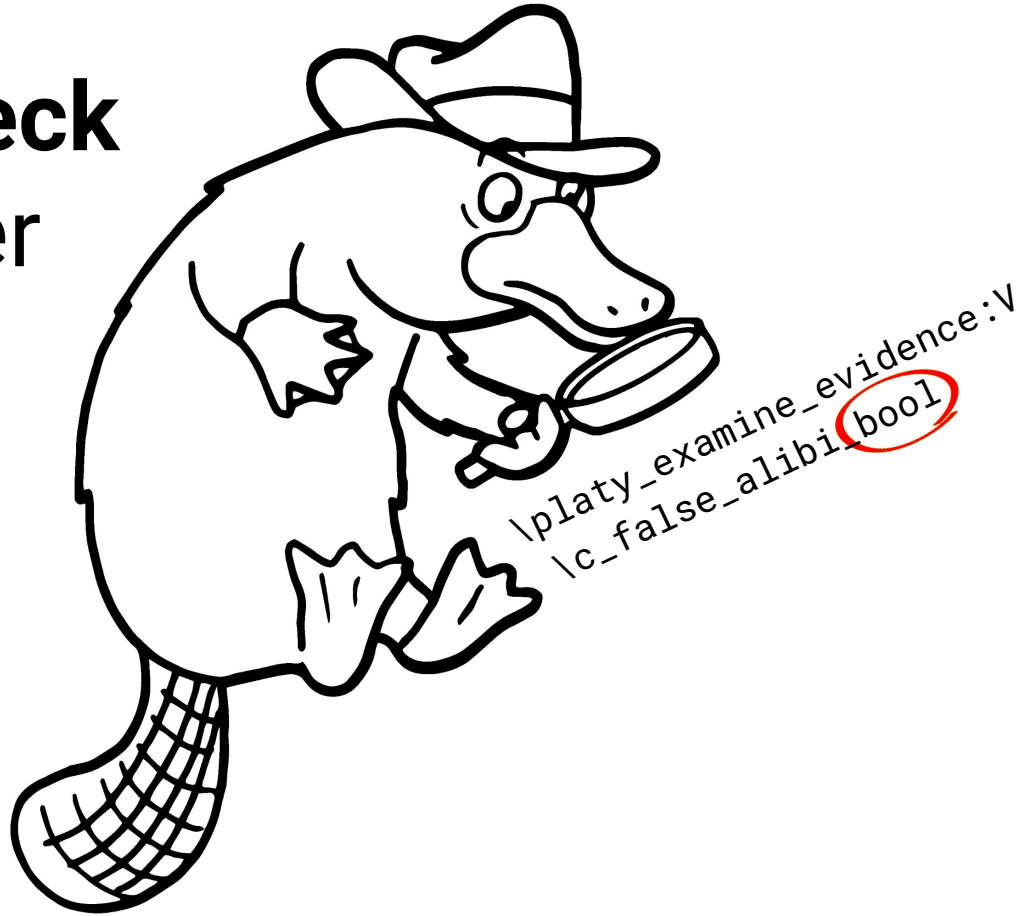
explcheck

a static analyzer
of expl3 code



github.com/witiko

cstug general assembly, brno
december 14, 2024

introduction static analysis

idea automatically determine if a tex program is correct

problem *rice's theorem* semantic properties of programs are undecidable

conclusion

can’t be done

thank you for your attention

1 introduction static analysis

idea automatically determine if a tex program is correct

problem 1 *rice's theorem* semantic properties of programs are undecidable
solutions *static analysis* annotate semantics in program syntax
dynamic analysis execute program to determine semantics

1 introduction expl3

idea automatically determine if a tex program is correct

problem 1 *rice's theorem* semantic properties of programs are undecidable
solution *static analysis* annotate semantics in program syntax

problem 2 tex's syntax is minimal and dynamic

```
\def\foo{...}  % a function, variable, or constant? what type is it? 🤔
\catcode`\<=0 <catcode`\ =0<catcode`\>=2<catcode`\==1 <def foo=...>  % 🤯
```

solution *expl3* a restricted and richly annotated subset of tex

```
\cs_new:Nn \foo:nnnn { ... }  % a public function with four arguments
\int_set:Nn \l_foo_int { ... }  % a local variable of type int
```

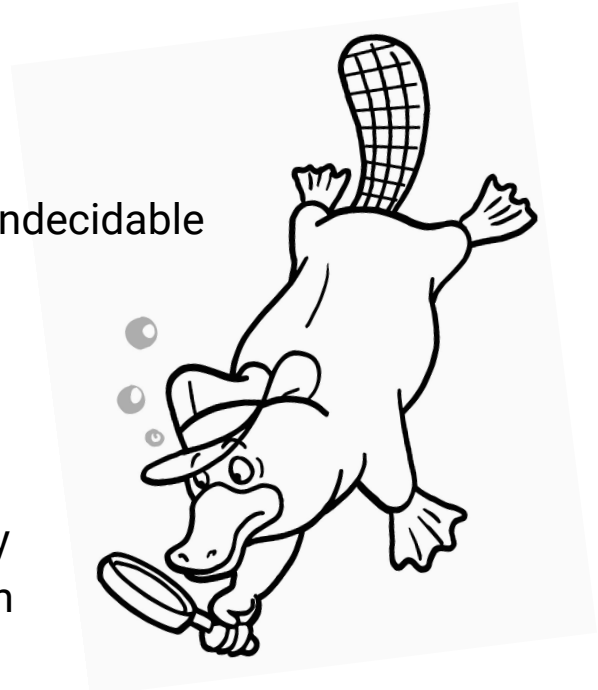
1 texdev.net/2024/11/11/the-mythical-latex3 2 ctan.org/pkg/l3kernel

