

# Инструкция по формированию отчёта БИОС из XML-файла с использованием шаблонизатора

# Оглавление

1. Краткое описание входных файлов . . . . .	3
1.1. Общее описание . . . . .	3
1.2. Группа файлов для заполнения шаблона . . . . .	3
1.3. Группа файлов LaTeX . . . . .	4
1.4. Группа файлов для запуска шаблонизатора через bat-файл . . . . .	5
1.5. Настройка файла конфигурации . . . . .	5
2. Краткое описание выходных файлов . . . . .	6
2.1. Общее описание . . . . .	6
2.2. При компиляции в TeXstudio . . . . .	6
2.3. При компиляции через bat-файл . . . . .	6
3. Подготовительные работы . . . . .	7
3.1. Подготовка файлов . . . . .	7
4. Компиляция файлов в TeXstudio . . . . .	7
4.1. Настройка программы TeXstudio . . . . .	7
4.2. Работа с компилируемым файлом . . . . .	11
4.3. Краткое описание результатов компиляции . . . . .	12
5. Компиляция файлов через пакетный файл . . . . .	19
5.1. Запуск шаблонизатора через bat-файл . . . . .	19
5.2. Краткое описание работы bat-файла . . . . .	24

# 1. Краткое описание входных файлов

## 1.1. Общее описание

1.1.1) Группа файлов 1.2 находится в папке «C:\avs\Utilities\Template\_generator\For DWNT From XML», группа файлов 1.3 расположена в папке «C:\avs\Racks\DWNT\docs\Reports». Эти две группы файлов необходимы для формирования PDF-отчёта из XML-файла через среду программирования TeXstudio. Группа файлов 1.4 является необязательной, но упрощает процесс запуска генерации PDF-отчёта.

## 1.2. Группа файлов, необходимых для заполнения шаблона сведениями из XML-файла:

1.2.1) `generate_report.py` — сценарий формирования отчёта. Является основным файлом среди файлов с расширением ".py". В данном файле описана инициализация всех важных объектов и файлов, а также взаимодействие с программным средством заполнения шаблонов Jinja2. Для нормальной работы `generate_report.py` необходимо наличие всех файлов группы 1.2;

1.2.2) `report_helper.py` — определение класса `ReportHelper`, содержащего вспомогательные функции для форматирования полученных данных из XML-файла и формирования TeX-структур, например, таблиц, графиков (в файле 1.2.1) и в файлах группы 1.3 обозначается переменной "h");

1.2.3) `for_read_xml.py` — файл, содержащий классы для чтения XML-файлов (класс `XMLParser` — общий класс для считывания всех XML-файлов, `XMLConfigParser` — класс для считывания конфигурационного файла 1.2.4), `XMLReportParser` (в файле 1.2.1) и в файлах группы 1.3 обозначается переменной "e") — для считывания данных с отчёта испытания `TestStand` в виде XML-файла);

1.2.4) `DWNT_Report_Configuration.xml` — файл конфигурации построения отчёта (о заполнении этого файла см. подробнее подраздел 1.5). Содержит тэги:

- а) `Headers`, описывающий типы заголовков формируемого отчёта и включающий в себя атрибуты:
  - 1) `Name` — наименование типа заголовка отчёта;
  - 2) `Head1` — первая строка заголовка;
  - 3) `Head2` — вторая строка заголовка;
  - 4) `Head3` — третья строка заголовка;
  - 5) `Date` — четвёртая строка заголовка (обычно в этой строке указывается дата и время начала тестирования, по которому формируется отчёт);
- б) `Report`, описывающий основные параметры о формируемом отчёте и включающий в себя атрибуты:
  - 1) `UUT_Name` — имя теста (на время написания инструкции атрибут носит только описательный характер и нигде не используется);
  - 2) `Headers` — имя используемого при формировании отчёта заголовка (если значение не совпадает ни с одним значением атрибута `Name` в тэгах `Headers`, то при использовании файла конфигурации в качестве заголовка определяется заголовок, описанный в файле 1.2.1) как элемент списка `list_of_headers` при первичной инициализации, иначе выбирается соответствующий заголовок из тэгов `Headers` с атрибутом `Name` таким же, как значение атрибута `Headers` в тэге `Report`).

### 1.3. Группа файлов LaTeX (группа файлов самого шаблонизатора), необходимых для генерации PDF-отчёта:

1.3.1) DWNT\_Report.tex — основной файл шаблонизатора. Заполняется сведениями из XML-файла с помощью сценария 1.2.1). DWNT\_Report.tex компилирует основной файл и включает в свой код содержимое других файлов данной группы;

1.3.2) TestDWNT1.tex — шаблон для первого шага;

1.3.3) TestDWNT2.tex — шаблон для второго шага;

1.3.4) TestDWNT3.tex — шаблон для третьего шага;

1.3.5) TestDWNT4.tex — шаблон для четвертого шага;

1.3.6) TestDWNT5.tex — шаблон для пятого шага;

1.3.7) TestDWNT6.tex — шаблон для шестого шага;

1.3.8) TestDWNT7.tex — шаблон для седьмого шага;

1.3.9) TestDWNT8.tex — шаблон для восьмого шага;

1.3.10) TestDWNT9.tex — шаблон для девятого шага;

1.3.11) TestDWNT10.tex — шаблон для десятого шага;

1.3.12) TestDWNT11.tex — шаблон для одиннадцатого шага;

1.3.13) TestDWNT12.tex — шаблон для двенадцатого шага;

1.3.14) TestDWNT13.tex — шаблон для тринадцатого шага;

1.3.15) TestDWNT14.tex — шаблон для четырнадцатого шага;

1.3.16) TestDWNT15.tex — шаблон для пятнадцатого шага;

1.3.17) TestDWNT16.tex — шаблон для шестнадцатого шага;

1.3.18) TestDWNT17.tex — шаблон для семнадцатого шага;

1.3.19) TestDWNT18.tex — шаблон для восемнадцатого шага;

1.3.20) TestDWNT19.tex — шаблон для девятнадцатого шага;

1.3.21) TestDWNT20.tex — шаблон для двадцатого шага;

1.3.22) TestDWNT21.tex — шаблон для двадцать первого шага;

1.3.23) TestDWNT22.tex — шаблон для двадцать второго шага;

1.3.24) TestDWNT23.tex — шаблон для двадцать третьего шага;

1.3.25) DWNT\_Get\_Status\_For\_Substep\_By\_Limits.tex — шаблон для получения статуса подшага испытания с помощью проверки вхождения полученного значения в отрезок допустимых значений (используется только в ситуациях, когда установленные пределы для подшага не содержатся в данном подшаге);

1.3.26) TestDWNTWriteNameStep.tex — шаблон построчного вывода имени текущего шага в шести центральных столбцах таблицы с учётом возможности переноса этого имени на следующую страницу и расчётом количества оставшихся свободных строк для этого шага на текущей странице с помощью шаблона 1.3.27);

1.3.27) DWNT\_Counter\_Of\_Rows.tex — шаблон для определения переноса страницы посредством подсчёта разницы между количеством оставшихся свободных строк на странице и высотой в строках следующего текстового объекта. Затем с помощью шаблона 1.3.28) вычисляется количество оставшихся свободных строк для этого шага на текущей странице;

1.3.28) DWNT\_Counter\_Of\_Share\_Rows.tex — шаблон для вычисления оставшегося на текущей странице количества свободных общих строк для текущего шага<sup>1</sup>. При нехватке этого количества используется шаблон рекурсивного подсчёта количества свободных общих строк 1.3.29).

1.3.29) DWNT\_Recursive\_Counter\_Of\_Share\_Rows.tex — шаблон рекурсивного вычисления количества свободных общих строк текущего шага на текущей странице.

---

<sup>1</sup> «Общими» эти строки называются, так как на них может разместиться информация с любого подшага текущего шага

1.3.30) DWNT\_Make\_Sub\_Name.tex — шаблон для отцентрованного по вертикали вывода имени подшага при определённом количестве вмещающихся на строке символов и при учёте максимально возможного количества строк, на которых может поместиться выводимое имя подшага;

1.3.31) HeaderDWNT.tex — шаблон для формирования одной строки заголовка отчёта (см. подробнее подраздел 1.5).

## 1.4. Группа файлов для запуска шаблонизатора через bat-файл:

1.4.1) «C:\avs\Racks\DWNT\docs\Reports\BuildTemplateMCMulti.bat» — пакетный файл для запуска построения отчёта через командную строку с возможностью построения нескольких отчётов в одном PDF-файле;

1.4.2) «C:\avs\Utilities\Template\_generator\For DWNT From XML\enter\_parameters.py» — файл, обрабатывающий ввод пользователя при запуске пакетного файла 1.4.1) и сохраняющий информацию о вводе в автоматически генерируемый файл 2.3.2).

## 1.5. Подробное описание настройки файла конфигурации

В файле 1.2.4), в тэге Report, можно прописать имя теста (атрибут UUT\_Name) и тип заголовка (атрибут Headers), считываемый с конфигурационного файла по умолчанию (на рис. 1 эти атрибуты указаны в чёрном прямоугольнике).

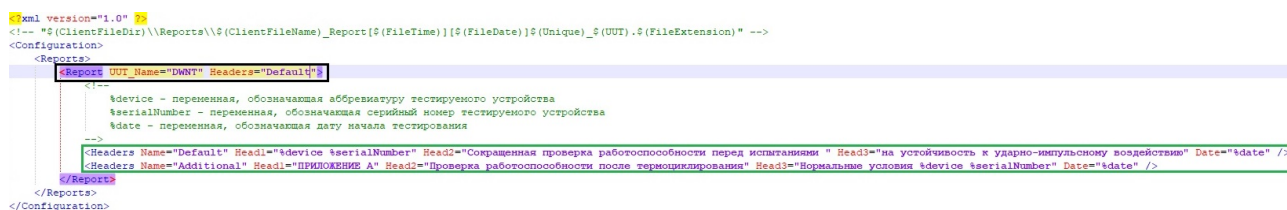


Рис. 1 Файл конфигурации

Зелёным цветом на рис. 1 выделены имеющиеся в файле виды заголовков. Название вида заголовка указывается в атрибуте Name. Важно знать, что при формировании отчёта в заголовке используются только четыре строки, не больше и не меньше. Такое константное количество используемых в заголовке строк объясняется необходимостью подсчёта каждой строки при формировании конечного PDF-файла. За содержание первых трёх строк отвечает атрибут Head $I$ , где  $I$  — номер строки заголовка от 1 до 3. Содержание четвёртой строки указывается в атрибуте Date. Этот атрибут назван так по причине частого размещения даты и времени начала испытания на четвёртой строке заголовка.

При формировании заголовка часто используется информация, которая считывается с XML-файла. Для указания на такую информацию в заголовке отчёта были созданы переменные, отмеченные знаком "%" в начале их названия. Обозначения каждой переменной представлены ниже:

- %device — переменная, обозначающая аббревиатуру тестируемого устройства;
- %serialNumber — переменная, обозначающая серийный номер тестируемого устройства;
- %date — переменная, обозначающая дату начала тестирования.

Обработку переменных, встреченных в заголовке, осуществляет файл 1.3.31).

## 2. Краткое описание выходных файлов

### 2.1. Общее описание

2.1.1) Все файлы, описанные в подразделе 2.2, размещаются во время создания отчёта в папке «C:\avs\Racks\DWNT\docs\Reports». В дополнение к этим файлам при запуске шаблонизатора через 1.4.1) генерируются файлы из раздела 2.3.

### 2.2. Генерируемые файлы при запуске через TeXstudio:

2.2.1) DWNT\_Report.pdf — файл основного отчёта в результате компиляции файла 1.3.1);

2.2.2) DWNT\_Report.gen.tex — файл TeXstudio, заполненный с помощью средства Jinja2 данными из считываемого XML-файла при компиляции 1.3.1);

2.2.3) DWNT\_Report.log — файл журнала построения 2.2.1) из 2.2.2) с описанием всех предупреждений и ошибок, возникших при этом процессе;

2.2.4) DWNT\_Report.aux — файл, генерируемый при построении 2.2.1) из 2.2.2), хранящий информацию о всех сносках, перекрёстных ссылках и т.п. для их корректного расположения в генерируемом PDF-файле.

### 2.3. Генерируемые файлы при запуске через bat-файл:

2.3.1) DWNT\_*serialN*\_*date*\_*time*\_*num*.pdf — файл основного отчёта в результате компиляции файла 1.3.1) через bat-файл, где *serialN* — серийный номер протестированного оборудования (при его отсутствии пишется EmptySerialNo), *date* — дата начала тестирования в формате ГГГГ-ММ-ДД (ГГГГ — год из четырёх цифр, ММ — месяц из двух цифр, ДД — день из двух цифр, при отсутствии указывается EmptyStartDate), *time* — время начала тестирования в формате ЧЧ-ММ-СС (ЧЧ — часы из двух цифр, ММ — минуты из двух цифр, СС — секунды из двух цифр, при отсутствии указывается EmptyStartTime), *num* — порядковый номер отчёта (когда не существует файлов с одинаковыми параметрами (серийный номер, дата, время), через bat-файл генерируется файл с названием, например, DWNT\_765765\_2020-05-20\_12-00-00, а при повторной компиляции сформируется PDF-файл с именем DWNT\_765765\_2020-05-20\_12-00-00\_1, при третьей компиляции сформируется DWNT\_765765\_2020-05-20\_12-00-00\_2 и т.д.) Файл создаётся в той же папке, где располагается выбранный пользователем при запуске bat-файла отчёт TestStand в формате XML;

2.3.2) parameters.txt — файл, генерируемый в директории «C:\avs\Utilities\Template\_generator\For DWNT From XML» и служащий для передачи данных между Python-скриптами 1.4.2), 1.2.1) и bat-файлом 1.4.1);

2.3.3) dropout.txt — файл для передачи серийного номера от Python-скрипта 1.2.1) к bat-файлу 1.4.1).

