

Vesti Transpiler User Manual

Sungbae Jeong

November 6, 2025

Contents

1	Introduction	1
2	Structure of Vesti File	2
3	Keywords	2
3.1	<code>docclass</code> keyword	2
3.2	<code>importpkg</code> keywords	3
3.3	<code>startdoc</code> keyword	3
3.4	<code>useenv</code> keyword	3
3.5	<code>beginenv</code> keyword and <code>endenv</code> keyword	4
4	Builtins	4
4.1	<code>#label</code> builtin	5
4.2	<code>#eq</code> builtin	5
4.3	<code>#mathchardef</code> builtin	5
5	Source Code of This Document	5

1 Introduction

I used to create several documents using \LaTeX (or plain \TeX but \TeX is quite cumbersome to write—especially when working with very complex tables or inserting images). Its markdown-like syntax is also not comfortable to use. For example, here is a simple \LaTeX document:

```
1 % coprime is my custom class. See https://github.com/e0328eric/coprime.
2 \documentclass[tikz, geometry]{coprime}
3
4 \settitle{My First Document}{Sungbae Jeong}{}
5 \setgeometry{a4paper, margin = 2.5cm}
6
7 \begin{document}
8 \section{Foo}
9 Hello, World!
10 \begin{figure}[ht]
11     \centering
12     \begin{tikzpicture}
13         \draw (0,0) -- (1,1);
14     \end{tikzpicture}
15 \end{figure}
16
17 The code above is a figure using TikZ.
18
19 \end{document}
```

What annoys me most when using it is the ‘\begin’ and ‘\end’ blocks. Is there a way to write something much simpler? This question led me to start this project. Currently, the following code is generated into the \LaTeX code above (except comments) using vesti:

```

1 % coprime is my custom class. See https://github.com/e0328eric/coprime.
2 \docclass coprime (tikz, geometry)
3
4 \settitle{My First Document}{Sungbae Jeong}{}
5 \setgeometry{a4paper, margin = 2.5cm}
6
7 \startdoc
8
9 \section{Foo}
10 Hello, World!
11 \useenv figure [ht] {
12     \centering
13     \useenv tikzpicture {
14         \draw (0,0) -- (1,1);
15     }
16 }
17
18 The code above is a figure using TikZ.

```

2 Structure of Vesti File

Vesti is similar as \LaTeX . Its structure consists with two parts: preamble and main. Preamble is the place where \LaTeX documentclass, packages, and several settings are located. Main body is where actual documentation is located. Below figure is the simple Vesti documentation.

```

1 \docclass article (10pt)
2 \importpkg {
3     geometry (a4paper, margin=2.2cm)
4 }
5 \startdoc
6 Hello, Vesti!

```

We will see later, but the very difference with \LaTeX is that Vesti has its own keywords (keywords are colored with purple). It makes the code readable and it is easier and faster to write the document. The keyword `\startdoc` splits the preamble and the main part of the documentation similar with

`\begin{document}` in \LaTeX . However, Vesti does not have the analogous part of `\end{document}`, because almost every \LaTeX document (99.999% I’m sure) does not have any code below `\end{document}`. For this reason, Vesti automatically ends document when EOF (End Of File) is found.

3 Keywords

Followings are reserved as keywords. In this document, every Vesti keyword has the form like **this**.

<code>\begenv</code>	<code>\compty</code>	<code>\cpfile</code>	<code>\defenv</code>
<code>\defun</code>	<code>\docclass</code>	<code>\endenv</code>	<code>\importmod</code>
<code>\importpkg</code>	<code>\importves</code>	<code>\startdoc</code>	<code>\useenv</code>

Table 1: Keywords in Vesti

3.1 `\docclass` keyword

Keyword `\docclass` is an analogous of `\documentclass` in \LaTeX . If `\docclass` is in the main paragraph, it acts just a normal word. In other words, `\docclass` actives only in the preamble. The syntax of `\docclass` is following:

`docclass` <class name> (<arguments>)

Here, arguments are separated by commas and embraced by (). Here are some examples.

