LinuxDoc+Emacs+Ispell-HOWTO

Автор: Philippe MARTIN (*feloy@wanadoo.fr*)
Перевод на английский: Sébastien Blondeel (*Sebastien.Blondeel@lifl.fr*), перевод на русский Alex Ott
ott@phtd.tpu.edu.ru
v0.4, 27 Февраля 1998

Этот документ предназначен в помощь людям пишущим и переводящим документы серии Linux HOWTOs или другие документы для Linux Documentation Project. Он дает советы по использованию разных утилит, включая Emacs и Ispell.

Содержание

Примечание переводчика: Шлите мне любый комментарии и замечания, даже небольшие.

1 Преамбула

1.1 Авторские права

1.2 Благодарности

Специальное спасибо Sébastien Blondeel, кто является "противной сволочью"и много спрашивал меня о настройке Emacs setup. Его умные вопросы позволили мне лучше понять эту программу и передать это знание вам в этом документе.

1.3 Комментарии

Не колеблясь говорите мне любые вещи, которые помогут сделать этот документ лучше. Я буду тщательно проверять вашу критику.
Также не колеблясь спрашивайте меня о любых вопросах относящихся к темам обсуждаемым здесь. Я буду больше чем счастлив ответить вам, так как это позволит мне в будущем улучшить этот документ.

1.4 Версии

Этот документ рассказывает о следующих версиях продуктов:

- Sgml-tools версии 0.99,
- Emacs версии 19.34,
- Ispell версии 3.1,
- Все библиотеки Emacs на которые ссылаются в этом документе распространяются с вышеуказанной версией Emacs, кроме `iso-sgml`, которая распространяется с XEmacs, и `psgml`, которая является автономной библиотекой.

2 Введение

2.1 SGML

Standard Generalised Mark-up Language, или **SGML**, это язык для определения типов документов.
Например, кто-то может определить тип документа *рецепты*, с первой частью представляющей используемые ингредиенты, вторая часть задает используемые приборы, а третья часть дает пошаговую инструкцию для приготовления кекса, и великолепная финальная картинка для показа результата всего этого.
Это называется *Document Type Definition (DTD, Определение типа документа)*. Оно не определяет как будет выглядеть заключительный результат, оно только определяет только что документ может содержать.
Мы используем тот же пример далее, я уверен, что вы примете мою идею, но у вас может быть свое видение. Тем не менее, они совершенно разные: мое видение представляется как ???? To use the same example again, I'm sure that upon reading my idea of a recipe, you recognised yours, or your favourite cook's. Nevertheless, they actually look different: mine have a picture in the upper left corner of the bathroom cupboard, and the ingredients list can be found in the back garden, between the swimming pool and the barbecue. Yours?
Благодаря этому стандартному определению, кто-то может писать документ, без размышления как он будет выглядеть в конце работы для читателя.

2.2 Определение типа `LinuxDoc`

Этот тип используется для написания, как вы могли догадаться, документов относящихся к Linux.
Такие документы в общем построены следующим образом: они начинаются с заголовка, за которым следует имя автора, номер версии и дата. Затем идет краткое изложение (??? abstract) (так что вам не надо просматривать документ полностью до понимания, того что этот документ не то что вам нужно), затем идет содержание, которое показывает структуру, так что вы в спешке перейти к части, которую вы хотите прочитать.
Затем идет список глав, разделов, параграфов. В их пределах вы можете вставлять куски программ, изменять шрифт для выделения слова иди предложения, вставлять списки, делать ссылки на другие разделы документа и т.п.
Для написания такого документа, вам всего лишь необходимо указать в нужное время заголовок документа, автора, дату и версию документа, главы и разделы, во время вставки списка указать его элементы и т.п.