## 3. Подготовительные работы

### 3.1. Подготовительная работа с файлами

3.1.1) Если папка «C:\avs\Utilities\Template\_generator\For DWNT From XML» не существует, создать её и поместить в неё все файлы группы 1.2. Если папки «C:\avs\Racks\DWNT\docs\Reports» нет, создать её и поместить в неё все файлы из группы 1.3.

3.1.2) Если планируется запуск шаблонизатора через bat-файл, необходимо проверить, что пути к файлам 1.4.1) и 1.4.2) совпадают с их действительным значением, иначе их нужно перенести в соответствующие папки.

## 4. Компиляция файлов в TeXstudio

### 4.1. Настройка программы TeXstudio

4.1.1) Запустить файл 1.3.1).

4.1.2) В главном окне TeXstudio перейти на вкладку "Options" — "Configure TeXstudio..." (рис. 2).

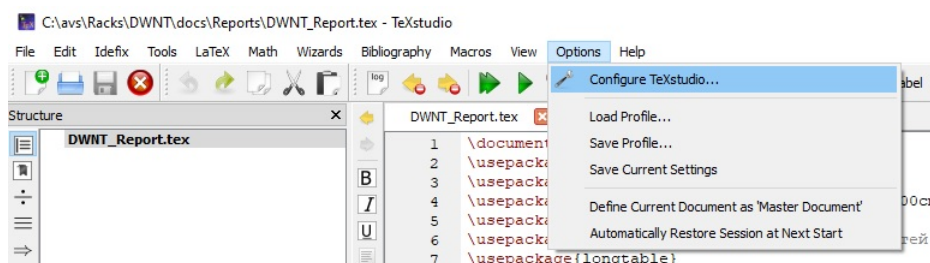


Рис. 2 Главное окно программы TeXstudio

4.1.3) В открывшемся окне перейти на вкладку "Build" и активировать расширенные настройки, установив соответствующий флажок, обведённый красным прямоугольником на рис. 3.

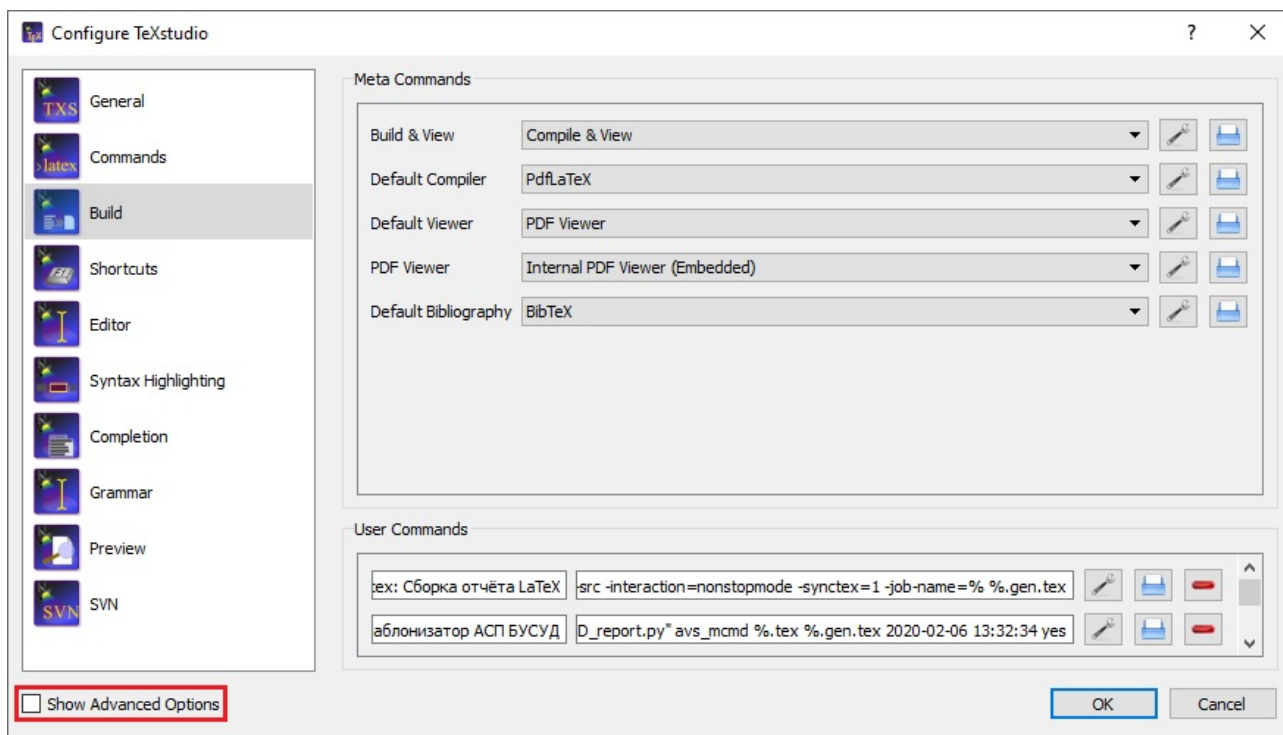


Рис. 3 Включение расширенных настроек компилятора TeXstudio

4.1.4) После этого текущее окно расширится, как показано на рис. 4. В его подокне "User Commands" необходимо найти кнопку добавления пользовательской команды "Add", выделенную на рис. 4 оранжевым прямоугольником, и нажать на неё.

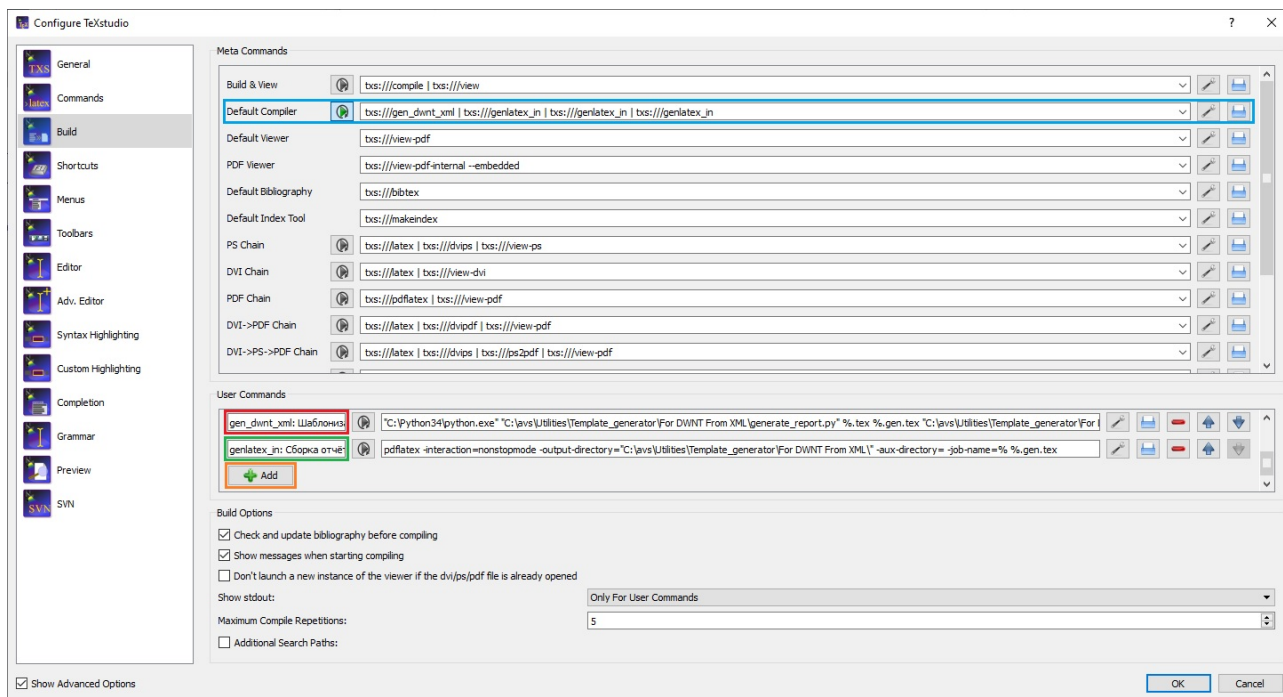


Рис. 4 Окно расширенных настроек компилятора TeXstudio

4.1.5) Для создаваемой пользовательской команды предлагается ввести название «gen\_dwnt\_xml: Шаблонизатор для АСП БИОС», удалив первоначальное название "user0", как показано на рис. 5. Двоеточие в названии отделяет само название команды и её краткое описание.



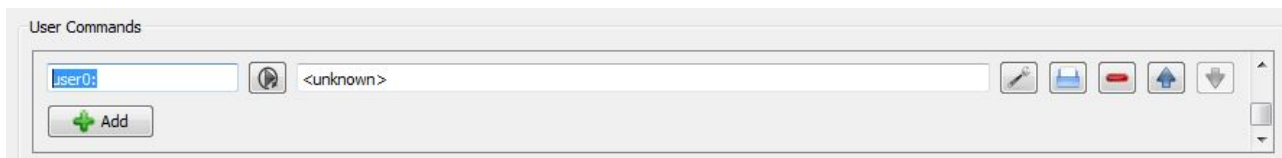


Рис. 5 Добавление новой пользовательской команды

4.1.6) Вместо слова "<unknown>" прописать команду «"*<путь к файлу-генератору>*" "*<путь к файлу-генератору>*" *%.tex %.gen.tex "<путь к xml>"*», где "*<путь к файлу-генератору>*" — полный путь к компилятору Python (обычно «C:\Python34\python.exe»), описанный в кавычках для корректной идентификации пути, содержащего пробельные символы; "*<путь к файлу-генератору>*" — полный путь к сценарию 1.2.1) в кавычках (обычно «C:\avs\Utilities\Template\_generator\For DWNT From XML\generate\_report.py»); *%.tex* — название входного файла с пустым шаблоном для сценария 1.2.1) (знак "%" при компиляции всегда заменяется на название текущего открытого файла в TeXstudio); *%.gen.tex* — имя выходного файла с уже заполненными данными из считываемого XML-файла с помощью сценария 1.2.1); "*<путь к xml>*" — абсолютный путь к XML-отчёту TestStand в кавычках, с которого должны считываться данные при заполнении шаблона. Данная команда будет осуществлять считывание сведений с указанного XML-файла и формировать PDF-файл с заголовком, соответствующим атрибуту "Headers" в тэге "Report" файла 1.2.4). Для указания какого-то определённого заголовка, описанного в этом конфигурационном файле, необходимо после пути к считываемому XML-файлу указать название заголовка из атрибута "Name" тэга необходимого заголовка. Пример команды, которая должна считывать данные из XML-отчёта TestStand о проведённом испытании по пути «C:\avs\Utilities\Template\_generator\For DWNT From XML\Main\_20GK-01\_Report [09 45 12] [03.12.2019]\_62097201.xml"» и формировать заголовок "Additional", показан на рис. 6. Её текст имеет такой вид: «C:\Python34\python.exe" "C:\avs\Utilities\Template\_generator\For DWNT From XML\generate\_report.py" *%.tex %.gen.tex* "C:\avs\Utilities\Template\_generator\For DWNT From XML\Main\_20GK-01\_Report [09 45 12] [03.12.2019]\_62097201.xml" Additional».

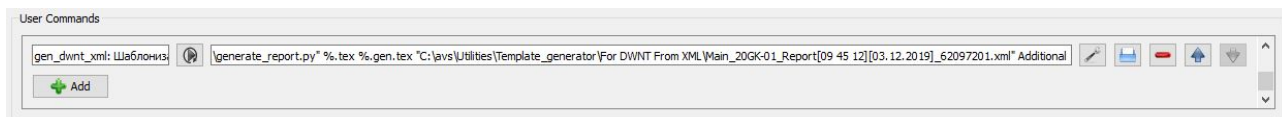


Рис. 6 Пользовательская команда с указанием вида заголовка

4.1.7) Шаблонизатор также поддерживает создание одного PDF-отчёта по нескольким испытаниям. При этом отчёт по каждому испытанию начинается с новой нумерации страниц. Для создания такого отчёта после ввода в пользовательскую команду на рис. 6 аргумента *%.gen.tex* нужно придерживаться следующей схемы размещения последующих аргументов: «"*<путь к файлу xml 1>*" [тип заголовка 1] "*<путь к файлу xml 2>*" [тип заголовка 2] ... "*<путь к файлу xml n>*" [тип заголовка n]», где *n* — количество испытаний, по результатам которых формируется отчёт; "*<путь к файлу xml i>*" — путь к *i*-тому XML-файлу в кавычках при *i* = 1, 2, ..., *n*; *тип заголовка i* — тип *i*-того заголовка для *i*-того испытания (является необязательным аргументом, на что указывают квадратные скобки). Примером команды, формирующей отчёт по двум XML-файлам, является команда «C:\Python34\python.exe" "C:\avs\Utilities\Template\_generator\For DWNT From XML\generate\_report.py" *%.tex %.gen.tex* "C:\avs\Utilities\Template\_generator\For DWNT From XML\Main\_20GK-01\_Report [09 45 12] [03.12.2019]\_62097201.xml" "C:\avs\Utilities\Template\_generator\For DWNT From XML\Main\_20GK-01-Re-

port[10 12 78][05.12.2019]\_62097203.xml" Additional». Данная команда создаёт один PDF-файл, в котором под заголовком, определяемым конфигурационным файлом, сначала будет размещена информация по первому XML-файлу «Main\_20GK-01\_Report[09 45 12][03.12. 2019]\_62097201.xml», а затем будет размещена информация по второму XML-файлу «Main\_20GK-01\_Report[10 12 78][05.12. 2019]\_62097203.xml» под заголовком "Additional".