1 introduction explcheck

idea automatically determine if a tex program is correct

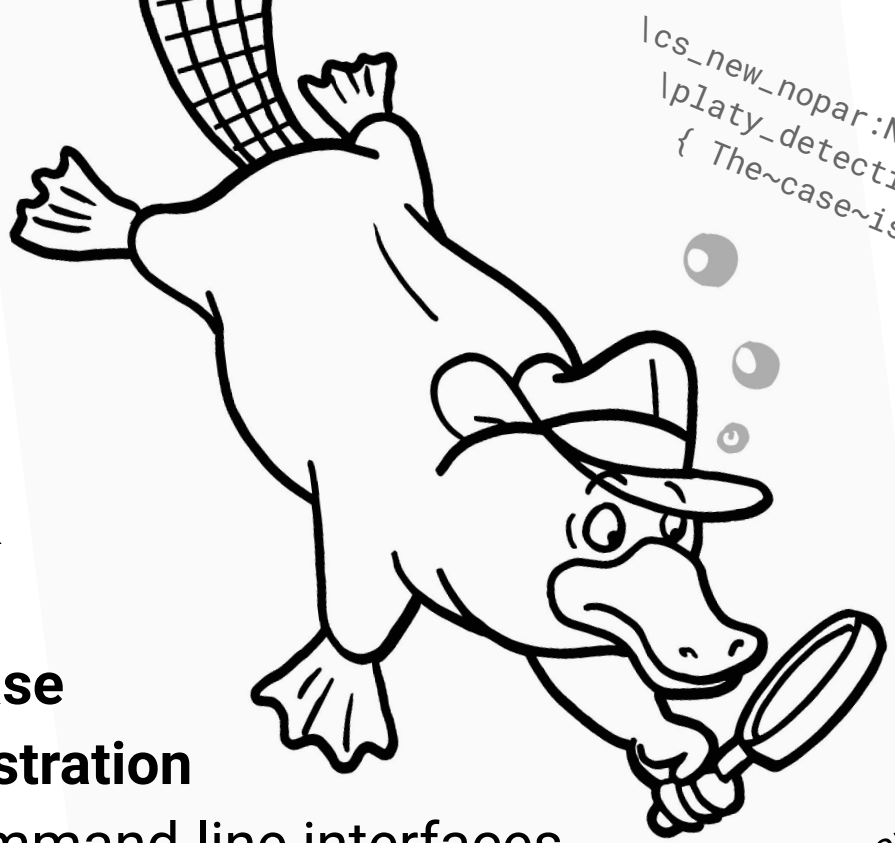
problem 1 *rice's theorem* semantic properties of programs are undecidable
solution *static analysis* annotate semantics in program syntax

problem 2 tex's syntax is minimal and dynamic
solution *expl3* a restricted and richly annotated subset of tex
explcheck a static analyzer (aka *linter*) of expl3 programs
platy explcheck mascot, an underwater detective, cc by
idea: paulo cereda, artist: fiverr.com/quickcartoon



3 witiko.github.io/Expl3-Linter-1 (april) 4 witiko.github.io/Expl3-Linter-2 5 witiko.github.io/Expl3-Linter-3
6 witiko.github.io/Expl3-Linter-4 7 tug.org/tc/devfund/documents/2024-09-expltools.pdf (september)

contents



```
\cs_new_nopar:Nn
 \platy_detective_report:
  { The~case~is~closed. \par }
```

```
\cs_generate_variant:Nn
 \platy_mission_done:
  { no }
```

2 project proposal 1 requirements *functional*

the linter should
1 accept a list of input expl3 files
2 process each input file
3 print out issues

initially, the linter should recognize at least issues with
style overlong lines, missing stylistic whitespaces, malformed names
functions multiple definitions, calling undefined functions, unused private functions, …
variables multiple declarations, setting undeclared variables, using undefined variables, …

2 project proposal 1 requirements *non-functional*

issues the linter should make distinction between *warnings* and *errors*
the design goal should be *robustness* to unexpected input and *precision over recall*
each issue should be assigned a *unique identifier* for ignoring it

architecture the linter should be written in base lua 5.3, initial dependence on luatex is ok
the linter should process input files in a series of steps, each available as lua module
the linter should support language server protocol (lsp)

validation each issue recognized by the linter should have at least one associated test
after project (september 2025), large-scale validation of *all* expl3 programs in tex live

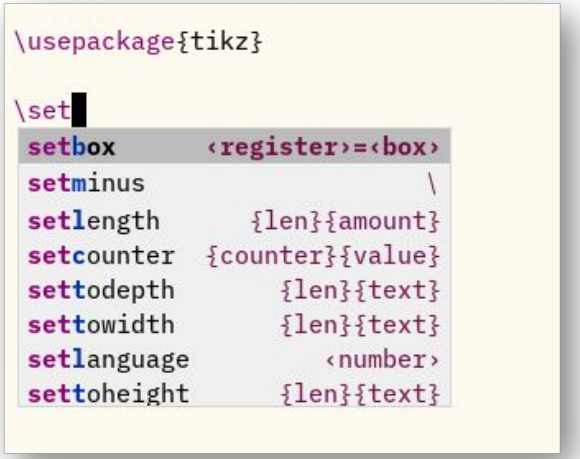
license dual: gnu gpl 2.0 or later, latex project public license (lppl) 1.3c or later

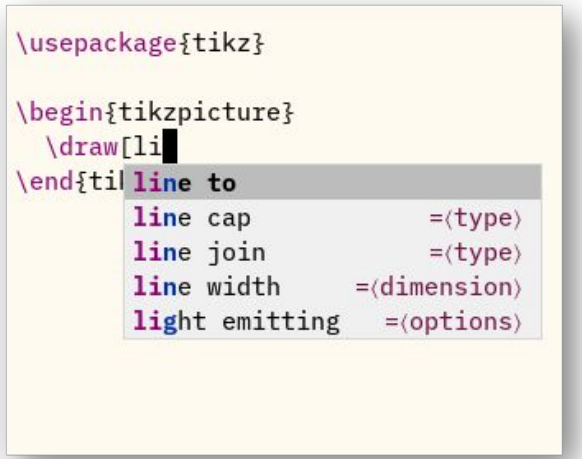
2 project proposal 2 related work

static analysis	*chktex, lacheck, chklref*	latex documents
	match_parens	tex and expl3 programs and documents
	(*luacheck, flake8*	lua and python, personal influence)
dynamic analysis	*cmdtrack, nag*	latex documents
	unravel, tex2tok	tex and expl3 programs and documents
language servers	*texlab, digestif*	tex programs and documents

5 witiko.github.io/Expl3-Linter-3 8 ctan.org/topic/debug-supp 9 ctan.org/topic/latex-qual


\usepackage{ti
tikz
tipa
tipx
times
tikz-cd
titleps
titlesec
titletoc
tikzexternal


\usepackage{tikz}
\set
setbox ‹register›=‹box›
setminus \
setlength {len}{amount}
setcounter {counter}{value}
settodepth {len}{text}
settowidth {len}{text}
setlanguage ‹number›
settoheight {len}{text}


\usepackage{tikz}
\begin{tikzpicture}
\draw[li
\end{til
line to
line cap =⟨type⟩
line join =⟨type⟩
line width =⟨dimension⟩
light emitting =⟨options⟩

Sed sint molestiae sed odit voluptate quia vero. Veniam accusantium explicabo vero error eos perferendis. Consequatur laudantium\kern 0.5pt ut vel. Dolorem ut ab architecto totam tempore ipsum. Atque molestiae illum facilis nulla perspiciatis.