2.3 `SGML-Tools`

SGML-Tools превратят спецификацию документа в окончательный результат в выбранной вами форме. Если вы хотите поместить документ в вашу персональную библиотеку, вы выберете

PostScript. Если вы хотите организовать доступ к нему через Web, то это будет *HTML*. Если вы не можете ничего сделать и должны читать его под Windows, вы можете преобразовать его в *RTF*, чтобы была возможность читать его любым текстовым процессором. Или можете использовать все три формата для отражения смены вашего настроения.
SGML-Tools доступны через анонимное FTP по адресу *ftp://ftp.lip6.fr/pub/sgml-tools/*

3 Ваш первый документ

3.1 From a text document

Если вы хотите превратить текстовый документ в SGML для переноса его в другие форматы, то вот как это делается:

1. Добавте следующие строки в самое начало документа:

```
<!doctype linuxdoc system>
<article>
<title>Title Goes Here</title>
<author>
  name of author, author's e-mail, etc.
</author>
<date>
 version and date
</date>
```

2. Если вы кратко описываете содержание документа в самом начале документа, то окружите этот параграф тагами <abstract> и </abstract>.

3. Затем вставьте таг <toc>, который обозначает *Содержание*.

4. В начале каждой главы замените каждую строку дающую номер и заголовок главы на:

```
<sect>Заголовок Главы
```

   и добавьте таг </sect> в конец главы.
   Замечание : Вам не надо помещать номер главы, это сделается автоматически.

5. Сделайте то же самое для разделов. Вам необходимо удалить их номера и отметить их тагами <sect1> в начале раздела и тагами </sect1> в конце раздела.

6. Вы можете также определить 4 уровня вложенности в разделах, используя <sectN> и </sectN>, где N= 2, 3, или 4 также как и в предыдущем случае.

7. В начале каждого параграфа вставьте таг <p>.

8. Если вам необходимо выделить некоторые части, то обозначьте их тагами <it> и </it> (*курсив*), <bf> и </bf> (**жирный шрифт**), или <tt> и </tt> (стиль пишущей машинки).

9. Для вставки примерно такого списка:

```
Это четырехстрочный список:

 - здесь первая строка
 - затем вторая строка
 - и еще одна
 - это все
```

вы должны заменить его на:

```
 Это четырехстрочный список:
<itemize>
<item>здесь первая строка
<item>затем вторая строка
<item>и еще одна
<item>это все
</itemize>
```

10. Когда весь блок является частью программы, или чего-нибудь другого, что нуждается в неизменной передаче, используйте:

```
<verb>
 10 REM Oh my God what's this?
 20 REM I thought this had long disappeared!
 30 PRINT "I am back to";
 40 PRINT "save the world."
 50 INPUT "From whom, do you reckon? ",M$
 60 IF M$="Bill" THEN PRINT "Thou art wise.":GOTO PARADISE
 70 ELSE PRINT "You ain't got a clue...":GOTO RICHMOND
 </verb>
```

11. Таким образом, ваше мастерство форматирования SGML будет довольно приличным. Если вы хотите усовершенствовать ваш документ, вы можете заглянуть в руководство пользователя **SGML-Tools**, которое дает более детальную информацию о типе документа **LinuxDoc**.

4 Настройка Emacs

4.1 Символы с диактрическими знаками

Если вы хотите писать документы на французском или на каком-то другом языке западной Европы, то вам необходимо использование восьмибитных символов. Здесь рассказывается как настроить Emacs, чтобы он воспринимал такие символы.

4.1.1 Отображение 8-битных символов

Чтобы позволить Emacs отображать 8-битные символы, вам необходимо добавить следующие строки в ваш файл `.emacs`:

```
(standard-display-european 1)
(load-library "iso-syntax")
```

Если вы используете Emacs на терминале, который не имеет 8-битной поддержки, то вы можете использовать библиотеку `iso-ascii` (`(load-library "iso-ascii")`), которая заставляет Emacs отображать такие символы с лучшим приближением.