4.1.8) Прописать ещё одну пользовательскую команду, как в пунктах 4.1.5)–4.1.6). Однако теперь в названии команды вместо строки по-умолчанию "user0" предлагается вписать «genlatex\_in: Сборка отчёта в определённую папку», а вместо "<unknown>" вписать «pdflatex -interaction=nonstopmode -output-directory="<путь к папке с pdf>" -aux-directory= -job-name=% %.gen.tex», где *pdflatex* — это приложение по преобразованию файла шаблона, содержащего команды LaTeX, в PDF-файл; *-interaction=nonstopmode* — флаг приложения *pdflatex*, указывающий, что выполнение компиляции происходит без остановки даже при возникновении ошибок; *-output-directory=* — флаг, указывающий папку назначения для сохранения создаваемого PDF-файла (по умолчанию PDF-файл сохраняется в той же папке, в которой находится преобразуемый в PDF-файл шаблон); "<путь к папке с pdf>" — заключённый в кавычки путь к папке назначения, в которой должен храниться создаваемый PDF-файл; *-aux-directory=* — флаг, указывающий папку назначения для хранения файлов с расширением .aux (см. подробнее 2.2.4)), при этом указанная пустая папка говорит о создании файла .aux в той же папке, где находится преобразуемый в PDF шаблон (по умолчанию файл .aux создаётся в папке вместе с генерируемым PDF-файлом); *-job-name=%* — флаг, указывающий имя генерируемого PDF-файла (знак "%" заменяется названием текущего открытого в TeXstudio документа); *%.gen.tex* — преобразуемый в PDF-файл шаблон. В итоге данные должны быть заполнены почти так же, как показано рис. 7. Отличными могут быть только последняя часть первой команды, следующая за словом "%.gen.tex" и заключённая в кавычки, так как эта часть команды уже отвечает за указание пути к считываемому XML-файлу, и во второй команде значение флага *-output-directory=*, указывающего на папку, в которую пользователь хочет сохранить создаваемый PDF-документ.

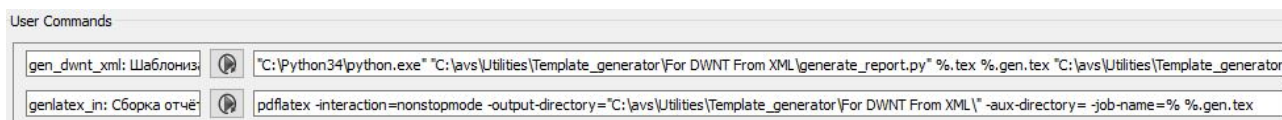


Рис. 7 Вид заполненных пользовательских команд

4.1.9) В окне "Meta Commands", в строке "Default Compiler" (выделена бирюзовым прямоугольником на рис. 4), описать порядок выполнения команд при компиляции файла LaTeX, то есть написать «txs:///gen\_dwnt\_xml | txs:///gen-latex\_in | txs:///genlatex\_in | txs:///genlatex\_in». Это строка означает, что будет сначала один раз заполняться шаблон отчёта с помощью сценария 1.2.1), а затем за три прохода будет создаваться PDF-файл. Тройная компиляция необходима, чтобы LaTeX смог настроить все перекрёстные ссылки и сумел построить все графики. Чтобы было возможным неоднократное выполнение команд компиляции, необходимо нажать левой кнопкой мыши на стрелочку рядом с полем описания команд (нужная кнопка обозначена на рис. 8 красной стрелкой). После этого нужно ещё раз нажать левой кнопкой мыши на любое свободное место окна "Meta Commands". После этого данная кнопка должна окраситься в зелёный цвет, как показано на рис. 9. Повторное нажатие на эту кнопку приведёт к запрету на многократное использование команд, а кнопка потеряет зелёный окрас и будет выглядеть как на рис. 8.



Рис. 8 Включение повторяемости команд компиляции



Рис. 9 Активированная кнопка многократной компиляции

4.1.10) Закрывать окно "Configure TeXstudio", полностью изображённое на рис. 4.

## 4.2. Работа с компилируемым файлом

4.2.1) Открыть в TeXstudio файл 1.3.1). Если в файле были сделаны несохранённые изменения, на вкладке, отображающей открытый файл, появится значок, обведённый в зелёный круг на рис. 10. Чтобы сохранить изменения в файле TeXstudio, нужно либо нажать сочетание клавиш "Ctrl + s", либо левой кнопкой мыши щёлкнуть по кнопке в среде TeXstudio, выделенной красной стрелкой на рис. 10.

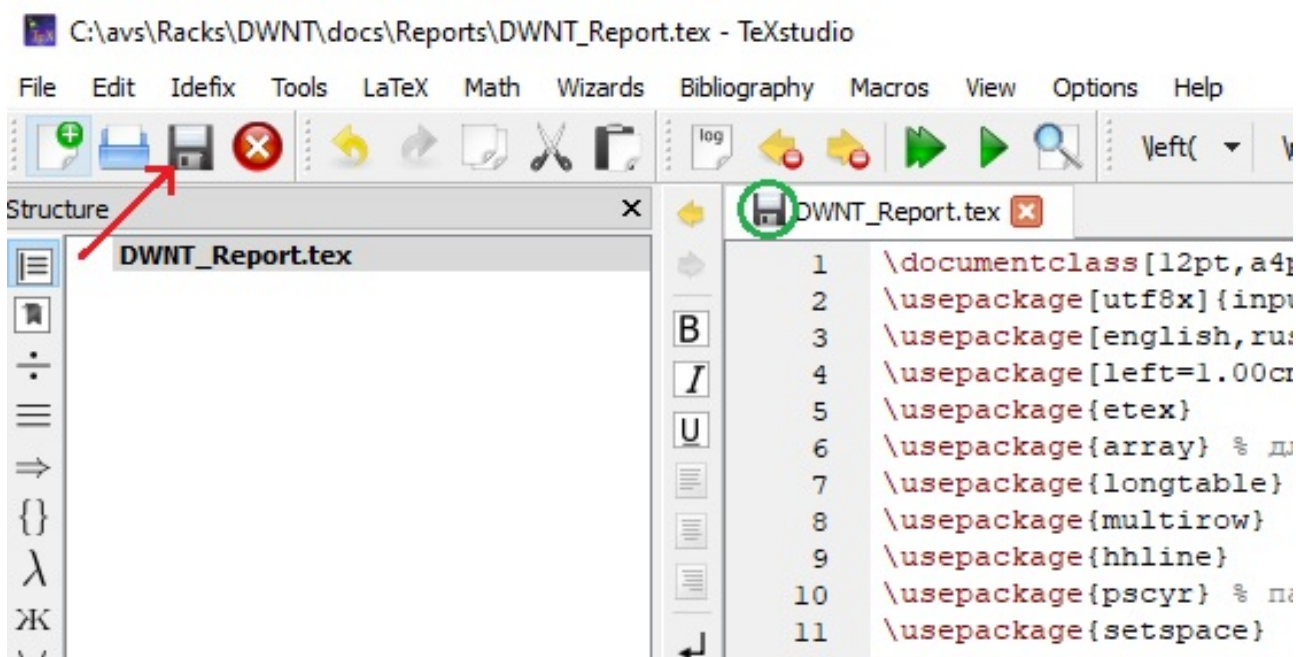


Рис. 10 Сохранение файла

4.2.2) Убедиться, что в программе TeXstudio открыт файл, на основе которого будет сгенерирован PDF-файл (для построения PDF-отчёта используется файл 1.3.1)), и запустить его компиляцию, нажав клавишу "F1" или одну из двух кнопок, показанных красной и синей стрелкой на рис. 11. Нажатие на кнопку, отмеченную красной стрелкой, не только начнёт генерацию нужного вам отчёта, но и, если создаваемый PDF-файл находится в той же папке, что и его шаблон, покажет в правой части среды TeXstudio результирующий PDF-файл, как показано на рис. 12. Нажатие на кнопку, отмеченную синей стрелкой, также начнёт генерацию отчёта, но не выведет её результат в среде TeXstudio. Сгенерированный отчёт в формате PDF будет создан в той папке, которую указал пользователь во флаге *-output-directory* внутри пользовательской команды «genlatex\_in» (см. подробнее в пункте 4.1.8)).

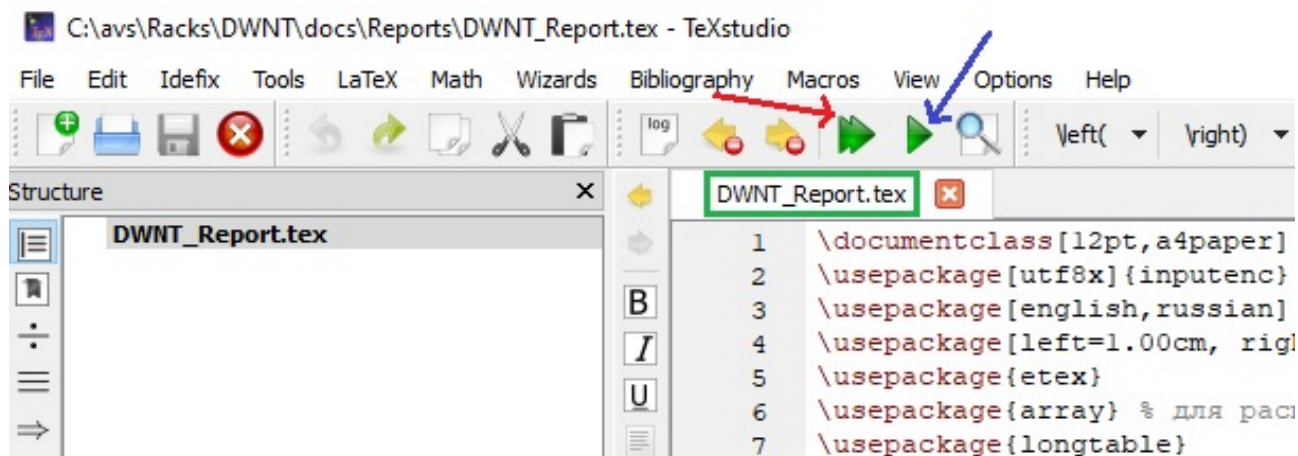


Рис. 11 Режимы компиляции

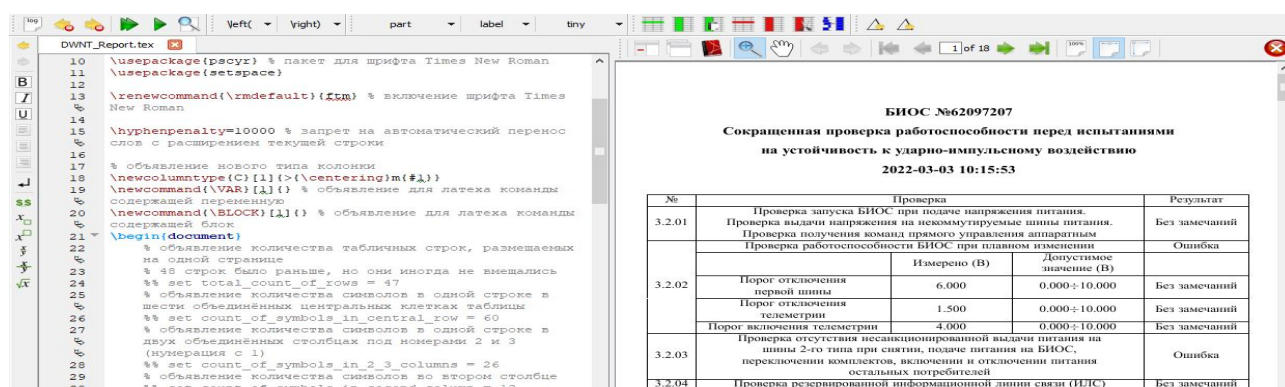


Рис. 12 Вывод результатов компиляции в среде TeXstudio

### 4.3. Краткое описание результатов компиляции

4.3.1) При успешном завершении считывания всех XML-файлов и заполнения шаблона 1.3.1) в среде разработки TeXstudio, в верхней части окна "Messages", появится сообщение, показанное на рис. 13. Часть сообщения, выделенная на этом рисунке в бирюзовый прямоугольник, раскрывает запущенную TeXstudio пользовательскую команду «gen\_dwnt\_xml». Строки, выделенные красным прямоугольником, выводит сценарий Python 1.2.1) в самом конце своей работы (такой вывод в самом Python-файле осуществляется с помощью функции «print()»). Строка, выделенная в оранжевый прямоугольник, говорит о том, что при выполнении текущей команды (команды «gen\_dwnt\_xml») не возникло ошибок, а строка в чёрном прямоугольнике раскрывает содержание пользовательской команды «genlatex\_in». Как видно из рис. 13 все знаки "%" из пользовательских команд превратились в название компилируемого в TeXstudio файла.