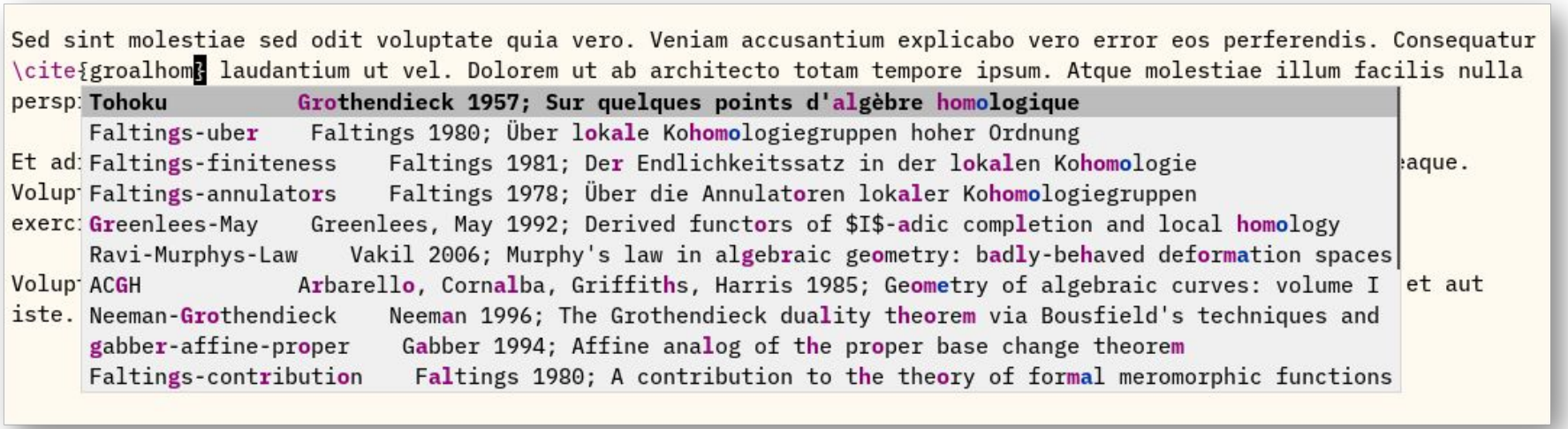
Et adipisci et a incidunt placeat. Voluptatum odit eius qui quam sint. Ab et tenetur esse ut repellat non eaque. Voluptates hic voluptatem officiis quia. Itaque libero est et laudantium facilis ut quis et. Veritatis et exercitationem ea officia laborum.

`\kern‹dimen›`: Produce a specified amount of space at which a break is not allowed.

Details

The effect of this command depends on the mode that TeX is in when it encounters it:

- In a horizontal mode, TeX moves its position to the right (for a positive kern) or to the left (for a negative kern).
- In a vertical mode, TeX moves its position down the page (for a positive kern) or up the page (for a negative kern).


Sed sint molestiae sed odit voluptate quia vero. Veniam accusantium explicabo vero error eos perferendis. Consequatur
\cite{groalhom} laudantium ut vel. Dolorem ut ab architecto totam tempore ipsum. Atque molestiae illum facilis nulla
Tohoku Grothendieck 1957; Sur quelques points d'algèbre homologique
Faltings-uber Faltings 1980; Über lokale Kohomologiegruppen hoher Ordnung
Faltings-finiteness Faltings 1981; Der Endlichkeitssatz in der lokalen Kohomologie
Faltings-annulators Faltings 1978; Über die Annulatoren lokaler Kohomologiegruppen
Greenlees-May Greenlees, May 1992; Derived functors of I-adic completion and local homology
Ravi-Murphys-Law Vakil 2006; Murphy's law in algebraic geometry: badly-behaved deformation spaces
ACGH Arbarello, Cornalba, Griffiths, Harris 1985; Geometry of algebraic curves: volume I
Neeman-Grothendieck Neeman 1996; The Grothendieck duality theorem via Bousfield's techniques and
gabber-affine-proper Gabber 1994; Affine analog of the proper base change theorem
Faltings-contribution Faltings 1980; A contribution to the theory of formal meromorphic functions

2 project proposal 2 related work

static analysis	*chktex, lacheck, chklref*	latex documents
	match_parens	tex and expl3 programs and documents
	(*luacheck, flake8*	lua and python, personal influence)
dynamic analysis	*cmdtrack, nag*	latex documents
	unravel, tex2tok	tex and expl3 programs and documents
language servers	*texlab, digestif*	tex programs and documents

5 witiko.github.io/Expl3-Linter-3 8 ctan.org/topic/debug-supp 9 ctan.org/topic/latex-qual

2 project proposal 3 design

processing steps	*preprocessing*	determine which parts of the input files contain expl3 code
	lexical analysis	convert expl3 parts of the input files into tex tokens
	syntactic analysis	convert tex tokens into a tree of function calls
	semantic analysis	determine the meaning of the different function calls
	(pseudo-)flow analysis	determine additional emergent properties of the code
warnings and errors	40 warnings and 26 errors with examples[10]	

6 witiko.github.io/Expl3-Linter-4 10 github.com/witiko/expltools/releases/download/2024-09-06/warnings-and-errors.pdf

Warnings and errors for the expl3 analysis tool

Vít Starý Novotný

September 6, 2024

Contents

2 project proposal 3 design

processing steps *preprocessing* determine which parts of the input files contain expl3 code
lexical analysis convert expl3 parts of the input files into tex tokens
syntactic analysis convert tex tokens into a tree of function calls
semantic analysis determine the meaning of the different function calls
(pseudo-)flow analysis determine additional emergent properties of the code

warnings and errors 40 warnings and 26 errors with examples[10]

limitations initial version will make some naïve assumptions[10, section caveats]
1 assume default expl3 catcodes everywhere
2 ignore non-expl3 and third-party code
3 do not analyze expansion and key–value calls

6 witiko.github.io/Expl3-Linter-4 10 github.com/witiko/expltools/releases/download/2024-09-06/warnings-and-errors.pdf

Caveats

The warnings and errors in this documents do not cover the complete expl3 language. The caveats currently include the following areas, among others:

- Functions with "weird" (`w`) argument specifiers
- Verifying the `nopar` restriction on functions [2, Section 4.3.1]
- Symbolic evaluation of expansion functions [2, sections 5.4–5.10]
- Validation of parameters in (inline) functions (c.f. E423 and E519)
- Shorthands such as `\~` and `\\` in message texts [2, sections 11.4 and 12.1.3]
- Quotes in shell commands and file names [2, Section 10.7 and Chapter 12]
- Functions used outside their intended context:
  - `\sort_return_*:` outside comparison code [2, Section 6.1]
  - `\prg_return_*:` outside conditional functions [2, Section 9.1]
  - Predicates (`\*_p:*`) outside boolean expressions [2, Section 9.3]
  - `\*_map_break:*` outside a corresponding mapping [2, sections 9.8]
  - `\msg_line_*:`, `\iow_char:N`, and `\iow_newline:` outside message text [2, sections 11.3 and 12.1.3]
  - `\iow_wrap_allow_break:` and `\iow_indent:n` outside wrapped message text [2, Section 12.1.4]
  - Boolean variable without an accessor function `\bool_to_str:N` outside boolean expressions [2, Section 21.4] (see E413)
  - Integer variable without an accessor function `\int_use:N` outside integer or floating point expressions [2, Section 21.4] (see E413)
  - Dimension variable without an accessor function `\dim_use:N` outside dimension or floating point expressions [2, Section 26.7] (see E413)
  - Skip variable without an accessor function `\skip_use:N` outside skip

2 project proposal 3 design

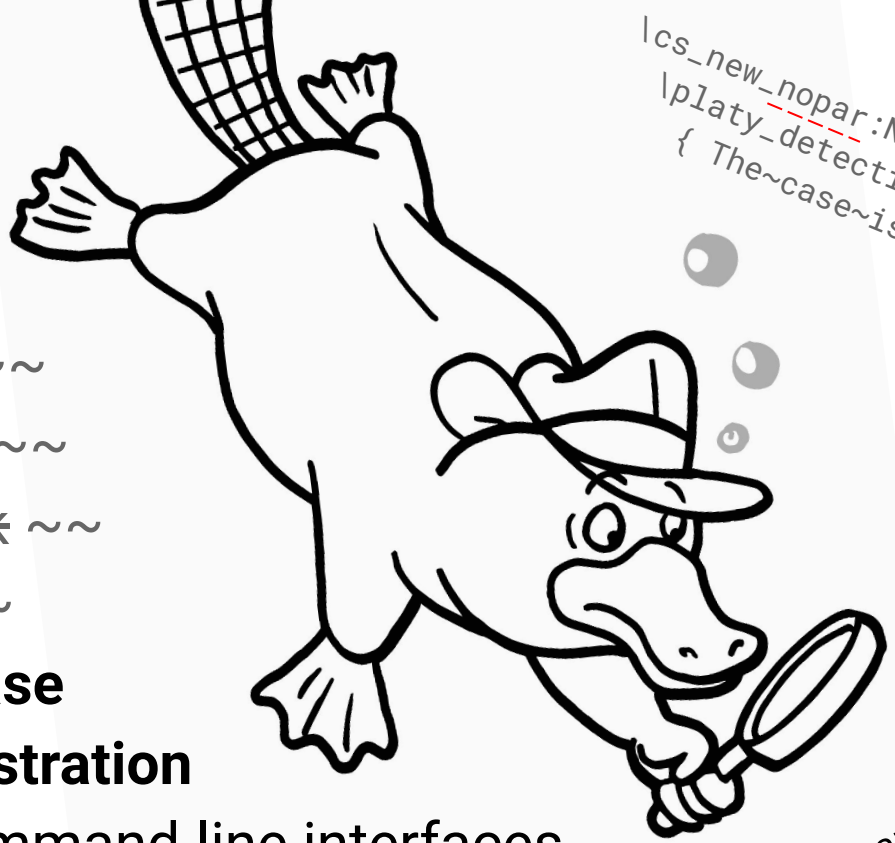
processing steps *preprocessing* determine which parts of the input files contain expl3 code
lexical analysis convert expl3 parts of the input files into tex tokens
syntactic analysis convert tex tokens into a tree of function calls
semantic analysis determine the meaning of the different function calls
(pseudo-)flow analysis determine additional emergent properties of the code

warnings and errors 40 warnings and 26 errors with examples[10]

limitations initial version will make some naïve assumptions[10, section caveats]
1 assume default expl3 catcodes everywhere
2 ignore non-expl3 and third-party code
3 do not analyze expansion and key–value calls

6 witiko.github.io/Expl3-Linter-4 10 github.com/witiko/expltools/releases/download/2024-09-06/warnings-and-errors.pdf

contents



```
\cs_new_nopar:Nn
  \platy_detective_report:
  { The~case~is~closed. \par }
```

```
\cs_generate_variant:Nn
  \platy_mission_done:
  { no }
```

3 first release

v0.1.0 released on 2024-12-04 at github.com/witiko/expltools and ctan.org/pkg/expltools

processing steps *preprocessing* determine which parts of the input files contain expl3 code
warnings and errors 3 warnings and 2 errors out of 40 warnings and 27 errors (added e104)

Warnings and errors for the expl3 analysis tool

Vít Starý Novotný

2024-12-04

Contents

3 first release

v0.1.0 released on 2024-12-04 at github.com/witiko/expltools and ctan.org/pkg/expltools

processing steps *preprocessing* determine which parts of the input files contain expl3 code
warnings and errors 3 warnings and 2 errors out of 40 warnings and 27 errors (added e104)

on 2024-12-05, i gave this talk to frank mittelbach on petr sojka's seminar on digital typography

11 github.com/witiko/expltools/releases/tag/2024-12-04 12 witiko.github.io/Expl3-Linter-5


Comparison – spreads 3 + 4
greedy viz, optimal, 0 turns
Rudolf Zilka
'0 STAGES OF DEBUGGING
1 introduction static analysis
idea automatically determine if a tea program is correct
problem 1 rice's theorem semantic properties of programs are undecidable
solutions static analysis annotate semantics in program syntax
dynamic analysis execute program to determine semantics

3 first release

v0.1.0 released on 2024-12-04 at github.com/witiko/expltools and ctan.org/pkg/expltools

processing steps *preprocessing* determine which parts of the input files contain expl3 code
warnings and errors 3 warnings and 2 errors out of 40 warnings and 27 errors (added e104)

on 2024-12-05, i gave this talk to frank mittelbach on petr sojka's seminar on digital typography

early adopters include my packages *markdown* and *lt3luabridge* (dogfood, more in next section)

v0.1.1 released on 2024-12-09, addressing bugs reported by latex team and early adopters

v0.2.0 released on 2024-12-13, adding support for machine-readable compiler output

11 github.com/witiko/expltools/releases/tag/2024-12-04 12 witiko.github.io/Expl3-Linter-5
13 github.com/witiko/expltools/releases/tag/2024-12-09 14 github.com/witiko/expltools/releases/tag/2024-12-13