4.1.2 Набор 8-битных символов

Если ваша клавиатура позволяет вам вводить символы, с диактрическими знаками то нет никаких проблем. А вот если не позволяет, то есть способ решающий эту проблему:

Библиотека `iso-acc` Библиотека `iso-acc` для Emacs позволит вам печатать 8-битные символы на 7-битной клавиатуре.
Для ее использования вставте в ваш файл `.emacs` такую строчку:

```
(load-library "iso-acc")
```

Затем, после запуска Emacs и открытия файла, который вам надо отредактировать, наберите `Meta-x iso-accents-mode`.
Вы можете затем ввести **é** во французском слове *café* набрав **'** затем `e`. В общем, вы будете набирать сначала знак ударения (??? accent), а затем символ на котором ставится диактричекий знак (в верхнем или нижнем регистре). Могут использоваться следующие диактрические знаки:

' : Acute

` : Grave

: Circumflex

" : Dieresis

: Тильда, cedilla, и другие частные случаи (cf iso-acc.el).

/ :Для перечеркивания символа, и т.п.

Если вам нужен один из этих символов, а не символ с диактрическим знаком, наберите пробел вслед за набором специального символа Например, для набора *l'éléphant*, наберите `l ' ⟨spc⟩ ' e l ' e ...`
Вы найдете список всех возможных комбинаций в файле `iso-acc.el`.

Клавиша ⟨*Meta*⟩ Некоторые терминалы позволят вам набрать 8-битные символы с помощью клавиши ⟨*Meta*⟩ (или ⟨*Alt*⟩). Например, нажатие ⟨*Meta*⟩**-i** даст вам символ **é**.
Но Emacs резервирует клавишу ⟨*Meta*⟩ для собственного использования, и я не знаю библиотеку, которая позволит вам использовать эту клавишу для символов с диактрическими знаками.
Вот решение этой проблемы:

```
(global-set-key "\ei" '(lambda () (interactive) (insert ?\351)))
```

Такая строка, если будет вставлена в ваш файл `.emacs`, позволит набирать вам **é** используя сочетание клавиш ⟨*Meta*⟩**-i**. Вы можете переопределить таким образом те сочетания клавиш которые вам нужны, если вы замените **i** нужной клавишей и **351** нужным кодом (код был взят из набора символов ISO-8859-1).
Предупреждение! Некоторые локальные режимы могут переопределять такие сочетания клавиш.

4.1.3 Отображение 8-битных символов SGML

В SGML, вы можете печатать символы с диактрическими знаками с помощью макросов. Например, клавиша **é** обозначена как **é**. В общем приложения, которым надо читать SGML могут читать 8-битные символы и нет необходимости использовать эти макросы. Но некоторые программы не могут делать это. Существует способ, который позволит избежать краха приложений.

Библиотека `iso-sgml` позволит вам печатать символы с диактрическими знаками в Emacs, но при сохранении вашего файла на диск, он превратит 8-битные символы в их SGML-эквиваленты.
Поэтому легко, спасибо библиотеке, набирать и читать ваши документы в Emacs, и вы можете быть уверены, что не 8-битные приложения будут понимать ваши документы.
Для использования этой библиотеки вам просто надо добавить следующие строки в ваш файл `.emacs`:

```
(setq sgml-mode-hook
'(lambda () "Defaults for SGML mode."
 (load-library "iso-sgml")))
```

4.2 Режим SGML

При загрузке файла с расширением **.sgml**, Emacs автоматически запускает **режим sgml**. Если это не делается, вы можете задать это вручную набрав `Meta-x sgml-mode`, или автоматически, добавив следующие строки в ваш файл `.emacs`:

```
(setq auto-mode-alist
(append '(("\.sgml$"  . sgml-mode))
          auto-mode-alist))
```

Этот режим позволит вам например, выбрать как вставлять 8-битные символы. С помощью `Meta-x sgml-name-8bit-mode` (или пункта меню *SGML/Toggle 8-bit insertion*), вы можете выбрать как печатать 8-битные символы — как есть, или в форме SGML form, например в форме **&...;**.
Этот режим также позволит вам показывать или прятать таги SGML, с помощью `Meta-x sgml-tags-invisible` (или пункта меню *SGML/Toggle Tag Visibility*).