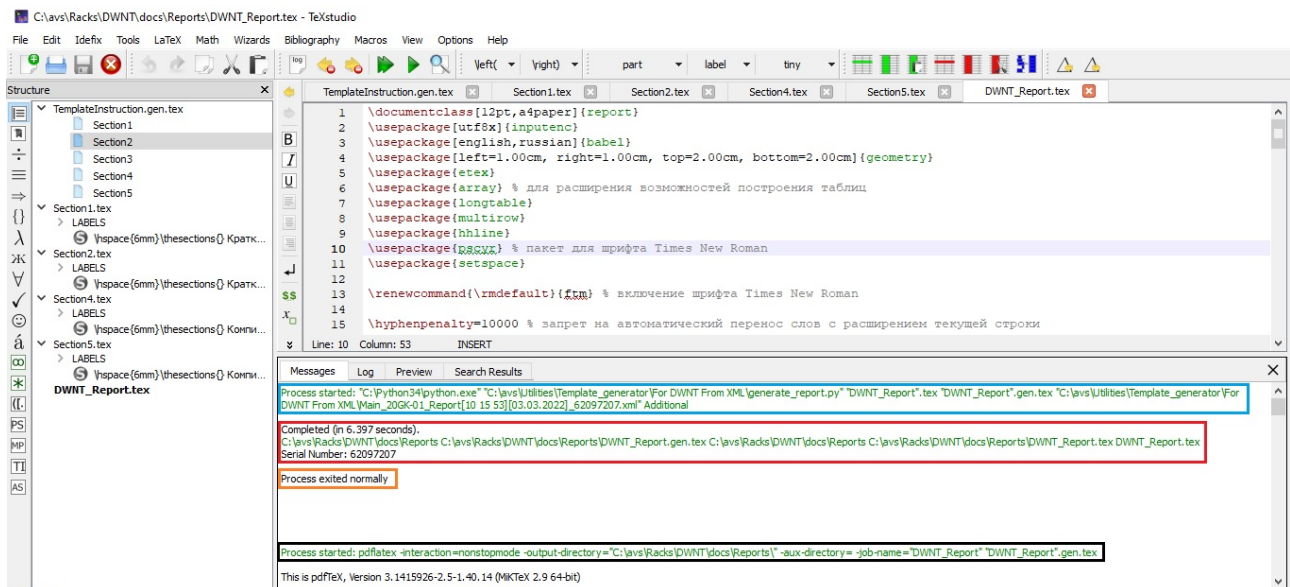


Рис. 13 Успешное завершение заполнения шаблона

4.3.2) Исходя из предыдущего пункта и рисунка, можно сделать вывод, что все ошибки, возникшие при выполнении команды «gen\_dwnt\_xml» в окне "Messages" среды разработки TeXstudio располагаются между строками в бирюзовом прямоугольнике и строкой в чёрном прямоугольнике, то есть в этом пространстве отображаются все ошибки, возникшие во время считывания XML-файлов и заполнения полученными данными шаблона 1.3.1).

4.3.3) Для подтверждения этого намеренно допускается ошибка в файле 1.2.3), в функции «get\_child\_elements\_by\_tag\_name»: вместо встроенной в Python функции «isinstance», сверяющей, соответствует ли тип первого аргумента этой функции типу, указанному во втором аргументе, используется неизвестная для Python функция «isintance». Для наглядности на рис. 14 введённая ошибка подчёркнута красной линией.

```

54 def get_child_elements_by_tag_name(self, parent_node, tag_name):
55     """
56     Возвращает список объектов, удовлетворяющих заданному имени тега
57     :param parent_node: родительский объект, дочерние объекты которого ищутся
58     :param tag_name: имя тега, по которому ищут дочерние объекты (строка)
59     :return: список дочерних элементов, подходящих под данное имя тега
60     """
61     child_nodes = []
62     if isintance(tag_name, str) and tag_name != '' and not(parent_node is None) and parent_node.nodeType == xml.dom.Node.ELEMENT_NODE:
63         all_child_nodes = parent_node.childNodes
64         for child_node in all_child_nodes:
65             if child_node.nodeType == xml.dom.Node.ELEMENT_NODE and child_node.tagName == tag_name:
66                 child_nodes.append(child_node)
67     return child_nodes

```

Рис. 14 Намеренная ошибка в файле for\_read\_xml.py

4.3.4) Затем осуществляется компиляция файла 1.3.1), как описано в пункте 4.2.2). При этом в среде разработки TeXstudio в самом верху окна "Messages" возникает сообщение, показанное на рис. 15 и указывающее на обнаруженную ошибку.

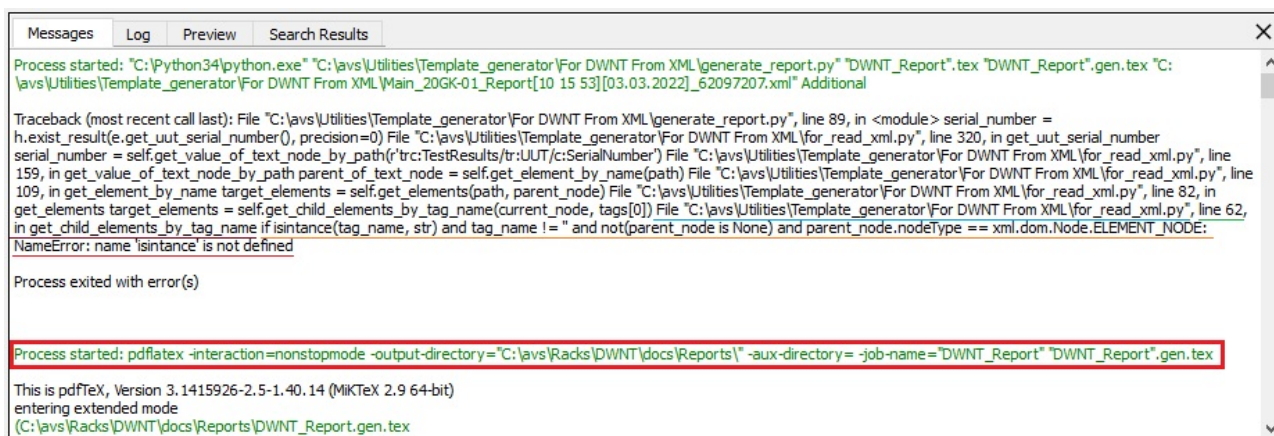


Рис. 15 Обнаруженная ошибка в файле for\_read\_xml.py

4.3.5) При разборе сообщения об ошибке можно заметить, что тип ошибки и её описание указаны только в самом конце описания ошибки (на рис. 15 тип ошибки подчеркнут красной линией). Далее в порядке справа налево отображаются: строка, в которой была найдена ошибка (подчеркнута оранжевой линией); название функции, в которой была найдена ошибка (подчеркнуто тёмно-красной линией), и специальное слово "in"; указание номера строки в файле с обнаруженной ошибкой (подчеркнуто зелёной линией); абсолютный путь к файлу, в котором была обнаружена ошибка (подчеркнут бирюзовой линией), и специальное слово "File". Как можно убедиться из пункта 4.3.3), описание обнаруженной ошибки полностью соответствует намеренно сделанной ошибке. Остальной вывод, касающийся сделанной в файле Python ошибки, в большинстве случаев можно не рассматривать, так как он лишь указывает стек вызовов функций, то есть при каком из вызовов функция «get\_child\_elements\_by\_tag\_name» вызвала ошибку.

4.3.6) Также стоит отметить, что, как видно из рис. 15, несмотря на возникшую ошибку, компилятор TeXstudio всё равно запустил выполнение команды «genlatex\_in» (информация об этом указана в строке, выделенной в красный прямоугольник на рис. 15), причём её выполнение завершилось даже без ошибок, как видно на рис. 16. Такое поведение компилятора объясняется тем, что, во-первых, выполнение пользовательских команд TeXstudio не прерывается и, во-вторых, преобразуемый в PDF-отчёт файл 2.2.2) с предыдущей компиляции уже существовал в рабочем каталоге, то есть команда «genlatex\_in» была выполнена по отношению к предыдущей версии файла 2.2.2).

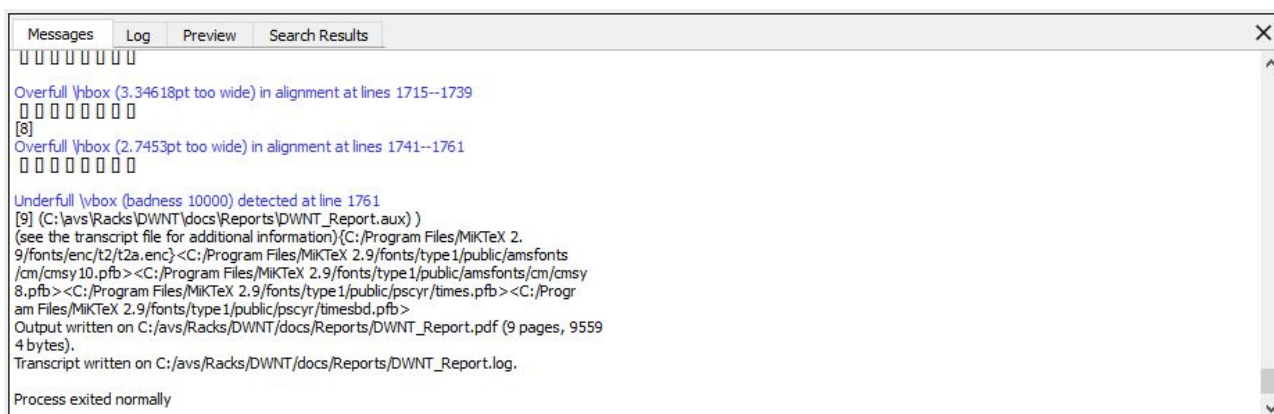


Рис. 16 Успешное завершение преобразования TeX-файла в PDF

4.3.7) Также отдельно следует упомянуть ошибки, которые могут возникнуть в самих

файлах шаблона. Это могут ошибки как средства заполнения шаблона Jinja2, так и ошибки компилятора LaTeX.

4.3.8) Для моделирования ошибки средства Jinja2 в файле 1.3.26) удаляется специальная команда "set" в строке, выделенной красной рамкой на рис. 17.

```

1  %% set min_len = splits[0]
2  % при подсчёте строк, занимаемых именем всегда ведётся подсчёт других строк шага с самого начала
3  %% set item = 0
4  %% set is_counter_for_name = True
5  %% include('DWNT_Counter_Of_Rows.tex')
6  %% set count_of_rows_for_step_number = splits[0] + ns.count_of_share_rows
7  %% for item in range(0, splits[0])
8      %% if item == 0
9          \multirow{\VAR{count_of_rows_for_step_number}}{*}{\VAR{full_number}} &
10         \multicolumn{6}{C{\VAR{w2_7}}}{\VAR{splits[1][item]}} &
11         \multirow{\VAR{splits[0]}}{*}{\VAR{step_status}} \tabularnewline
12     %% else
13     & \multicolumn{6}{C{\VAR{w2_7}}}{\VAR{splits[1][item]}} & \tabularnewline
14     %% endif
15     %% if item == splits[0] - 1
16     % если после имени шага больше ничего не помещается на странице
17     %% if ns.count_of_share_rows == 0
18     \hline
19     % если после имени шага на странице помещаются другие строки
20     %% else
21     \hhline{~|-----|~}
22     %% endif
23     %% endif
24     %% endfor

```

Рис. 17 Намеренная ошибка в шаблоне Jinja2

4.3.9) После этого осуществляется компиляция файла 1.3.1), как описано в пункте 4.2.2). При этом в среде разработки TeXstudio в самом верху окна "Messages" возникает сообщение, показанное на рис. 18 и указывающее на обнаруженную ошибку.

Process started: "C:\Python34\python.exe" "C:\javs\Utilities\Template\_generator\FOR DWNT From XML\generate\_report.py" "DWNT\_Report".tex "DWNT\_Report".gen.tex "C:\javs\Utilities\Template\_generator\FOR DWNT From XML\Main\_20GK-01\_Report[10 15 53][03.03.2022\_62097207.xml" Additional

Traceback (most recent call last): File "C:\javs\Utilities\Template\_generator\FOR DWNT From XML\generate\_report.py", line 140, in <module> output.write(template.render()) File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\environment.py", line 1008, in render return self.environment.handle\_exception(exc\_info, True) File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\environment.py", line 780, in handle\_exception reraise(exc\_type, exc\_value, tb) File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\compat.py", line 37, in reraise raise value.with\_traceback(tb) File "C:\javs\Racks\DWNT\docs\Reports\TestDWNTWriteNameStep.tex", line 4, in template % is\_counter\_for\_name = True File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\environment.py", line 1005, in render return concat(self.root\_render\_func(self.new\_context(vars))) File "C:\javs\Racks\DWNT\docs\Reports\DWNT\_Report.tex", line 97, in top-level template code %% include('TestDWNT' + number + '.tex') File "C:\javs\Racks\DWNT\docs\Reports\TestDWNT1.tex", line 5, in top-level template code %% include('TestDWNTWriteNameStep.tex') File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\environment.py", line 780, in handle\_exception reraise(exc\_type, exc\_value, tb) File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\compat.py", line 37, in reraise raise value.with\_traceback(tb) File "C:\javs\Racks\DWNT\docs\Reports\TestDWNTWriteNameStep.tex", line 4, in template % is\_counter\_for\_name = True File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\environment.py", line 497, in \_parse return Parser(self, source, name, encode\_filename(filename)).parse() File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\parser.py", line 901, in parse result = nodes.Template(self.\_subparse(), lineno=1) File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\parser.py", line 144, in \_subparse self.fail\_unknown\_tag(token.value, token.lineno) File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\parser.py", line 97, in fail\_unknown\_tag return self.\_fail\_ut\_eof(name, self.\_end\_token\_stack, lineno) File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\parser.py", line 90, in \_fail\_ut\_eof self.fail(''.join(message), lineno) File "C:\Python34\lib\site-packages\jinja2-2.10.1-py3.4.egg\jinja2\parser.py", line 59, in fail raise exc(msg, lineno, self.name, self.filename) jinja2.exceptions.TemplateSyntaxError: Encountered unknown tag 'is\_counter\_for\_name'.