4 demonstration 1 command-line interfaces

command-line interface for distributors:

l3build *check* run all tests in the repository

tag, ctan prepare a release archive for upload to ctan.org

Run tests
succeeded 10 minutes ago in 8s

Search logs

- Set up job 1s
- Checkout repository 1s
- Install TeX Live 5s
- Run tests 0s

```
1  ▶ Run l3build check
6  Running l3build with target "check" for module "explcheck"
7  Running 7 tests
8
9  Checking w100.lua       OK
10 Checking w100-01.lua    OK
11 Checking w100-02.lua    OK
12 Checking w101.lua       OK
13 Checking e102.lua       OK
14 Checking s103.lua       OK
15 Checking e104.lua       OK
16
17 Total: 0 errors in 7 files
```

- Post Install TeX Live 0s
- Post Checkout repository 0s

4 demonstration 1 command-line interfaces

command-line interface for distributors:

l3build *check* run all tests in the repository

tag, ctan prepare a release archive for upload to ctan.org

command-line interface for users:

explcheck.lua usage: *explcheck [options] [filenames]*

options *--max-line-length=N* maximum line length

--porcelain produce machine-readable output

--warnings-are-errors produce non-zero exit code after warnings

Check code style (expl3)
succeeded 4 days ago in 9s

- Set up job — 1s
- Checkout repository — 1s
- Install TeX Live — 4s
- Run explcheck — 0s
- Post Install TeX Live
- Post Checkout repository
- Complete job

```
1  ▶ Run set -e
9  Checking 3 files
10
11 Checking explcheck/support/markdownthemewitiko_expltools.sty          OK
12 Checking ...wnthemewitiko_expltools_explcheck_warnings-and-errors.sty OK
13 Checking ...kdownthemewitiko_expltools_explcheck_project-proposal.sty OK
14
15 Total: 0 errors, 0 warnings in 3 files
```

Run explcheck

```
1  ▶ Run explcheck --warnings-are-errors -- *.tex *.sty
7  Checking 6 files
8
9  Checking markdown.tex                                          OK
10 Checking markdownthemewitiko_markdown_defaults.tex             OK
11 Checking t-markdown.tex                                        OK
12 Checking t-markdownthemewitiko_markdown_defaults.tex           OK
13 Checking markdown.sty                                          OK
14 Checking markdownthemewitiko_markdown_defaults.sty             OK
15
16 Total: 0 errors, 0 warnings in 6 files
```

Run explcheck

```
1  ▶ Run explcheck --warnings-are-errors -- lt3luabridge.tex
6  Checking 1 files
7
8  Checking lt3luabridge.tex                                      OK
9
10 Total: 0 errors, 0 warnings in 1 file
```

4 demonstration 1 command-line interfaces

command-line interface for distributors:

l3build *check* run all tests in the repository

tag, ctan prepare a release archive for upload to ctan.org

command-line interface for users:

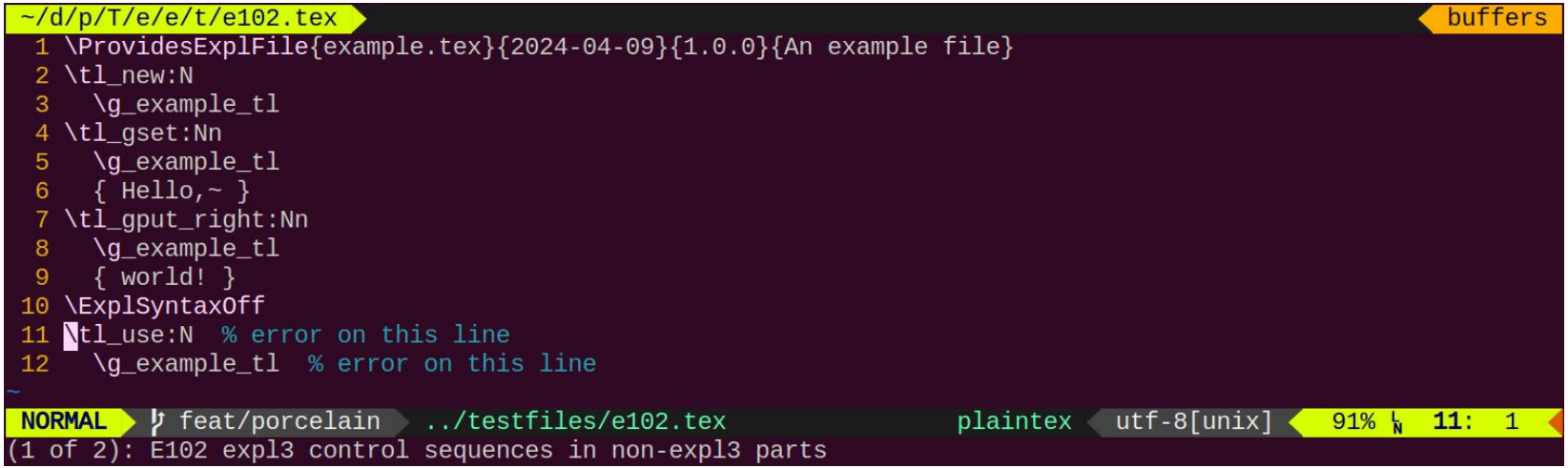
explcheck.lua usage: *explcheck [options] [filenames]*