- `docclass` article
- `docclass` article (10pt)
- `docclass` article (10pt, twocols)
- `docclass` article (10pt,twocols)

3.2 `importpkg` keywords

Keyword `importpkg` is an analogous of `\usepackage` in \LaTeX . If `importpkg` is in the main paragraph, it acts just a normal word. In other words, `importpkg` activates only in the preamble.

`importpkg` has two different syntax. First one is same as `docclass`.

`importpkg` pkg-name (arguments)

Here, arguments are separated by commas and embraced by (). In the practical case, one should include several packages with options. `importpkg` also supports such case. We will look at an example instead of giving rigorous grammar.

```
1 importpkg {  
2   amsmath, amssymb, amsthm,  
3   geometry (a4paper, margin=2.2cm),  
4 }
```

As one can see, inside of {}, several packages can be used together with their options.

3.3 `startdoc` keyword

Keyword `startdoc` tells to Vesti that the main document starts. In the main document, you can also write `startdoc` in the main document. In that case, `startdoc` does nothing.

3.4 `useenv` keyword

As the name implies, keyword `useenv` is an analogous of `\begin{...}` and `\end{...}` pair in \LaTeX . The simplest `useenv` is like this.

```
useenv center {  
    Hello, World!    or    useenv center { Hello, World! }  
}
```

As you can see, `useenv` center is the part of `\begin{center}`, and the single } is the part of `\end{center}`. Since Vesti knows their pair, one can write a code with several environment, and each pair is properly matched. For instance, above example is written in Vesti like follows. Here, `\useenv` just prints `useenv` in that style.

```
1 useenv figure [ht] {  
2   \centering  
3   useenv tikzpicture {  
4     useenv scope {  
5       \path (0,0) node {\vbox{  
6         %#\hbox{\tt\code{useenv} center \{}  
7         %#\hbox{\tt\obeyspaces Hello, World!}  
8         %#\hbox{\tt\obeyspaces\}}  
9       }  
10    }  
11    \path (2.3,0) node {or};  
12    useenv scope [shift={(6,0)}] {
```

```

13         \path (0,0) node {\tt\useenv center \{ Hello, World! \}};
14     }
15 }
16 }

```

Full syntax about `useenv` is the following.

`useenv` <environment name> <argument>* { <body> }

where “*” means that the number of <argument> is zero or at least one, and

$$\text{<argument>} = \begin{cases} (\text{<argument>}) & \text{mandatory arguments} \\ [\text{<argument>}] & \text{optional arguments} \end{cases}$$

For instance, below one is a valid Vesti code (environment `foo` is undefined in general). As one can see, spaces can exist in between <argument>s.

```

1 useenv foo (asd)(fff)[\ames and \awdsa] (askws) [\rrsaa] {
2     foobar
3 }

```

3.5 `beginv` keyword and `endenv` keyword

As the name implies, both keywords `beginv` and `endenv` are analogous of `\begin{...}` and `\end{...}` pair in \LaTeX , respectively. Thus below code

```

1 beginv center
2     asdsad
3 endenv center

```

is exactly same as

```

1 useenv center {
2     asdsad
3 }

```

Then why we need `beginv` and `endenv` if we already have `useenv`?

4 Builtins

Vesti also has its own builtin functions, which are prefixed with `#`. One might wonder what distinguishes builtins from keywords. In fact, from the compiler’s internal perspective, there is no real difference. However, in actual language usage, constantly typing the prefix can be somewhat tedious, especially for functions that are commonly used.

From the perspective of language design—particularly in Vesti—it is sometimes desirable to use names that cannot serve as keywords. For example, Vesti provides a built-in function `#label`, which will be explained later. Since Vesti is a typewriting-oriented language, the word “label” is often used in its ordinary sense rather than in its special semantic meaning within the language.

Followings are reserved as builtin functions.

<code>#chardef</code>	<code>#enum</code>	<code>#eq</code>	<code>#get_filepath</code>	<code>#label</code>
<code>#ltx3_off</code>	<code>#ltx3_on</code>	<code>#makeatletter</code>	<code>#makeatother</code>	<code>#mathchardef</code>
<code>#mathmode</code>	<code>#noltx3</code>	<code>#picture</code>	<code>#showfont</code>	<code>#textmode</code>
<code>#xparse</code>				

Table 2: Builtins in Vesti

Since `fancyvrb` does not allow verbatim-like commands inside of `Verbatim` environment, builtins are not colored in example codes.