4.3 Режим PSGML

Режим PSGML позволит вам более удобно редактировать документы в SGML с помощью Emacs.
Документация *psgml-linuxdoc* объясняет как установить этот режим и использовать его вместе с *LinuxDoc*.

4.4 Разное

4.4.1 Режим auto-fill

В нормальном режиме, когда вы печатаете абзац и достигаете конца строки, вы должны сами использовать клавишу ⟨*Return*⟩ для перехода на следующую строку, или весь параграф будет состоять из одной строки. Когда вы используете клавишу ⟨*Return*⟩ для перехода к следующей строке, то вы получаете абзацы с неровными правыми границами.
Если вы позволите некоторым строкам превысить разумную ширину, то вы не сможете просматривать их в некоторых редакторах.
Режим **auto-fill** автоматизирует эту скучную задачу: когда вы перейдете далее определенной колонки, (по умолчанию 70-ая), то вы автоматически перейдете на следующую строку.
Теперь расскажем как установить этот режим, и установить ширину текста равной 80 символам:

```
(setq sgml-mode-hook
      '(lambda () "Defaults for SGML mode."
          (auto-fill-mode)
          (setq fill-column 80)))
```

5 Ispell

Если вы хотите проверять свои тексты внутри Emacs, вы можете использовать программу **Ispell** и ее режим для Emacs.

5.1 Выбор ваших словарей по умолчанию

Вы можете установить Emacs так, что при загрузке файла он будет автоматически выбирать, который словарь использовать (вы можете использовать несколько). Первый из них, наверное самый важный, это главный словарь, распространяемый вместе с Ispell. Вы можете выбирать между несколькими языками. Второй словарь это ваш персональный словарь, куда Ispell будет вставлять слова, которые он не может найти в основном словаре, но вы требуете, чтобы он запомнил их.
Если вы хотите французский словарь Ispell как словарь по умолчанию, и хотите использовать файл `.ispell-dico-perso` в вашей домашней директории как личный словарь, то вставте следующие строки в ваш файл `.emacs` file:

```
(setq sgml-mode-hook
'(lambda () "Defauts for SGML mode."
(setq ispell-personal-dictionary "~/.ispell-dico-perso")
(ispell-change-dictionary "francais")
))
```

5.2 Выбор специальных словарей для определенных файлов

У вас могут быть небольшие проблемы, если вы не все время используете проверку документов на одном и том же языке. Если вы переводите документ, то скорее всего, что вы очень часто меняете языки (и словари).
Я не знаю способа (на языке Lisp) выбора, либо автоматически, либо одним щелчком мыши, основного и персонального словаря ассоциированных с языком, который в настоящее время используется. (Если вы знаете как, пожалуйста сообщите мне!)
Однако, возможно показать в конце файла, какие словари вы хотите использовать для текущего файла (и только для него). Достаточно только добавить их как комментарий, так что Ispell сможет прочитать эти установки при запуске проверки:

```
<!-- Local IspellDict: english -->
<!-- Local IspellPersDict: ~/emacs/.ispell-english -->
```

Если вы первоначально определили в вашем файле `.emacs`, что вашим словарем по умолчанию является французский, то вы можете добавить эти строки в конец любого файла, написанного на английском языке.

5.3 Проверка вашего документа

Для проверки всего документа, используйте команду `Meta-x ispell-buffer`. Вы можете также проверить только выделенный блок текста:

- Отметьте начало блока сочетанием `Ctrl-Spc` (mark-set-command),
- Перейдите в конец проверяемого блока,
- наберите команду `Meta-x ispell-region`.