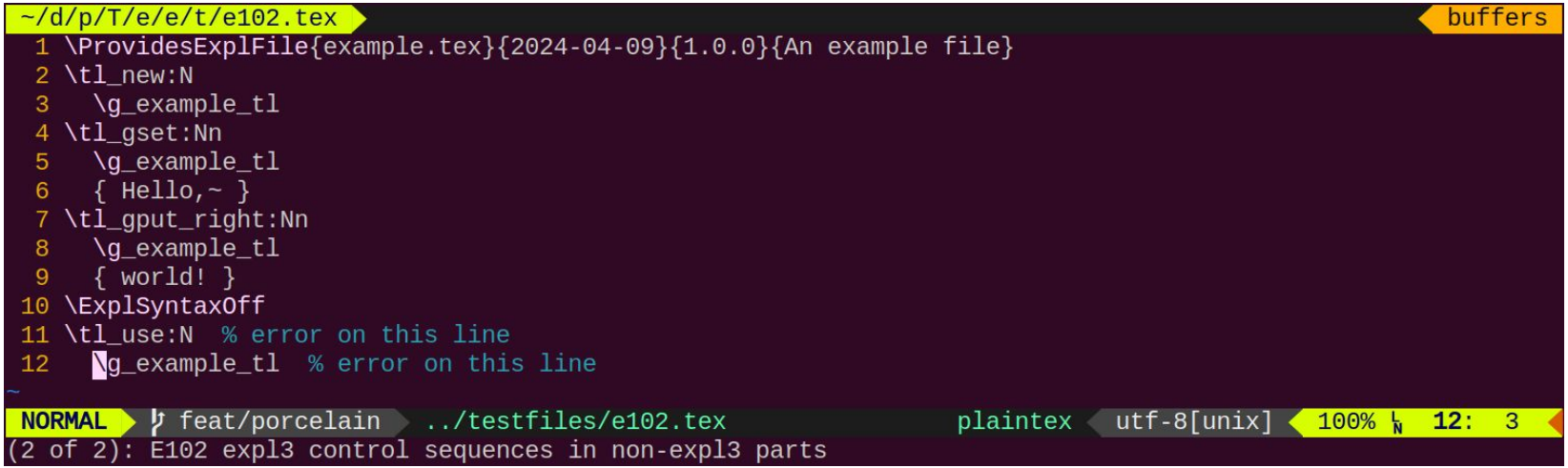
options *--max-line-length=N* maximum line length

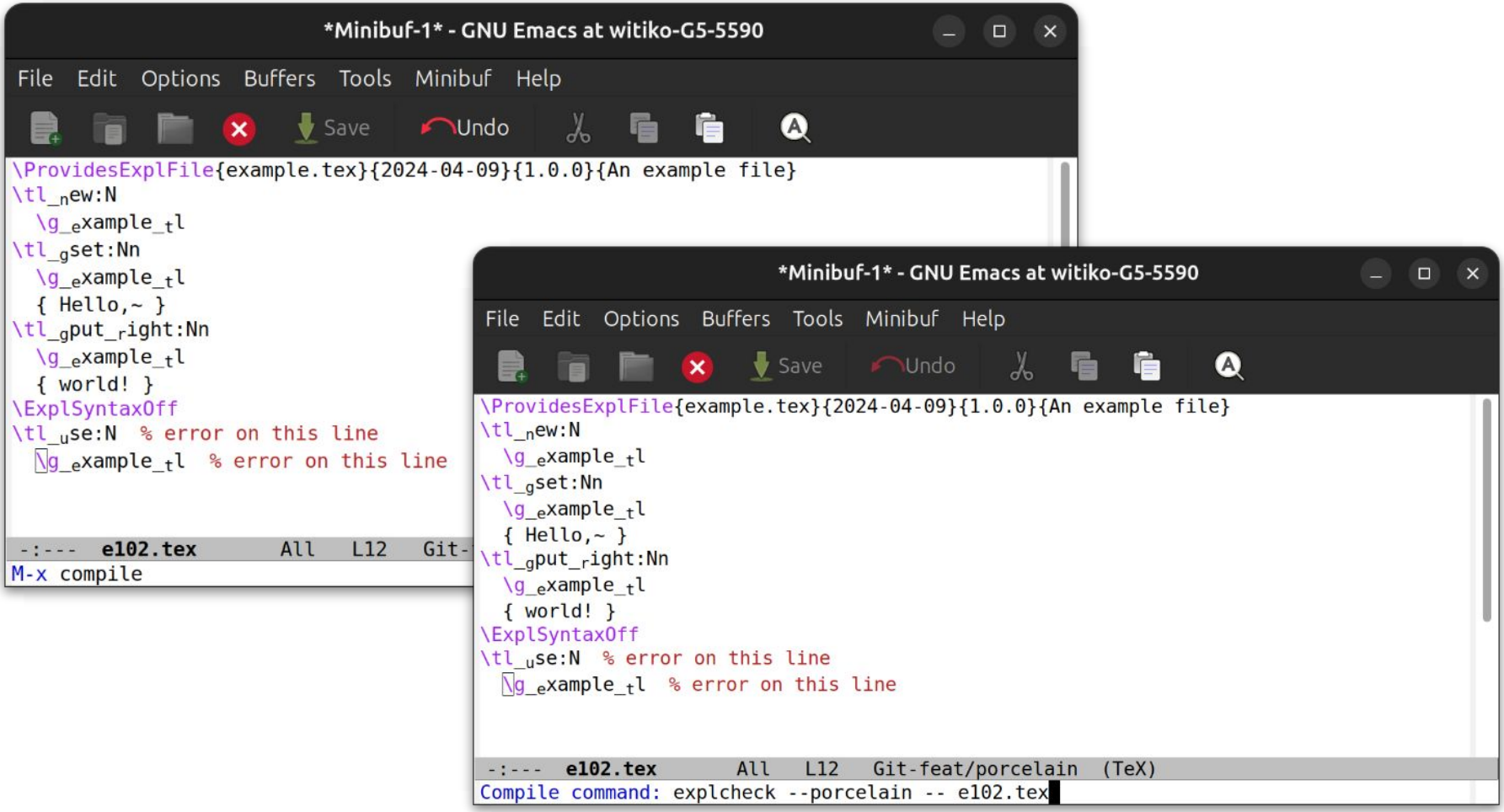
--porcelain produce machine-readable output