4.1 #label builtin

4.2 #eq builtin

#eq is a abbreviation of 'useenv equation'. Actually, below codes are same.

```
1 #label(eq:1)
2 useenv equation {
3     \sum_{n=1}^{\infty} {1//n^2} = {\pi^2//6}.
4 }
```

```
1 #eq(eq:1) {
2     \sum_{n=1}^{\infty} {1//n^2} = {\pi^2//6}.
3 }
```

Since Vesti is used in the mathematical documents mainly, small syntactic suger is needed, so Vesti introduce #eq builtin.

4.3 #mathchardef builtin

#mathchardef defines a math token using the unicode codepoint. Here is the grammar of #mathchardef builtin.

`#mathchardef <kind> <font-num> <unicode-codepoint> <function>`

Before explaining each parameter, introduce some examples.

- `#mathchardef_0_29d8_\lfoo`
 - `#mathchardef_0_2202_\diff`
- ```
select font "[STIXTwoMath-Regular.ttf]/OT:script=math;language=dflt;"
```

# 5 Source Code of This Document

Below code was generated by inline lua.

```
1 docclass article (10pt)
2 importpkg {
3 geometry (a4paper, margin = 2.2cm),
4 xcolor,
5 tikz,
6 fancyvrb,
7 }
8
9 \title{Vesti Transpiler User Manual}
10 \author{Sungbae Jeong}
11
12 importves (font.ves)
13
14 % read file contents using lua
15 #lu:
16 local function read_all(path)
17 local f, err = io.open(path, "rb")
18 assert(f, ("cannot open %s: %s"):format(path, err))
19 local data = f:read("*a")
20 f:close()
21 return data
22 end
23 :lu#<readAll>
24
```

```

25 % definition of \keyword command
26 #xparse defun [!] keyword (m) {\tt\color{purple}#1}}
27 #xparse defun [!] builtin (v) {\tt\color{blue!65!yellow}\##1}}
28 #xparse defun useenv (s) {\IfBooleanTF{#1}{\keyword{%-useenv-}}{\keyword{%-useenv-}} }
29 #xparse defun begenv (s) {\IfBooleanTF{#1}{\keyword{%-begenv-}}{\keyword{%-begenv-}} }
30 #xparse defun endenv (s) {\IfBooleanTF{#1}{\keyword{%-endenv-}}{\keyword{%-endenv-}} }
31
32 % space character used in the texbook.
33 #makeatletter
34 #chardef 2423 \@b
35 defun ob {\tt\@b}
36 #makeatother
37
38 startdoc
39 \maketitle
40 \tableofcontents
41
42 \section{Introduction}
43 I used to create several documents using \LaTeX\ (or plain \TeX\ but \TeX\ is
44 quite cumbersome to write-especially when working with very complex tables or
45 inserting images). Its markdown-like syntax is also not comfortable to use. For
46 example, here is a simple \LaTeX\ document:
47
48 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
49 %%% coprime is my custom class. See https://github.com/e0328eric/coprime.
50 %\documentclass[tikz, geometry]{coprime}
51 %#
52 %\settitle{My First Document}{Sungbae Jeong}{ }
53 %\setgeometry{a4paper, margin = 2.5cm}
54 %#
55 %\begin{document}
56 %\section{Foo}
57 %Hello, World!
58 %\begin{figure}[ht]
59 %# \centering
60 %# \begin{tikzpicture}
61 %# \draw (0,0) -- (1,1);
62 %# \end{tikzpicture}
63 %\end{figure}
64 %#
65 %#The code above is a figure using TikZ.
66 %#
67 %\end{document}
68 }
69
70 What annoys me most when using it is the \lq\verb@\begin@\rq\ and
71 \lq\verb@\end@\rq\ blocks. Is there a way to write something much simpler? This
72 question led me to start this project. Currently, the following code is
73 generated into the \LaTeX\ code above (except comments) using vesti:
74
75 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
76 %%% coprime is my custom class. See https://github.com/e0328eric/coprime.
77 %#+keyword|docclass@ coprime (tikz, geometry)
78 %#
79 %\settitle{My First Document}{Sungbae Jeong}{ }
80 %\setgeometry{a4paper, margin = 2.5cm}
81 %#
82 %#+keyword|startdoc@
83 %#
84 %\section{Foo}