Затем Emacs запустит Ispell. Если Ispell встречает неизвестное слово, он покажет слово (обычно подсвеченное) и будет ожидать нажатие клавиши:

- **spc** пропускает текущее слово
- **i** считает слово правильным и вставляет в персональный словарь,
- **a** считает слово правильным для текущего сеанса работы
- **A** читает слово правильным и вставляет его в словарь данного файла
- **r** позволяет вам исправить слово вручную
- **R** позволяет вам исправить все вхождения неправильно написанного слова,
- **x** останавливает проверку, и помещает курсор на стартовую позицию,
- **X** останавливает проверку и оставляет курсор на текущей позиции, позволяя вам откорректировать ваш файл; вы сможете продолжить проверку, если вы введете команду `Meta-x ispell-continue`,
- **?** дает оперативную справку.

Если ispell находит одно или несколько слов похожих на неизвестное, то он показывает их в маленьком окне, каждое из слов начинается с числа. Просто наберите это число для замены неправильного слова правильным.

5.4 Персональный словарь против словаря локального файла

Клавиша **i** позволит вам вставить слово в ваш персональный словарь, в то время как клавиша **A** вставит слово в словарь локального файла
Словарь локального файла — это последовательность слов, вставленных в конец файла как комментарий, они считываются Ispell каждый раз, когда он запускается для этого файла. Этим способом вы можете указать слова, которые будут считаться правильными только для этого файла, но не для остальных файлов.
Так как я имею отношение к рассматриваемому вопросу, я думаю, что персональный словарь зарезервирован для слов, которые не знает главный словарь, но которые принадлежат этому языку (подобно перенесенным словам), плюс для некоторых общих слов, таких как имен собственных или других (таких как *Linux*), если они не выглядят слишком похоже на слова в главном словаре; добавление слишком большого количества слов в персональный словарь, таких как имена, может быть опасным, потому-что они могут выглядеть как слова текущего языка (некотрые могут представить, что Ispell был озадачен следующим предложением: '*When the going gets tof, the tof get going*[1]'!).

[1] *Tof* это французское сокращение имени *Christophe*.

5.5 Проверка при наборе текста

Ispell может проверять ваш файл во время набора текста. Для этого вам необходимо использовать **ispell-minor-mode**. Для его запуска наберите `Meta-x ispell-minor-mode`. Ispell будет *подавать звуковой сигнал*, каждый раз когда вы набираете слово, которое он не знает.
Если эти *сигналы* раздражают вас (или ваш сосед по комнате лег отдохнуть), то вы можете заменить эти надоедливые сигналы на вспышку на экране с помощью команды `Meta-x set-variable RET visible-bell RET t RET`. Вы можете добавить следующую строку в в ваш файл `.emacs` и заглушить Emacs навсегда:

```
(setq visible-bell t)
```

6 "Грязные"приемы

6.1 Вставка заголовка автоматически

Emacs позволяет вам *перехватить* некоторые действия на любые события (открытие файла, сохранение, запуск в новом режиме и т.п.).
Библиотека **autoinsert** использует это свойство: когда вы открываете новый файл в Emacs, эта библиотека вставляет, соответственно типу файла некоторый *стандартный* заголовок.
В нашем случае этот *стандартный* заголовок, может содержать объявления типа документа (LinuxDoc), заголовок самого документа, имя автора, и дату.
Я здесь опишу два способа для вставки таких заголовков. Вы можете вставить файл шаблона, который содержит вставляемую информацию, или вы можете запустить процедуру на **elisp**.

6.1.1 Вставка файла

Вы должны сначала заставить Emacs запустить `auto-insert` при открытии файла, затем считать библиотеку **autoinsert**, которая объявляет список `auto-insert-alist`, который необходимо изменить. Этот список определяет вставляемые заголовки для каждого типа файлов. По умолчанию, вставляемый файл должен находиться в директории `/insert/`, но можно переопределить переменную `auto-insert-directory`, если вы хотите поместить заголовки в другое место.
Добавьте следующие строки в ваш фал `.emacs` для вставки файла `/emacs/sgml-insert.sgml` при каждом открытии нового файла SGML:

```
(add-hook 'find-file-hooks 'auto-insert)
(load-library "autoinsert")
(setq auto-insert-directory "~/emacs/")
(setq auto-insert-alist
      (append '((sgml-mode .  "sgml-insert.sgml"))
              auto-insert-alist))
```