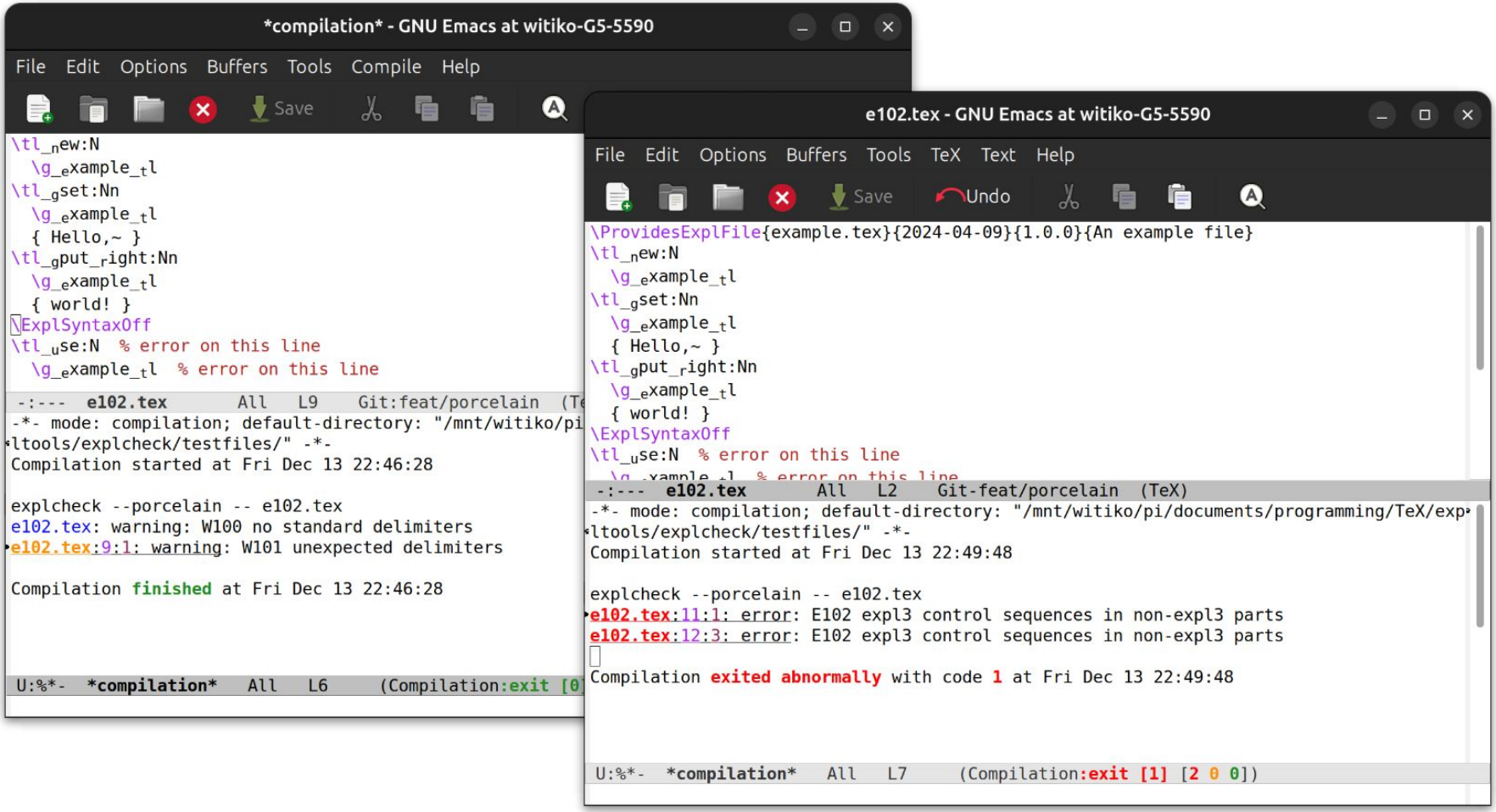
--warnings-are-errors produce non-zero exit code after warnings

15 github.com/Witiko/expltools/pull/15


~/d/p/T/e/e/t/e102.tex
buffers
1 \ProvidesExplFile{example.tex}{2024-04-09}{1.0.0}{An example file}
2 \tl_new:N
3 \g_example_tl
4 \tl_gset:Nn
5 \g_example_tl
6 { Hello,~ }
7 \tl_gput_right:Nn
8 \g_example_tl
9 { world! }
10 \ExplSyntaxOff
11 \tl_use:N % error on this line
12 \g_example_tl % error on this line
NORMAL feat/porcelain ../testfiles/e102.tex plaintex utf-8[unix] 91% ℓ N 11: 1
(1 of 2): E102 expl3 control sequences in non-expl3 parts


~/d/p/T/e/e/t/e102.tex
buffers
1 \ProvidesExplFile{example.tex}{2024-04-09}{1.0.0}{An example file}
2 \tl_new:N
3 \g_example_tl
4 \tl_gset:Nn
5 \g_example_tl
6 { Hello,~ }
7 \tl_gput_right:Nn
8 \g_example_tl
9 { world! }
10 \ExplSyntaxOff
11 \tl_use:N % error on this line
12 \g_example_tl % error on this line
NORMAL feat/porcelain ../testfiles/e102.tex plaintex utf-8[unix] 100% ℓ N 12: 3
(2 of 2): E102 expl3 control sequences in non-expl3 parts


Minibuf-1 - GNU Emacs at witiko-G5-5590
File Edit Options Buffers Tools Minibuf Help
Save Undo
\ProvidesExplFile{example.tex}{2024-04-09}{1.0.0}{An example file}
\tl_new:N
\g_example_tl
\tl_gset:Nn
\g_example_tl
{ Hello,~ }
\tl_gput_right:Nn
\g_example_tl
{ world! }
\ExplSyntaxOff
\tl_use:N % error on this line
\g_example_tl % error on this line
-:--- e102.tex All L12 Git-
M-x compile
Minibuf-1 - GNU Emacs at witiko-G5-5590
File Edit Options Buffers Tools Minibuf Help
Save Undo
\ProvidesExplFile{example.tex}{2024-04-09}{1.0.0}{An example file}
\tl_new:N
\g_example_tl
\tl_gset:Nn
\g_example_tl
{ Hello,~ }
\tl_gput_right:Nn
\g_example_tl
{ world! }
\ExplSyntaxOff
\tl_use:N % error on this line
\g_example_tl % error on this line
-:--- e102.tex All L12 Git-feat/porcelain (TeX)
Compile command: explcheck --porcelain -- e102.tex


compilation - GNU Emacs at witiko-G5-5590
File Edit Options Buffers Tools Compile Help
Save
\tl_new:N
\g_example_tl
\tl_gset:Nn
\g_example_tl
{ Hello,~ }
\tl_gput_right:Nn
\g_example_tl
{ world! }
\ExplSyntaxOff
\tl_use:N % error on this line
\g_example_tl % error on this line
-:--- e102.tex All L9 Git:feat/porcelain (Te
-*- mode: compilation; default-directory: "/mnt/witiko/pi
ltools/explcheck/testfiles/" -*-
Compilation started at Fri Dec 13 22:46:28
explcheck --porcelain -- e102.tex
e102.tex: warning: W100 no standard delimiters
e102.tex:9:1: warning: W101 unexpected delimiters
Compilation finished at Fri Dec 13 22:46:28
U:%*- *compilation* All L6 (Compilation:exit [0
e102.tex - GNU Emacs at witiko-G5-5590
File Edit Options Buffers Tools TeX Text Help
Save
Undo
\ProvidesExplFile{example.tex}{2024-04-09}{1.0.0}{An example file}
\tl_new:N
\g_example_tl
\tl_gset:Nn
\g_example_tl
{ Hello,~ }
\tl_gput_right:Nn
\g_example_tl
{ world! }
\ExplSyntaxOff
\tl_use:N % error on this line
-:--- e102.tex All L2 Git-feat/porcelain (TeX)
-*- mode: compilation; default-directory: "/mnt/witiko/pi/documents/programming/TeX/exp
ltools/explcheck/testfiles/" -*-
Compilation started at Fri Dec 13 22:49:48
explcheck --porcelain -- e102.tex
e102.tex:11:1: error: E102 expl3 control sequences in non-expl3 parts
e102.tex:12:3: error: E102 expl3 control sequences in non-expl3 parts
Compilation exited abnormally with code 1 at Fri Dec 13 22:49:48
U:%*- *compilation* All L7 (Compilation:exit [1] [2 0 0])