Process exited with error(s)

Process started: pdflatex -interaction=nonstopmode -output-directory="C:\javs\Racks\DWNT\docs\Reports\" -aux-directory= -job-name="DWNT\_Report" "DWNT\_Report".gen.tex

This is pdfTeX, Version 3.1415926-2.5-1.40.14 (MKTeX 2.9 64-bit)  
entering extended mode

Рис. 18 Обнаружение ошибки в Jinja2

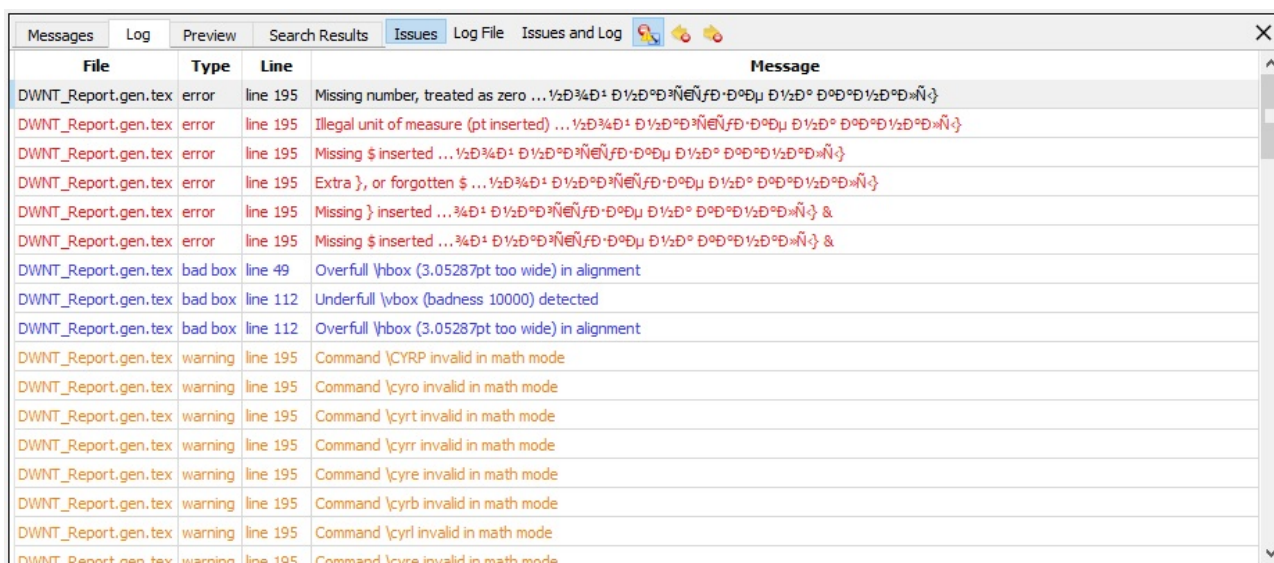
4.3.10) В данном случае описание ошибки уже немного сложнее найти, чем это было в пункте 4.3.5), так как при возникновении ошибок в шаблонизаторе Jinja2 сама ошибка и её тип указываются в конце сообщения об ошибке (тип и сама ошибка подчёркнуты красной линией). Остальные части ошибки нужно искать в середине рассматриваемого абзаца. Для упрощения поиска можно отметить, что искать надо один из файлов, создаваемых



самим программистом, остальные описываемые в сообщении файлы являются только посредниками между программными файлами самого разработчика. В описываемом случае путь к файлу с ошибкой обозначается вместе со словом "File" и подчёркнут на рис. 18 бирюзовой линией, номер строки кода, в которой произошла ошибка, подчёркнут зелёной линией, описание кода строки с ошибкой и специальные слова "in template", указывающие появление ошибки именно в файле шаблона, подчёркнуты оранжевой линией.

4.3.11) Также для наглядности поиска ошибок можно отметить, что сообщения об ошибках в файлах Python (рис. 15) описываются обычно текстом чёрного цвета, а сообщения об ошибках средства заполнения шаблонов Jinja2 в TeX-файлах (рис. 18) описываются текстом зелёного цвета.

4.3.12) Самыми сложными для выявления являются ошибки компилятора LaTeX. Во-первых, LaTeX компилирует файл 2.2.2), который включает в себя содержание всех файлов группы 1.3, поэтому номер строки с возникшей ошибкой, указанный при её обнаружении, нужно искать не в файлах группы 1.3, а в уже заполненном данными из XML-файла шаблоне 2.2.2). Во-вторых, компилятор LaTeX не поддерживает кириллицу, поэтому описание строк с возникшими ошибками иногда дополняется непонятными для разработчика символами. Также следует отметить, что возникшие при такой компиляции ошибки отображаются обычно не в окне "Messages", как это происходит с другими видами ошибок, а в окне "Log". Например, на рис. 19 отображается сообщение об ошибке, возникшей в результате того, что в файле 1.3.9), в строке 140, в качестве значения ширины пользовательского типа столбца таблицы "C" была передана строка «w2\_7», которая не может указывать на величину какой-либо размерности (на рис. 20 ошибочный блок кода выделен чёрной рамкой).



File	Type	Line	Message
DWNT_Report.gen.tex	error	line 195	Missing number, treated as zero ... ½D¾D¹ D½D°D³ÑEÑfD·D°Dµ D½D° D°D°D½D°D»Ñ·}
DWNT_Report.gen.tex	error	line 195	Illegal unit of measure (pt inserted) ... ½D¾D¹ D½D°D³ÑEÑfD·D°Dµ D½D° D°D°D½D°D»Ñ·}
DWNT_Report.gen.tex	error	line 195	Missing \$ inserted ... ½D¾D¹ D½D°D³ÑEÑfD·D°Dµ D½D° D°D°D½D°D»Ñ·}
DWNT_Report.gen.tex	error	line 195	Extra }, or forgotten \$ ... ½D¾D¹ D½D°D³ÑEÑfD·D°Dµ D½D° D°D°D½D°D»Ñ·}
DWNT_Report.gen.tex	error	line 195	Missing } inserted ...¾D¹ D½D°D³ÑEÑfD·D°Dµ D½D° D°D°D½D°D»Ñ·} &
DWNT_Report.gen.tex	error	line 195	Missing \$ inserted ...¾D¹ D½D°D³ÑEÑfD·D°Dµ D½D° D°D°D½D°D»Ñ·} &
DWNT_Report.gen.tex	bad box	line 49	Overfull \hbox (3.05287pt too wide) in alignment
DWNT_Report.gen.tex	bad box	line 112	Underfull \vbox (badness 10000) detected
DWNT_Report.gen.tex	bad box	line 112	Overfull \hbox (3.05287pt too wide) in alignment
DWNT_Report.gen.tex	warning	line 195	Command \CYRP invalid in math mode
DWNT_Report.gen.tex	warning	line 195	Command \cyro invalid in math mode
DWNT_Report.gen.tex	warning	line 195	Command \cyrt invalid in math mode
DWNT_Report.gen.tex	warning	line 195	Command \cyrr invalid in math mode
DWNT_Report.gen.tex	warning	line 195	Command \cyre invalid in math mode
DWNT_Report.gen.tex	warning	line 195	Command \cyrb invalid in math mode
DWNT_Report.gen.tex	warning	line 195	Command \cyrl invalid in math mode
DWNT_Report.gen.tex	warning	line 195	Command \cyre invalid in math mode

Рис. 19 Обнаружение ошибки в LaTeX



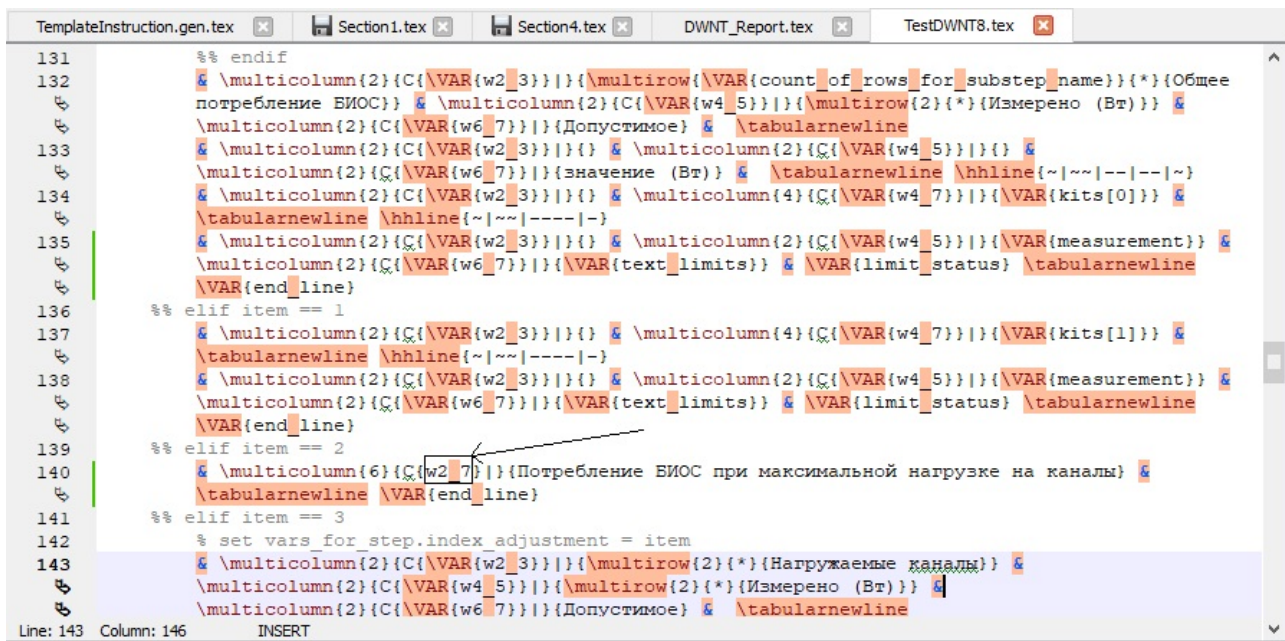


Рис. 20 Намеренная ошибка в команде LaTeX

4.3.13) Из рис. 19 можно увидеть, что одна возникшая ошибка породила сразу шесть ошибок LaTeX, что также затрудняет выявление причин возникшей ошибки. Однако у обработчика ошибок TeXstudio есть свои преимущества. Если дважды нажать левой кнопкой мыши по любой строчке, описанной в окне "Log" на рис. 19, то TeXstudio автоматически откроет файл, содержащий ошибку, и перекинет к той строке, в которой, по мнению компилятора LaTeX, содержится ошибка. Результат такого перехода показан на рис. 21.

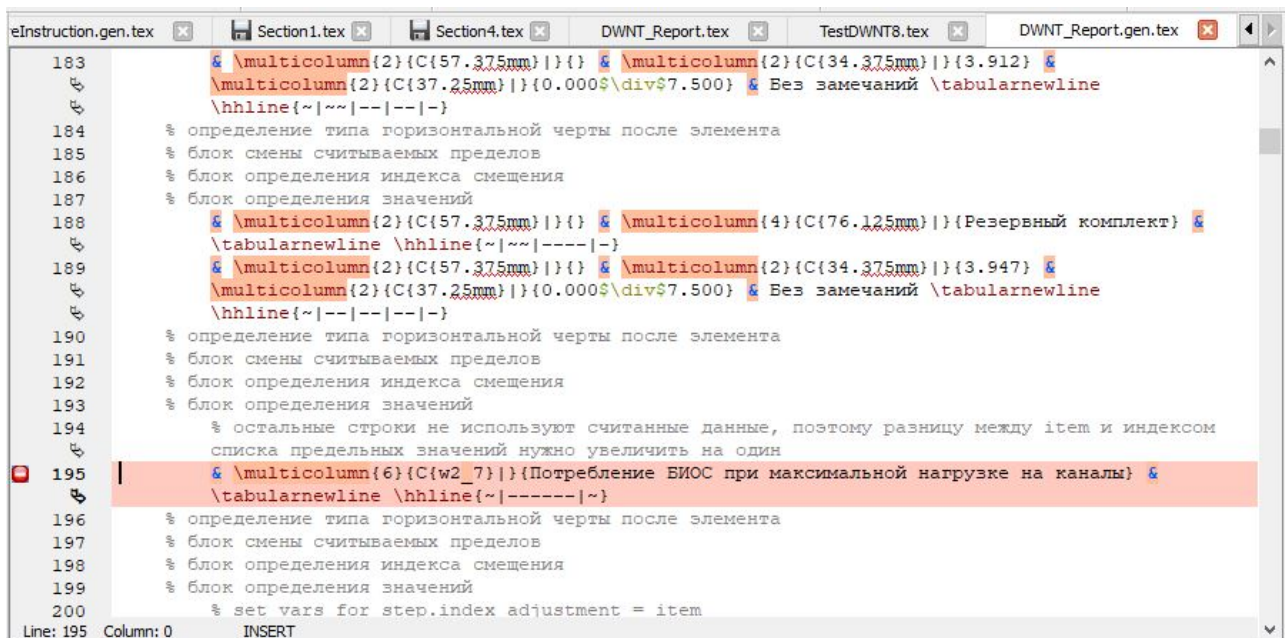


Рис. 21 Указание ошибки LaTeX при переходе к ошибочной строке

4.3.14) Также следует сказать, что в окне "Messages" среды разработки TeXstudio возникшая ошибка тоже указывается, но в менее читабельном виде (рис. 22). В этом же окне формируется и сообщение о неуспешной компиляции файла 2.2.2) (рис. 23).



