of '`\E[m\E[H\E[J\E(B`' might result in output like:

```
: Esc [ m
# start: clear
: Esc [ H
: Esc [ J
# end: clear
: Esc ( B
```

(Label and description lines have been removed from this example.)

No commitment has been made to any particular output format for this feature; the above is intended purely as an example of one possible approach.

8 Contact and Information

If you have any questions, comments, suggestions, or bug reports, please take advantage of the mailing list at `bug-teseq@gnu.org`. To subscribe, send an empty email with the Subject "subscribe" to `bug-teseq-request@gnu.org`; or use the web interface at `http://lists.gnu.org/mailman/listinfo/bug-teseq`. You can also contact the author/maintainer directly at `micah@addictivecode.org`.

The official website for GNU Teseq is at `http://www.gnu.org/software/teseq/`. For the latest updates and other information, please check that site (or better yet, stay appraised by subscribing to the mailing list.

GNU Teseq uses Savannah to manage bug-tracking, and to host the development source repository in Mercurial: `https://savannah.gnu.org/projects/teseq/`.

Appendix A Copying

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and

that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called

an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

"

" 11

&

& 10

-

-" 4
-& 4
--buffered 3, 4
--color 3, 13
--colour 3, 13
--halts (reseq) 12, 14
--help 3
--help (reseq) 14
--no-interactive 3, 4
--replay (reseq) 12, 14, 15
--timings 4
--timings (reseq) 14, 15
--version 3
--version (reseq) 14
-: 4
-^ 4
-b 3, 4
-C 3, 7
-D 3
-E 3, 11
-h 3
-h (reseq) 14
-I 3, 4
-L 3, 11
-t 4
-t (reseq) 14, 15
-V 3
-V (reseq) 14
-x 4

.

. 7

@

@ 12

|

| 5

A

ASCII 6, 18

C

color 13
colour 13
control character 1, 3, 6
control characters table 7
control sequence 1, 9, 17
control string 10
control-character line 4, 6, 15
CSI 9

D

delay line 4, 12, 15
delete character 7
description line 3, 11

E

Ecma-35 ..... 9, 17, 18
Ecma-48 ..... 3, 7, 9, 10, 11, 12, 17, 18, 19
Ecma-6 6
escape sequence 17
escape sequence, defined 9
escape sequence, final byte 9
escape sequence, intermediate byte 9
escape-sequence line 3, 15
escape-sequence lines 7
example 2
examples 1

F

final byte 9, 10

H

halt line 12
Hello, world! 1
help 3

I

intermediate byte 9, 10
invoking 3
ISO 646 6
ISO/IEC 2022 ..... 9, 17, 18
ISO/IEC 6429 ..... 3, 7, 9, 10, 11, 12, 17, 18, 19

L

label line 3, 10
line prefix 2, 4, 5, 15
line prefix, reserved 15

N

non-canonical mode 4

O

options 3

P

parameter byte 10
Perl 15
prefix, line 2, 4, 5, 15

R

`reseq` 2, 3, 12, 14, 18
reserved line prefixes 15
Reversing the output of `teseq` 14

S

`script` 1, 4, 12, 14
`scriptreplay` 14
standards 3, 6, 7, 9, 10, 11, 12, 17, 18, 19
strip 2
stripping 2

T

table of control characters 7
table of printable ASCII characters 9
terminal 1
terminal emulator 1
terminfo 19
`teseq`, interactive mode 4
`teseq`, reversing the output 14
TESEQ_COLORS 13
text line 4, 15, 18
typescript 1, 2

U

usage 3