4 demonstration 1 command-line interfaces

command-line interface for distributors:

l3build *check* run all tests in the repository

tag, ctan prepare a release archive for upload to ctan.org

command-line interface for users:

explcheck.lua usage: *explcheck [options] [filenames]*

options *--max-line-length=N* maximum line length

--porcelain produce machine-readable output

--warnings-are-errors produce non-zero exit code after warnings

other interfaces:

explcheck-issues.lua, explcheck-preprocessing.lua lua library (more on that in next section)

ghcr.io/witiko/expltools/explcheck docker image (8M) with alpine linux, lua5.3, and explcheck

4 demonstration 2 expl3 usage statistics *method*

$ cat expl3-usage-statistics.sh

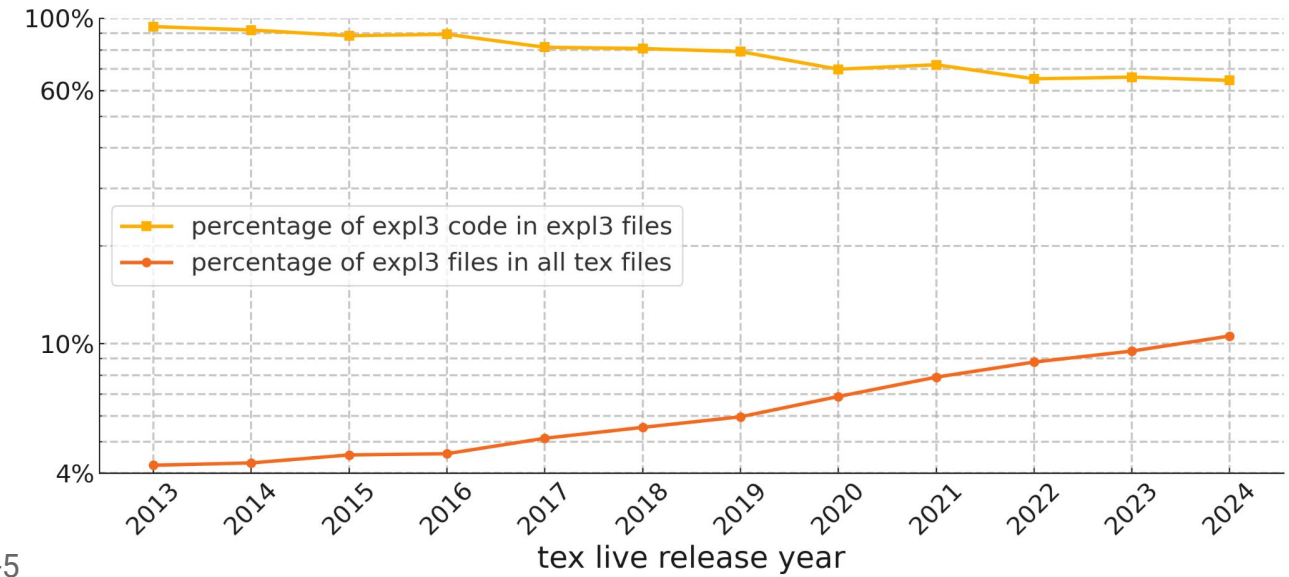
```
YEAR=$1; IMAGE=`(( YEAR == 2024 )) && echo latest || echo TL$YEAR-historic`; printf '%d,' $YEAR
docker run --rm -i -v "$PWD"/explcheck/src:/workdir:ro -w /workdir texlive/texlive:$IMAGE bash -c '
  find /usr/local/texlive/ -iregex ".*\.\(tex\|sty\|cls\|opt\)$" | texlua expl3-usage-statistics.lua'
```

$ cat expl3-usage-statistics.lua

```
local iss, pre = require("explcheck-issues"), require("explcheck-preprocessing")
local keywords = {"Expl", "\\tl_", "\\int_", "\\str_", "\\cs_", "\\prg_", "\\scan_", "\\group_",
"\\seq_", "\\clist_", "\\sort_", "\\regex_", "\\sys_", "\\msg_", "\\ior_", "\\iow_", "\\lua_",
"\\legacy_", "\\quark_", "\\flag_", "\\char_", "\\prop_", "\\dim_", "\\keys_", "\\intarray_", "\\fp_",
"\\fparray_", "\\bitset_", "\\cctab_", "\\codepoint_", "\\text_", "\\box_", "\\hbox_", "\\vbox_",
"\\coffin_", "\\color_", "\\pdf_", "\\q_", "\\g_", "\\l_", "\\c_"}
local files, expl_files, expl_bytes, code = 0, 0, 0, 0
for filename in io.lines() do local content = io.open(filename, "r"):read("*a") files = files + 1
  local is_expl = false for _, k in ipairs(keywords) do if content:find(k) then is_expl = true end end
  if not is_expl then goto cnt end expl_files, expl_bytes = expl_files + 1, expl_bytes + #content
  for _, range in ipairs(({pre(iss(), content)})[2]) do code=code+(range[2]-range[1]-1) end ::cnt:: end
print(("%.2f,%.2f"):format(100 * expl_files / files, 100 * code / expl_bytes))
```

4 demonstration 2 expl3 usage statistics *results*

```
$ parallel -j 1 -- bash expl3-usage-statistics.sh ::: {2013..2024}
2013,4.23,94.32
2014,4.30,92.06
2015,4.55,88.42
2016,4.59,89.35
2017,5.12,81.49
2018,5.53,80.74
2019,5.96,79.01
2020,6.88,69.75
2021,7.89,72.01
2022,8.78,65.18
2023,9.49,65.96
2024,10.56,64.44
```



5 conclusion

the correctness of tex programs is difficult to check
expl3 is rising in popularity, programs are much easier to analyze than with tex

explcheck is an expl3 linter, developed as part of the tex development fund[15]
current version implements the first processing step, early adopters are encouraged to try it

current version is *not* a complete vertical slice
1 can't ignore issues based on a config file, command-line options, or tex comments
2 no support for the language server protocol yet

in the following nine months, i should have the above and the other four processing steps 🤞

15 tug.org/tc/devfund