Рис. 22 Указание ошибки LaTeX в окне "Messages"

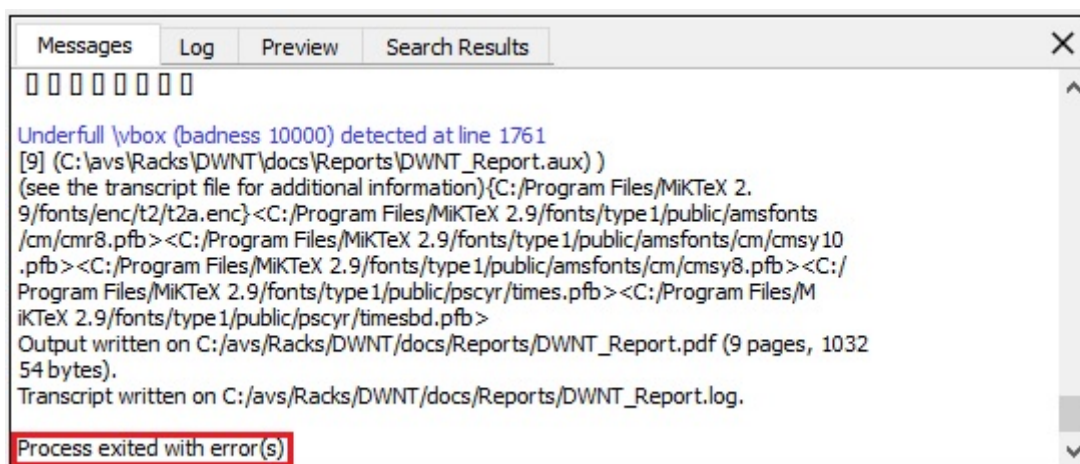


Рис. 23 Сообщение о неуспешной компиляции TeX-файла

4.3.15) Для исправления ошибки, описанной в пункте 4.3.12), необходимо в качестве аргумента пользовательского типа колонки "C" установить не строку «w2\_7», а строку «\VAR{w2\_7}», как показано в бирюзовом прямоугольнике на рис. 24. Команда «\VAR» выводит значение переменной Jinja2 в шаблон TeX-файла, то есть строка «w2\_7», переданная в качестве аргумента функции «\VAR» заменится на соответствующее ей значение, установленное средством заполнения шаблонов Jinja2 в файле 1.3.1), как показано в красном прямоугольнике на рис. 25.

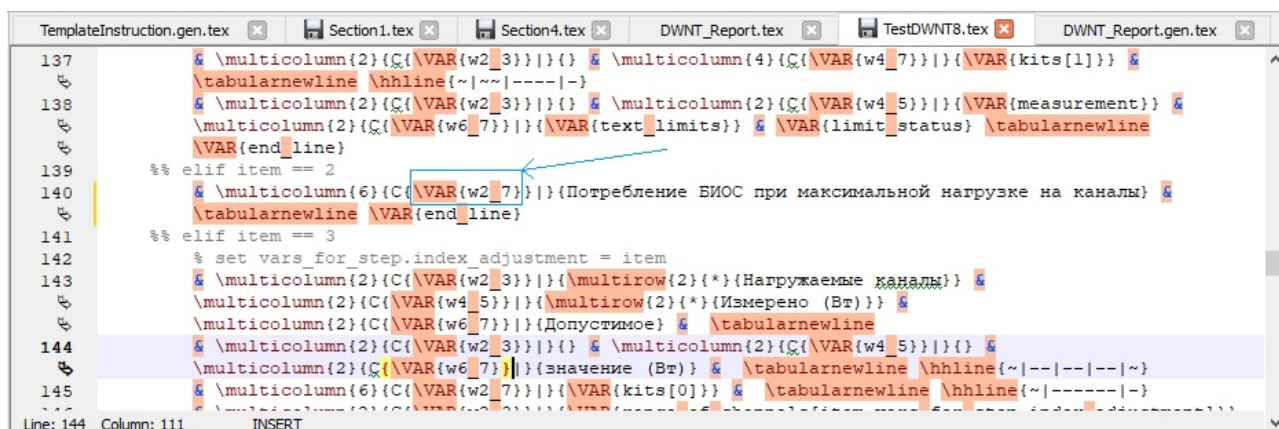


Рис. 24 Исправленная ошибка LaTeX

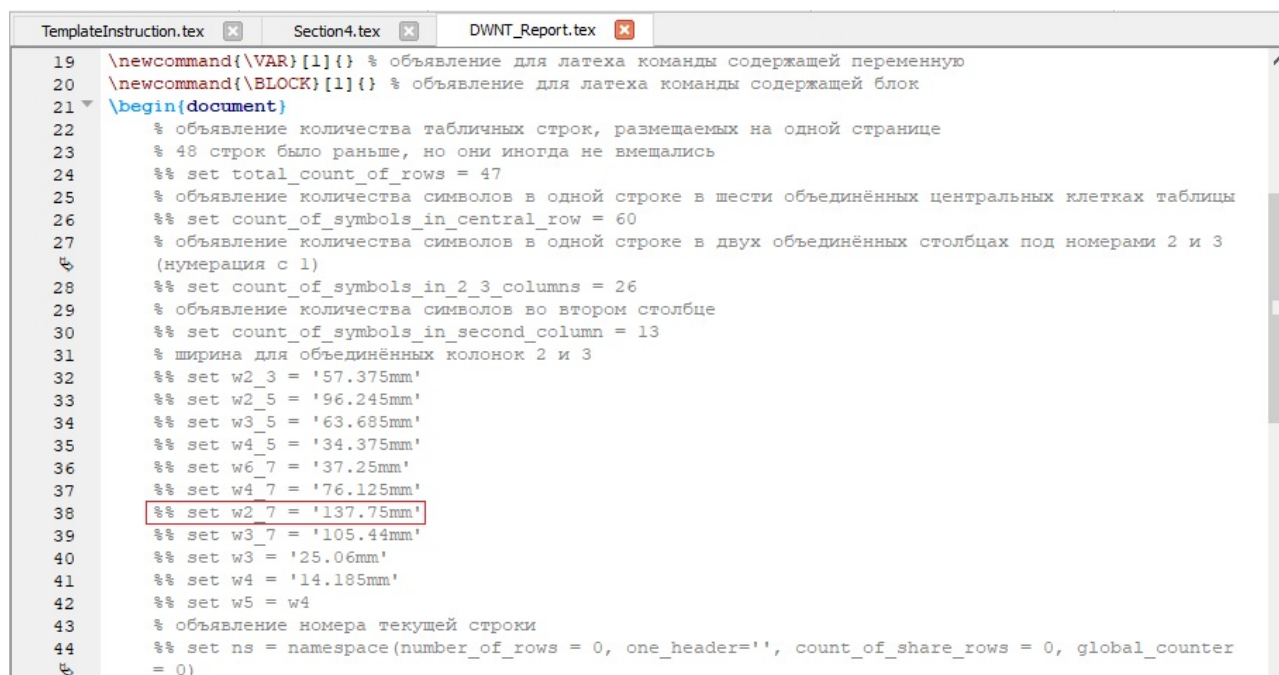


Рис. 25 Установка значения переменной в Jinja2

4.3.16) Теперь при исправленной ошибке в результате компиляции файла 1.3.1) в окне TeXstudio "Messages" будет появляться сообщение, показанное на рис. 16.

## 5. Компиляция файлов через пакетный файл

### 5.1. Запуск шаблонизатора через bat-файл

5.1.1) Запустить пакетный файл 1.4.1). При этом откроется консоль командной строки и окно выбора считываемого XML-файла, как показано на рис. 26. По умолчанию окно выбора XML-файла открывается в папке «C:\avs\Racks\DWNT\dwnt\_20gk-01\Reports», так как в эту папку TestStand обычно генерирует свои отчёты на Автоматизированном стенде проверки Блока информационного обеспечения и синхронизации (АСП БИОС). Однако, если вышеописанной папки не существует, то окно выбора XML-файла открывается в папке, где хранится группа файлов 1.2, как и показано на рис. 26. Такое поведение описано в запуске через bat-файл сценарии 1.4.2). Этот же сценарий осуществляет диалог с пользователем, формируя необходимый список считываемых XML-файлов и их типов заголовков, а затем записывая этот список в текстовый файл 2.3.2).



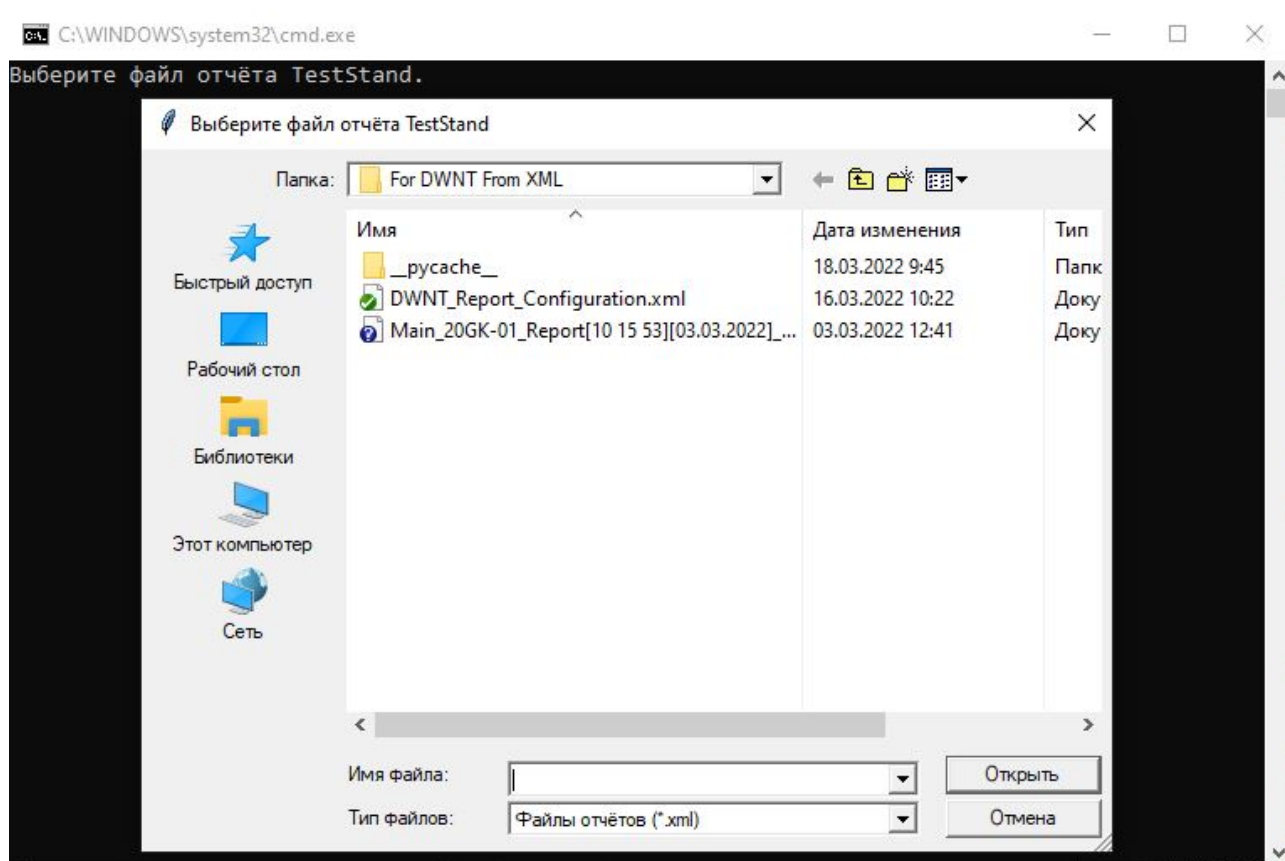


Рис. 26 Окно выбора XML-файла

5.1.2) Если пользователь при выборе первого файла XML нажимает на кнопку "Отмена", то пакетный файл завершает работу, выводя сообщение, указанное на рис. 27.

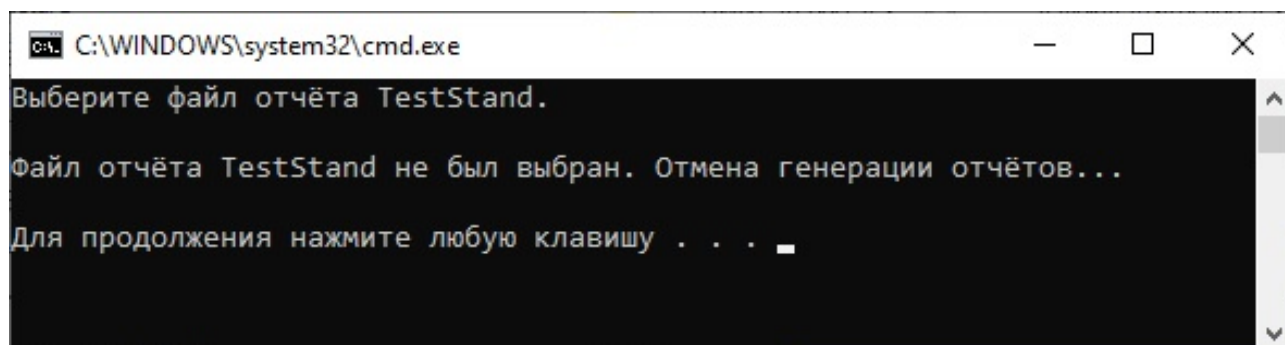


Рис. 27 Сообщение при отмене выбора первого XML-файла

5.1.3) Если же пользователь выбирает файл XML и нажимает кнопку "Открыть", то далее пользователю предлагается ввести тип заголовка для формируемого отчёта по выбранному файлу, как показано на рис. 28. Стоит отметить, что в этой ситуации важно правильно вводить типы заголовков, иначе при получении несуществующего в конфигурационном файле 1.2.4) типа заголовка программа для генерации отчёта выберет тип заголовка, указанный в атрибуте Headers тэга Report этого конфигурационного файла.