```

```

85 %%Hello, World!
86 %#+keyword|useenv@ figure [ht] {
87 %% \centering
88 %#+keyword|useenv@ tikzpicture {
89 %% \draw (0,0) -- (1,1);
90 %% }
91 %#+}
92 %%
93 %The code above is a figure using TikZ.
94 }
95
96 \section{Structure of Vesti File}
97 Vesti is similar as \LaTeX. Its structure consists with two parts: {\tt preamble} and
98 {\tt main}. Preamble is the place where \LaTeX\ documentclass, packages, and
99 several settings are located. Main body is where actual documentation is located.
100 Below figure is the simple Vesti documentation.
101
102 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
103 %#+keyword|docclass@ article (10pt)
104 %#+keyword|importpkg@ {
105 %% geometry (a4paper, margin=2.2cm)
106 %#+}
107 %#+keyword|startdoc@
108 %Hello, Vesti!
109 }
110
111 We will see later, but the very difference with \LaTeX\ is that Vesti has its
112 own keywords (keywords are colored with purple). It makes the code readable and
113 it is easier and faster to write the document. The keyword startdoc splits
114 the preamble and the main part of the documentation similar with
115
116 % Don't ask why I chose Q for catcode 0.
117 %#{\tt\catcode`Q=0 Qcatcode`\\=12 \beginQ{documentQ}} in \LaTeX.
118 However, Vesti does not have the analogous part of
119 %#{\tt\catcode`Q=0 Qcatcode`\\=12 \endQ{documentQ}},
120 because almost every \LaTeX\ document (99.999% I'm sure) does not have any code
121 below %#{\tt\catcode`Q=0 Qcatcode`\\=12 \endQ{documentQ}}.
122 For this reason, Vesti automatically ends document when EOF (End Of File) is
123 found.
124
125 \section{Keywords}
126 Followings are reserved as keywords. In this document, every Vesti keyword has
127 the form like \keyword{this}.
128 useenv table [ht] {
129 \centering
130 #lu:
131 local content = read_all("../src/lexer/Token.zig")
132
133 -- Lua's built-in patterns don't support lookahead.
134 -- We capture both the keyword and the TokenType, then filter out 'deprecated'.
135 -- Pattern breakdown:
136 -- %.%{ => matches ".{"
137 -- %s*"([~"]+)" => a quoted string -> capture 1
138 -- %s*,%s*TokenType%.([%w_]+) => TokenType.<Name> -> capture 2
139 local pat = "%.%{%s*"([~"]+)"%s*,%s*TokenType%.([%w_]+)"
140
141 local keywords = {}
142 for name, tok in content:gmatch(pat) do
143 if tok ~= "deprecated" then
144 keywords[#keywords + 1] = name

```

```

145 end
146 end
147
148 table.sort(keywords)
149
150 vesti.print([[\\begin{tabular}{cccc}]]
151
152 for i, kw in ipairs(keywords) do
153 local cell = string.format("\\keyword{%s}", kw)
154 if (i % 4) == 0 then
155 vesti.print(cell .. [[\\]])
156 else
157 vesti.print(cell .. "&")
158 end
159 end
160
161 vesti.print([[\\end{tabular}]]
162 :lu#[readAll]
163 \\caption{Keywords in Vesti}
164 }
165
166 \\subsection{\\keyword{docclass} keyword}
167 Keyword \\keyword{docclass} is an analogous of \\verb|\\documentclass| in \\LaTeX.
168 If \\keyword{docclass} is in the main paragraph, it acts just a normal word.
169 In other words, \\keyword{docclass} actives only in the preamble.
170 The syntax of \\keyword{docclass} is following:
171
172 useenv center {
173 \\keyword{docclass}\\kern0.5em <class name>\\kern0.5em {\\tt(<arguments>{\\tt})}
174 }
175 Here, arguments are separated by commas and embraced by {\\tt ()}. Here are some
176 examples.
177
178 \\goodbreak
179 useenv itemize {
180 \\item \\keyword{docclass} {\\tt article}
181 \\item \\keyword{docclass} {\\tt article (10pt)}
182 \\item \\keyword{docclass} {\\tt article (10pt, twocols)}
183 \\item \\keyword{docclass} {\\tt article (10pt,twocols)}
184 }
185
186 \\subsection{\\keyword{importpkg} keywords}
187 Keyword \\keyword{importpkg} is an analogous of \\verb|\\usepackage| in \\LaTeX.
188 If \\keyword{importpkg} is in the main paragraph, it acts just a normal word.
189 In other words, \\keyword{importpkg} actives only in the preamble.
190
191 importpkg has two different syntax. First one is same as docclass.
192 useenv center {
193 \\keyword{importpkg}\\kern1em pkg-name\\kern1em {\\tt(<arguments>{\\tt})}
194 }
195 Here, arguments are separated by commas and embraced by {\\tt ()}.
196 In the practical case, one should include several packages with options.
197 importpkg also supports such case. We will look at an example instead of
198 giving rigorous grammar.
199 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
200 %#+keyword|importpkg@ {
201 %# amsmath, amssymb, amsthm,
202 %# geometry (a4paper, margin=2.2cm),
203 %#}
204 }