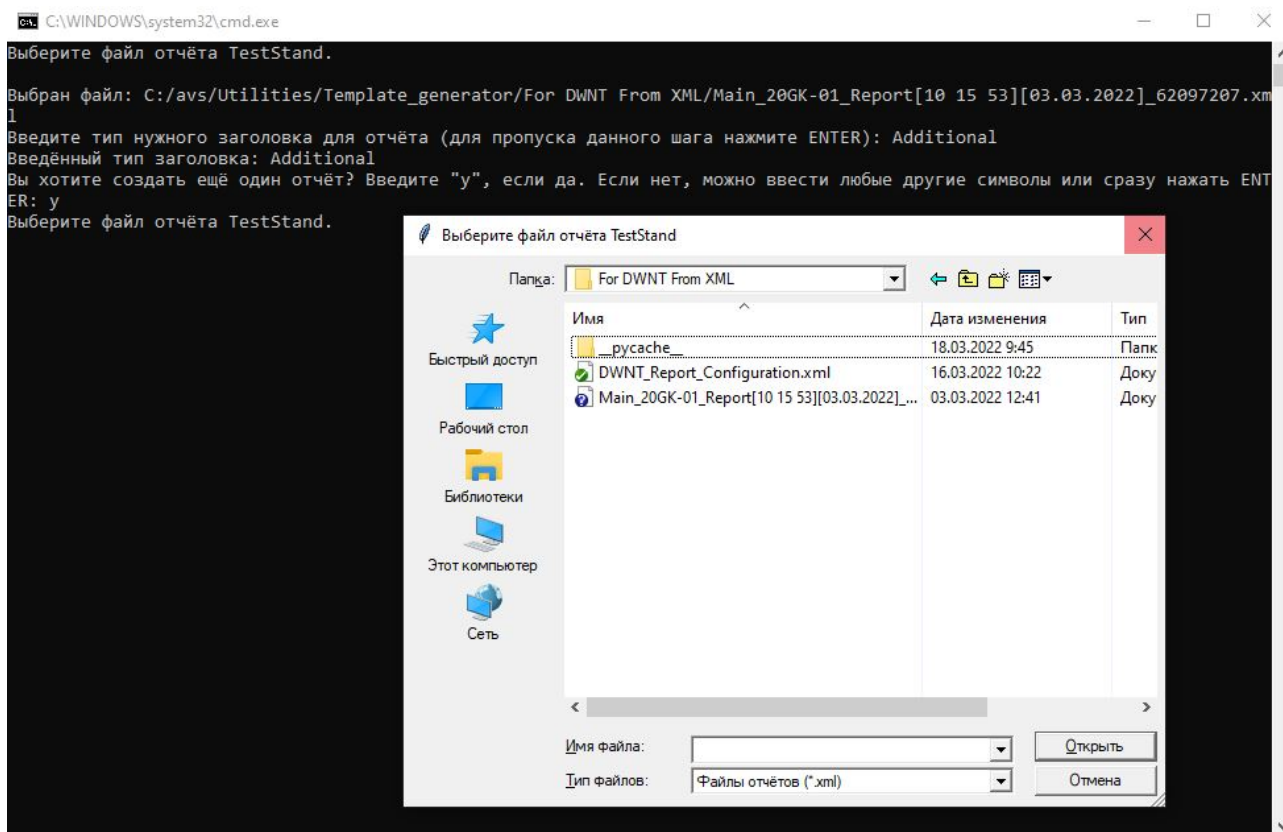


Рис. 28 Один цикл работы пакетного файла

5.1.4) После ввода типа заголовка или подтверждения с помощью нажатия клавиши "Enter" решения об использовании типа заголовка, указанного в конфигурационном файле 1.2.4), пользователю предлагается составить отчет по ещё одному XML-файлу. При вводе символа "y" открывается новое окно с выбором считываемого XML-файла, как показано на рис. 28. Если же пользователь вводит любое другое значение, то запускается генерация PDF-отчёта, как продемонстрировано на рис. 29

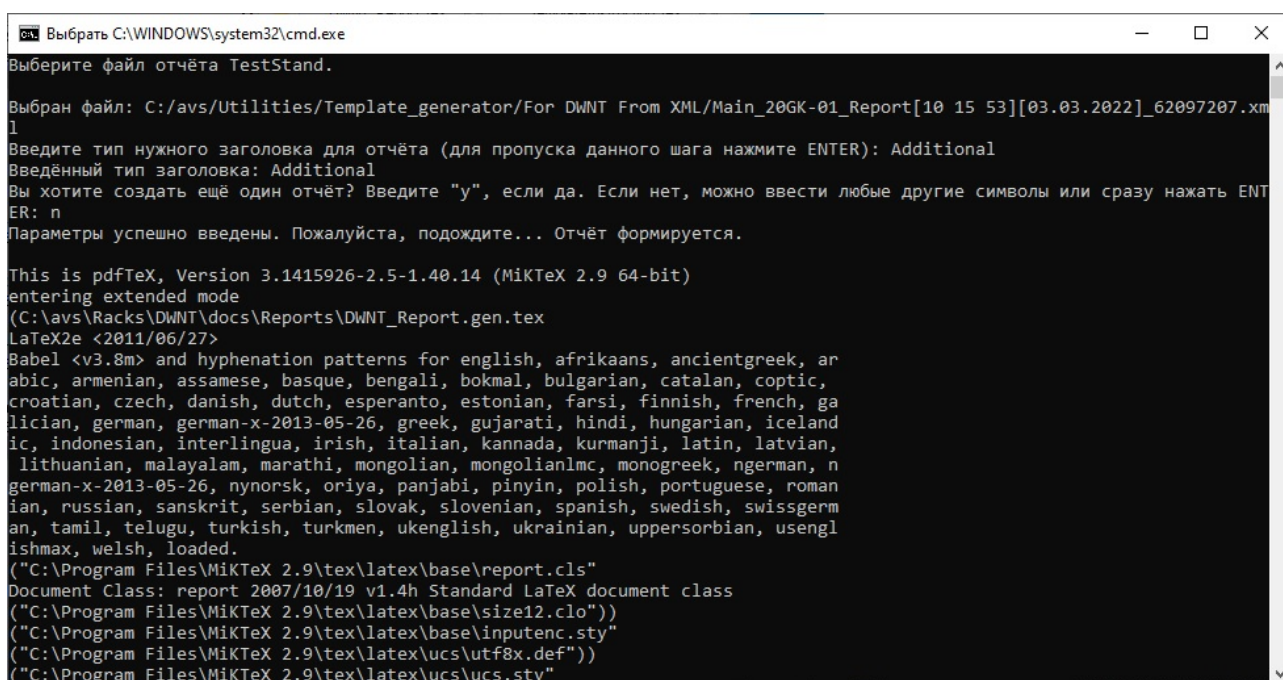


Рис. 29 Запуск генерации при отказе от выбора дополнительного XML-файла

5.1.5) Если же пользователь выбрал ещё один файл XML для формирования отчёта, то пункты 5.1.3) и 5.1.4) повторяются до тех пор, пока пользователь либо не откажется от добавления в генерируемый отчёт информации из ещё одного XML-файла в пункте 5.1.4), либо не нажмёт кнопку "Отмена" в окне выбора очередного файла XML, либо не закроет программу. В первых двух случаях будет сгенерирован PDF-отчёт, включающий в себя данные из всех выбранных ранее файлов XML. На рис. 30 в зелёной прямоугольной рамке показано сообщение пакетного файла шаблонизатора в случае, когда пользователь уже выбрал несколько XML-файлов для генерации по ним отчёта, а при выборе очередного XML-файла в окне выбора отчёта TestStand нажал кнопку "Отмена".

```

C:\WINDOWS\system32\cmd.exe
Выберите файл отчёта TestStand.

Выбран файл: C:/avs/Utilities/Template_generator/For DWNT From XML/Main_20GK-01_Report[10 15 53][03.03.2022]_62097207.xml
1
Введите тип нужного заголовка для отчёта (для пропуска данного шага нажмите ENTER):
Тип заголовка будет определяться по конфигурационному файлу.
Вы хотите создать ещё один отчёт? Введите "у", если да. Если нет, можно ввести любые другие символы или сразу нажать ENTER: у
Выберите файл отчёта TestStand.

Выбран файл: C:/avs/Utilities/Template_generator/For DWNT From XML/Main_20GK-01_Report[10 15 53][03.03.2022]_62097207.xml
1
Введите тип нужного заголовка для отчёта (для пропуска данного шага нажмите ENTER): Additional
Введённый тип заголовка: Additional
Вы хотите создать ещё один отчёт? Введите "у", если да. Если нет, можно ввести любые другие символы или сразу нажать ENTER: у
Выберите файл отчёта TestStand.

Файл отчёта TestStand не был выбран. Будут сгенерированы отчёты только по ранее выбранным файлам.
Параметры успешно введены. Пожалуйста, подождите... Отчёт формируется.

This is pdfTeX, Version 3.1415926-2.5-1.40.14 (MiKTeX 2.9 64-bit)
entering extended mode
(C:/avs/Racks/DWNT/docs/Reports/DWNT_Report.gen.tex
LaTeX2e <2011/06/27>
Babel <v3.8m> and hyphenation patterns for english, afrikaans, ancientgreek, arabic, armenian, assamese, basque, bengali, bokmal, bulgarian, catalan, coptic, croatian, czech, danish, dutch, esperanto, estonian, farsi, finnish, french, galician, german, german-x-2013-05-26, greek, gujarati, hindi, hungarian, icelandic, indonesian, interlingua, irish, italian, kannada, kurmanji, latin, latvian,

```

Рис. 30 Запуск генерации при отмене выбора дополнительного XML-файла

5.1.6) При успешной генерации отчёта выведется сообщение, указанное на рис. 31.

```

C:\WINDOWS\system32\cmd.exe
Overfull \hbox (3.34618pt too wide) in alignment at lines 3385--3451
[] [] [] [] [] [] [] []

Overfull \hbox (3.34618pt too wide) in alignment at lines 3451--3475
[] [] [] [] [] [] [] []
[8]
Overfull \hbox (2.7453pt too wide) in alignment at lines 3477--3497
[] [] [] [] [] [] [] []

Underfull \vbox (badness 10000) detected at line 3497
[9] (C:/avs/Racks/DWNT/docs/Reports/DWNT_Report.aux) )
(see the transcript file for additional information){C:/Program Files/MiKTeX 2.9/fonts/enc/t2/t2a.enc}<C:/Program Files/MiKTeX 2.9/fonts/type1/public/amsfonts/cm/cmsy10.pfb><C:/Program Files/MiKTeX 2.9/fonts/type1/public/amsfonts/cm/cmsy8.pfb><C:/Program Files/MiKTeX 2.9/fonts/type1/public/psycr/times.pfb><C:/Program Files/MiKTeX 2.9/fonts/type1/public/psycr/timesbd.pfb>
Output written on "C:/avs/Utilities/Template_generator/For DWNT From XML/DWNT_Report.pdf" (18 pages, 139487 bytes).
Transcript written on C:/avs/Racks/DWNT/docs/Reports/DWNT_Report.log.
-----"SUCCESS"-----
The PDF report file has been successfully generated in the path 'C:/avs/Utilities/Template_generator/For DWNT From XML/DWNT_62097207_2022-03-03_10-15-53_5.pdf'.
-----
Для продолжения нажмите любую клавишу . . .

```

Рис. 31 Успешная генерация отчёта

5.1.7) При возникновении критических ошибок как в процессе заполнения шаблона данными из XML-файла, так и по ходу преобразования заполненного шаблона в PDF-



файл будет выведено сообщение, продемонстрированное на рис. 32. Также пользователю будет предложено выбрать другой XML-файл, так как предыдущий файл(ы), возможно, был(и) повреждён(ы). Стоит отметить, что при возникновении такой ошибки сбрасывается набор всех выбранных до этого момента пользователем XML-файлов, поэтому, если пользователь нажмёт кнопку "Отмена", то программа завершит работу так же, как и при отмене выбора самого первого XML-файла (подробнее см. пункт 5.1.2)).