```



\noindent As one can see, inside of \verb|{}|, several packages can be used together with thier options.

\subsection{\keyword{startdoc} keyword}  
Keyword \keyword{startdoc} tells to Vesti that the main document starts. In the main document, you can also write \keyword{startdoc} in the main document. In that case, \keyword{startdoc} does nothing.

\subsection{\useenv keyword}  
As the name implies, keyword \useenv is an analogous of \verb|\begin{...}| and \verb|\end{...}| pair in \LaTeX.  
The simplest \useenv is like this.

```
useenv figure [ht] {
 \centering
 useenv tikzpicture {
 useenv scope {
 \path (0,0) node {\vbox{
 \%#\hbox{\tt\useenv center \{}
 \%#\hbox{\tt\obeyspaces Hello, World!}
 \%#\hbox{\tt\obeyspaces\}}
 }};
 }
 \path (2.3,0) node {or};
 useenv scope [shift={(6,0)}]{
 \path (0,0) node {\tt\useenv center \{ Hello, World! \}};
 }
 }
}
```

As you can see, {\tt\useenv center} is the part of \verb|\begin{center}|, and the single {\tt\}} is the part of \verb|\end{center}|. Since Vesti knows their pair, one can write a code with several environment, and each pair is properly matched. For instance, above example is written in Vesti like follows. Here, \verb|\useenv| just prints \useenv in that style.

```
useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
 \%#|+color|purple@useenv@ figure [ht] {
 \%# \centering
 \%# |+color|purple@useenv@ tikzpicture {
 \%# |+color|purple@useenv@ scope {
 \%# \path (0,0) node {\vbox{
 \%# |+color|blue@%#\hbox{\tt\useenv center \{}
 \%# |+color|blue@%#\hbox{\tt\obeyspaces Hello, World!}
 \%# |+color|blue@%#\hbox{\tt\obeyspaces\}}
 }};
 \%# }
 \%# \path (2.3,0) node {or};
 \%# |+color|purple@useenv@ scope [shift={(6,0)}] {
 \%# \path (0,0) node {\tt\useenv center \{ Hello, World! \}};
 \%# }
 \%# }
 \%# }
 \%# }
 }
}
```

Full syntax about \useenv is the following.

```
useenv center {
 \useenv\kern0.5em <environment name>\kern0.5em <argument>*\kern0.5em
 {\tt\{} <body> {\tt\}}
```

```

265 }
266 where '*' means that the number of <argument> is zero or at least one, and
267 $$
268 "<argument>" = useenv cases {
269 "<argument>" & "mandatory arguments" \\
270 "[<argument>]" & "optional arguments"
271 }
272 $$
273 For instance, below one is a valid Vesti code (environment {\tt foo} is
274 undefined in general). As one can see, spaces can exist in between <argument>s.
275 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
276 %#+color|purple@useenv@ foo (asd)(fff)[\ames and \awdsa] (askws) [\rrsaa] {
277 %# foobar
278 %#}
279 }
280
281 \subsection{\begenv keyword and \endenv keyword}
282 As the name implies, both keywords \begenv and \endenv are analogous of
283 \verb|\begin{...}| and \verb|\end{...}| pair in \LaTeX, respectively.
284 Thus below code
285 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
286 %#+color|purple@begenv@ center
287 %# asdsad
288 %#+color|purple@endenv@ center
289 }
290 is exactly same as
291 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
292 %#+color|purple@useenv@ center {
293 %# asdsad
294 %#}
295 }
296 Then why we need \begenv and \endenv if we already have \useenv*?
297
298 \section{Builtins}
299 Vesti also has its own builtin functions, which are prefixed with \#.
300 One might wonder what distinguishes builtins from keywords. In fact, from the
301 compiler's internal perspective, there is no real difference. However, in actual
302 language usage, constantly typing the prefix can be somewhat tedious, especially
303 for functions that are commonly used.
304
305 From the perspective of language design --particularly in Vesti-- it is
306 sometimes desirable to use names that cannot serve as keywords. For example,
307 Vesti provides a built-in function {\tt\#label}, which will be explained later.
308 Since Vesti is a typewriting-oriented language, the word \lq\lq label\rq\rq is
309 often used in its ordinary sense rather than in its special semantic meaning
310 within the language.
311
312 Followings are reserved as builtin functions.
313
314 useenv table [ht] {
315 \centering
316 #lu:
317 local content = read_all("../src/lexer/Token.zig")
318
319 -- match .{ "here" }
320 local pat = "%.{%s*\"([^\"]+)\">%s*%}"
321
322 local builtins = {}
323 for name, tok in content:gmatch(pat) do
324 builtins[#builtins + 1] = name

```

```

325 end
326 table.sort(builtins)
327
328 vesti.print([[\\begin{tabular}{cccc}]]
329
330 for i, kw in ipairs(builtins) do
331 local cell = string.format("\\builtin@%s@", kw)
332 if (i % 5) == 0 then
333 vesti.print(cell .. [[\\]])
334 else
335 vesti.print(cell .. "&")
336 end
337 end
338
339 vesti.print([[\\end{tabular}]]
340 :lu#[readAll]
341 \\caption{Builtins in Vesti}
342 }
343 Since \\tt fancyvrb does not allow verbatim-like commands inside of
344 \\tt Verbatim environment, builtins are not colored in example codes.
345
346 \\subsection{\\builtin|label| builtin}
347
348 \\subsection{\\builtin|eq| builtin}
349 \\builtin|eq| is a abbreviation of \\lq\\keyword{% -useenv-%} equation\\rq.
350 Actually, below codes are same.
351
352 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
353 %##label(eq:1)
354 %#+keyword|useenv@ equation {
355 %# \\sum_{n=1}^\\infty {1/{n^2}} = {\\pi^2}/{6}.
356 %#}
357 }
358 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
359 %##eq(eq:1) {
360 %# \\sum_{n=1}^\\infty {1/{n^2}} = {\\pi^2}/{6}.
361 %#}
362 }
363 Since Vesti is used in the mathematical documents mainly, small syntactic suger
364 is needed, so Vesti introduce \\builtin|eq| builtin.
365
366 \\subsection{\\builtin|mathchardef| builtin}
367 \\builtin|mathchardef| defines a math token using the unicode codepoint.
368 Here is the grammar of \\builtin|mathchardef| builtin.
369 useenv center {
370 \\builtin|mathchardef|\\ob<kind>\\ob<font-num>\\ob<unicode-codepoint>\\ob<function>
371 }
372 Before explaining each parameter, introduce some examples.
373 useenv itemize {
374 \\item \\builtin|mathchardef|\\ob.opening\\ob0\\ob29d8\\ob\\verb|\\lfoo|
375 \\item \\builtin|mathchardef|\\ob.ordinary\\ob0\\ob2202\\ob\\verb|\\diff|
376 }
377
378 #showfont 0
379
380 \\section{Source Code of This Document}
381 Below code was generated by inline lua.
382 useenv Verbatim [numbers=left, numbersep=5pt, frame=single] {
383 #lu:
384 local content = read_all("vesti_man.ves")

```

```
385 for line in content:gmatch("(^[r\n]*)\r?\n?") do
386 vesti.print(line)
387 end
388 :lu#[readAll]
389 }
390
```