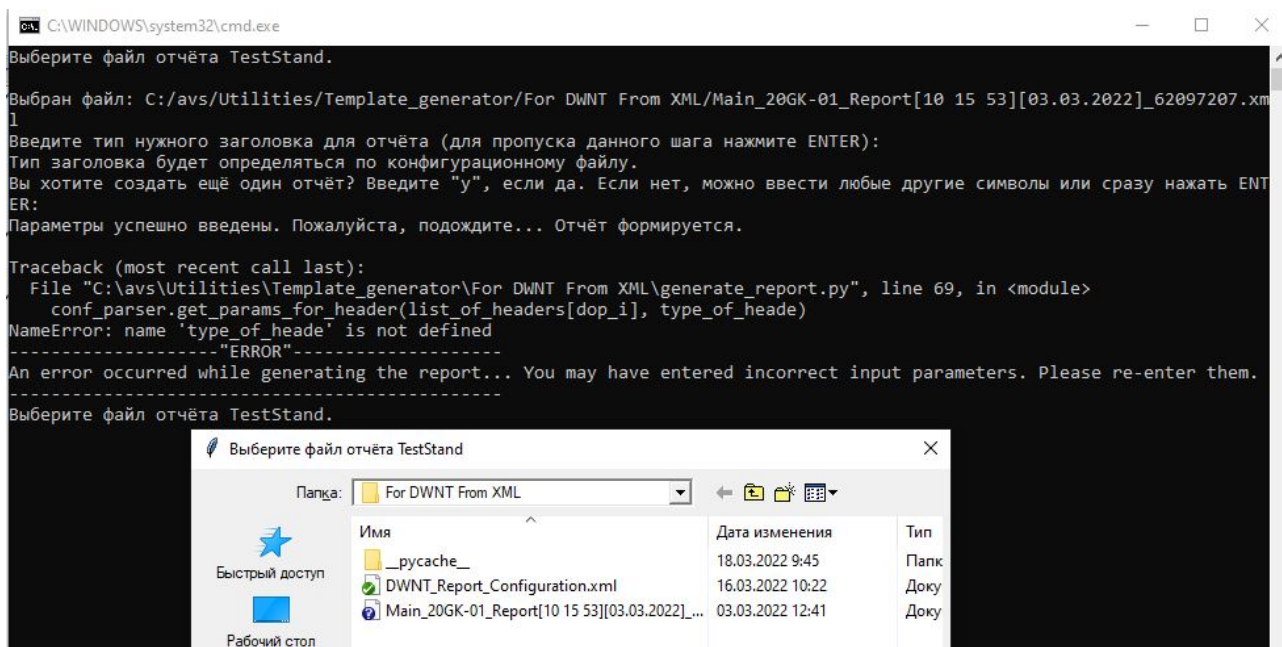


Рис. 32 Возникновение ошибки во время генерации отчёта

5.1.8) Важно отметить, что при возникновении не критических ошибок сообщение, показанное на рис. 32, не будет выведено, а будет указано сообщение об успешной генерации PDF-отчёта. Одним из примеров таких не критических ошибок является отключение одной из используемых в файле 1.3.1) библиотек. Например, для отключения возможности использования команды прорисовки горизонтальной линии различного стиля в каждом столбце таблицы (команды «\hhline{ }») можно удалить строку «\usepackage{hhline}» из начала файла 1.3.1). Но даже после этого генерация отчёта будет успешной, как показано на рис. 33. Единственное, что может смутить пользователя, - это сообщение об обнаруженной неизвестной команде, выделенное на рис. 33 красным прямоугольником. Если же посмотреть на сформированный PDF-файл, то можно увидеть, что горизонтальные линии в таблице были прорисованы неправильно (рис. 34), поэтому важно после любой генерации отчёта визуально просматривать сгенерированные PDF-файлы на наличие ошибок.

```

C:\WINDOWS\system32\cmd.exe

1.1728 ... \tabularnewline \hhline
{~|-|---|-|-|}
! Undefined control sequence.
<recently read> \hhline
1.1732 ... \tabularnewline \hhline
{~|~|---|-|-|}

Overfull \hbox (3.34618pt too wide) in alignment at lines 1715--1739
[] [] [] [] [] [] []
[15] [16]
Overfull \hbox (2.7453pt too wide) in alignment at lines 1741--1761
[] [] [] [] [] [] []

Underfull \vbox (badness 10000) detected at line 1761
[17] (C:\avs\Racks\DWNT\docs\Reports\DWNT_Report.aux)
(see the transcript file for additional information){C:/Program Files/MiKTeX 2.
9/fonts/enc/t2/t2a.enc}<C:/Program Files/MiKTeX 2.9/fonts/type1/public/amsfonts
/cm/cmsy10.pfb><C:/Program Files/MiKTeX 2.9/fonts/type1/public/amsfonts/cm/cmsy
8.pfb><C:/Program Files/MiKTeX 2.9/fonts/type1/public/psycr/times.pfb><C:/Progr
am Files/MiKTeX 2.9/fonts/type1/public/psycr/timesbd.pfb>
Output written on "C:/avs/Utilities/Template_generator/For DWNT From XML\DWNT_R
eport.pdf" (17 pages, 90543 bytes).
Transcript written on C:/avs/Racks/DWNT/docs/Reports/DWNT_Report.log.
-----"SUCCESS"-----
The PDF report file has been successfully generated in the path 'C:\avs\Utilities\Template_generator\For DWNT From XML\DW
NT_62097207_2022-03-03_10-15-53_7.pdf'.
-----
Для продолжения нажмите любую клавишу . . .

```

Рис. 33 Генерация PDF-файла с некритическими ошибками

№	Проверка			Результат
3.2.01	Проверка запуска БИОС при подаче напряжения питания.			Без замечаний
	Проверка выдачи напряжения на некоммутируемые шины питания.			
	Проверка получения команд прямого управления аппаратным			
	Проверка работоспособности БИОС при плавном изменении			Ошибка
3.2.02	Порог отключения первой шины	Измерено (В)	Допустимое значение (В)	Без замечаний
		6.000	0.000÷10.000	
	Порог отключения телеметрии	1.500	0.000÷10.000	Без замечаний
	Порог включения телеметрии	4.000	0.000÷10.000	Без замечаний

Рис. 34 Фрагмент отчёта с ошибочной прорисовкой горизонтальных линий

## 5.2. Краткое описание работы bat-файла

5.2.1) Сначала осуществляется переход в папку, в которой содержится группа файлов 1.2.

5.2.2) Затем осуществляется запуск файла 1.4.2). Данный файл осуществляет диалог с пользователем: собирает сведения об XML-файлах, информацию из которых пользователь хочет занести в формируемый PDF-отчёт, и об их заголовках, которые должны указываться в генерируемом PDF-отчёте. Если данный сценарий запускается впервые, то он определяет, существует ли в системе директория «C:\avs\Racks\DWNT\dwnt\_20gk-01\Reports», являющаяся папкой по умолчанию для формирования XML-отчётов TestStand на АСП БИОС. Если эта папка существует, то окно выбора XML-файла откроется в ней, иначе это окно откроется в папке, где хранится группа файлов 1.2. При повторных запусках окно выбора XML-файла открывается уже в той папке, в которой был выбран XML-файл в предыдущий раз.



5.2.3) В случае выбора XML-файла и указания его заголовка сценарий 1.4.2) сохраняет путь к XML-файлу и заголовок, а при подтверждении пользователем запуска генерации PDF-отчёта заносит сохранённые данные в файл 2.3.2). Если же пользователь отменил генерацию отчёта, то файл 2.3.2) просто очищается записью в него пустой строки.

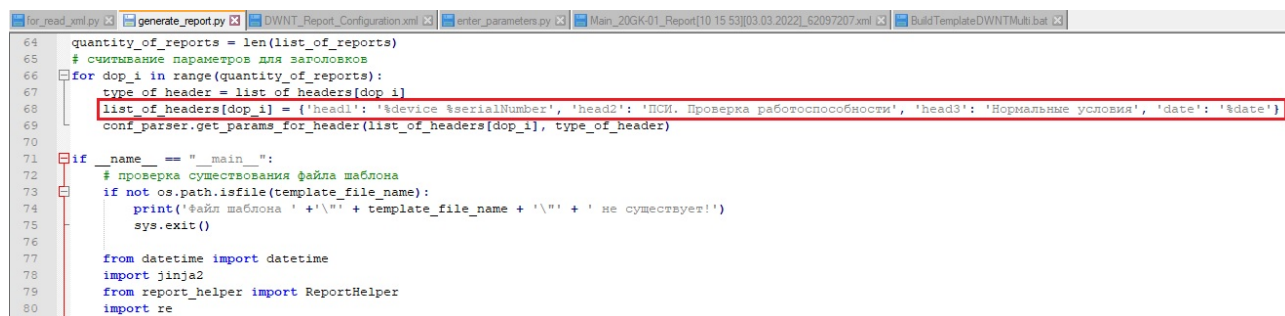
5.2.4) После выполнения работы сценария 1.4.2) пакетный файл 1.4.1) считывает первую строку из файла 2.3.2). Если эта строка пустая, значит, пользователь отменил выбор XML-файла, поэтому bat-файл завершает свою работу. Если же строка непустая, то происходит переход в папку, в которой содержится группа файлов 1.3.

5.2.5) В этой папке с помощью компилятора Python происходит запуск сценария 1.2.1). В качестве входных параметров сценарию 1.2.1) передаются:

- а) входной файл 1.3.1), который нужно заполнить с помощью средства Jinja2 данными из выбранных пользователем XML-файлов;
- б) выходной файл 2.2.2), в который осуществляется запись уже заполненного шаблона;
- в) специальный параметр "bat", указывающий запущенному сценарию, что все остальные параметры (пути к выбранным пользователем XML-файлам и типы их заголовков) нужно считывать с файла 2.3.2).

5.2.6) Из-за переданного в сценарий 1.2.1) параметра "bat" программа считывает дополнительные аргументы с файла 2.3.2) (при компиляции через TeXstudio дополнительные аргументы уже переданы скрипту, поэтому их не нужно считывать с файла 2.3.2)). Для распознавания типа аргумента применяется поиск символа "." в нём. Если символ "." найден, то аргумент принимается за считываемый XML-файл, так как у любого файла в его абсолютном пути есть символ ".", после которого указывается расширение файла. Если же символ "." не найден, аргумент считается типом заголовка, который должен использоваться в формируемом PDF-отчёте перед описанием данных с соответствующего XML-файла. Отсюда следует вывод, что тип заголовка в своём названии никогда не должен содержать символ ".". Если же среди аргументов были встречены два подряд идущих XML-файла, то первому из этих двух файлов в качестве типа заголовка назначается пустая строка, которая в дальнейшем будет являться указанием на использование типа заголовка, прописанного в атрибуте Headers тэга Report в конфигурационном файле 1.2.4).

5.2.7) По считанным типам заголовка для каждого указанного пользователем XML-файла считываются строки самих заголовков из конфигурационного файла 1.2.4). При этом, если указанный пользователем тип заголовка не описан в конфигурационном файле, то вместо него берётся заголовок указанный в атрибуте Headers тэга Report. Но, если и это название типа заголовка некорректно, то в качестве заголовка берётся значение по умолчанию, описанное в файле 1.2.1) и выделенное на рис. 35 в красный прямоугольник. За такое поведение отвечает класс XMLConfigParser в файле 1.2.3).



```
64 quantity_of_reports = len(list_of_reports)
65 # считывание параметров для заголовков
66 for dop_i in range(quantity_of_reports):
67     type_of_header = list_of_headers[dop_i]
68     list_of_headers[dop_i] = {'head1': '%device %serialNumber', 'head2': 'ПСИ. Проверка работоспособности', 'head3': 'Нормальные условия', 'date': '%date'}
69     conf_parser.get_params_for_header(list_of_headers[dop_i], type_of_header)
70
71 if __name__ == "__main__":
72     # проверка существования файла шаблона
73     if not os.path.isfile(template_file_name):
74         print('файл шаблона ' + '\n' + template_file_name + '\n' + ' не существует!')
75         sys.exit()
76
77 from datetime import datetime
78 import Jinja2
79 from report_helper import ReportHelper
80 import re
```

Рис. 35 Строки заголовка по умолчанию

5.2.8) Затем сценарий 1.2.1) считывает первый XML-файл, указанный пользователем, путём создания объекта "e" класса XMLReportParser из файла 1.2.3).

5.2.9) Также создаётся объект "h" класса ReportHelper из файла 1.2.2) для его последующей передачи в средство заполнения шаблона Jinja2.

5.2.10) Следующей операцией считанные из конфигурационного файла 1.2.4) заголовки передаются в созданный объект "h".

5.2.11) Согласно пункту 2.3.1), при генерации PDF-файла через bat-файл имя создаваемого файла должно состоять из серийного номера испытанного прибора, даты и времени начала его тестирования. В связи с этим сценарий 1.2.1), используя созданные ранее объекты "e" и "h", считывает серийный номер, дату и время начала тестирования прибора, указанные в самом первом выбранном пользователем XML-файле, и записывает дату и время в файл 2.3.2). Серийные номера, даты и время приборов, результаты тестирования которых были записаны в другие XML-файлы, при формировании имени генерируемого PDF-файла не участвуют.

При компиляции шаблонизатора через TeXstudio считывается только серийный номер протестированного прибора для его последующего вывода в окно "Messages" среды разработки TeXstudio, как показано на рис. 13 в последней строке блока текста, выделенного в красный прямоугольник.

5.2.12) Затем сценарий 1.2.1) настраивает средство Jinja2, передаёт ему созданные объекты классов "e", "h" и список путей к файлам "list\_of\_reports".

5.2.13) После этого сценарий 1.2.1) запускает заполнение шаблона 1.3.1) с формированием файла 2.2.2) для заполненного шаблона. Само заполнение осуществляется средствами Jinja2.

5.2.14) По завершению работы средства заполнения шаблона Jinja2 сценарий 1.2.1) выводит в стандартный поток вывода сообщение о завершении заполнения шаблона. Среди выводимой информации также есть информация о серийном номере, которая считывается bat-файлом 1.4.1) и записывается в файл 2.3.3).

5.2.15) Затем информация о серийном номере прибора, результаты тестирования которого были указаны в первом считанном XML-файле, снова считывается с файла 2.3.3) для формирования названия генерируемого PDF-файла.

5.2.16) Если считанный серийный номер пуст, значит, во время выполнения сценария 1.2.1) произошла ошибка, поэтому пакетный файл выводит сообщение о возникшей ошибке пользователю и предлагает ему выбрать другие XML-файлы, так как выбранные ранее файлы отчётов TestStand могли оказаться повреждёнными. Таким образом, bat-файл начинает свою работу заново.

5.2.17) Если же серийный номер был записан в файл 2.3.3), то запускается компиляция файла 2.2.2) с помощью приложения «pdflatex» (описание опций см. подробнее в пункте 4.1.8)).

5.2.18) После завершения компиляции 2.2.2) проверяется наличие сгенерированного файла 2.2.1). Если он отсутствует, то bat-файл начинает свою работу заново, предлагая пользователю снова выбрать XML-файлы.

5.2.19) Если же файл 2.2.1) был успешно сгенерирован, то bat-файл считывает с файла 2.3.2) дату и время начала тестирования, результаты которого были описаны в первом выбранном пользователем XML-файле. Затем считанные данные, как и серийный номер прибора, считанный в пункте 5.2.15), используются для формирования нового имени сгенерированного PDF-файла по правилам, описанным в пункте 2.3.1).

5.2.20) Далее выводится информация об успешном завершении работы шаблонизатора и bat-файл заканчивает свою работу.