Затем вы можете написать файл `/emacs/sgml-insert.sgml` тот заголовок, который вам необходим, затем перезапустить Emacs и открыть какой-нибудь новый файл (например `foobar.sgml`). Emacs должен спросить у вас подтверждение автоматической вставки заголовка, и если вы ответите утвердительно, то он вставит заголовок.

6.1.2 Запуск программы

Этот метод работает почти как предыдущий, но вместо установки указателя `auto-insert-alist` на вставляемый файл, вам необходимо указать какую процедуру надо выполнить. Теперь как это работает, допуская, что вы хотите записать эту функцию в файл, названный `/emacs/sgml-header.el`. (нет необходимости перегружать ваш файл `.emacs` такими функциями, поскольку они могут долго не использоваться):

```
(add-hook 'find-file-hooks 'auto-insert)
(load-library "autoinsert")
(add-to-list 'load-path "~/emacs")
(load-library "sgml-header")
(setq auto-insert-alist
      (append '(((sgml-mode .  "SGML Mode") . insert-sgml-header))
              auto-insert-alist))
```

Вы найдете в разделе A (appendix) пример функции `insert-sgml-header`.

A Функция `insert-sgml-header`

Эта функция позволит пользователю вставить настраиваемый заголовок для документа Linux Documentation Project в файл. Она может быть вызвана автоматически при открытии нового файла, или может быть вызвана пользователем.
Эта функция запрашивает у пользователя, пользуясь *mini-buffer*, некоторую информацию, часть из которой необходима, а часть нет. not.
Сначала идет заголовок документа. Если ничего не дано, то функция сразу прекращает выполнение и ничего не вставляет. Затем идет дата, автор, его электронный адрес и домашняя страница (эти два последних пункта являются опциональными).
Затем идет запрос имени переводчика документа. Если его нет, то просто нажмите клавишу *Return*, и больше запросов не будет. А если переводчик есть, то вас запросят о его электронном адресе и домашней странице.
Затем эта функция выводит все в текущий буфер,включая конечно всю информацию, которую вы ввели и включая также необходимые таги для краткого содержания и первой главы. В конце он поместит курсор в том место, где будет набираться краткое содержание.

```
(defun insert-sgml-header ()
  "Inserts the header for a LinuxDoc document"
  (interactive)
  (let (title author email home translator email-translator home-translator date
              starting-point)
    (setq title (read-from-minibuffer "Title: "))
    (if (> (length title) 0)
        (progn
          (setq date (read-from-minibuffer "Date: ")
                author (read-from-minibuffer "Author: ")
                email (read-from-minibuffer "Author e-mail: ")
                home (read-from-minibuffer "Author home page: http://")
                translator (read-from-minibuffer "Translator: "))
          (insert "<!doctype linuxdoc system>\n<article>\n<title>")
          (insert title)
        (insert "</title>\n<author>\nAuthor: ") (insert author) (insert "<newline>\n")
```

```
  (if (> (length email) 0)
      (progn
        (insert "<htmlurl url=\"mailto:")
        (insert email) (insert "\" name=\"") (insert email)
        (insert "\"><newline>\n")))
  (if (> (length home) 0)
      (progn
        (insert "<htmlurl url=\"http://")
        (insert home) (insert "\" name=\"") (insert home)
        (insert "\">\n<newline>")))
  (if (> (length translator) 0)
      (progn
      (setq email-translator (read-from-minibuffer "Translator e-mail: ")
          home-translator (read-from-minibuffer "Translator home page: http://")
        (insert "Translator : ")
        (insert translator)
        (insert "<newline>\n")
        (if (> (length email-translator) 0)
            (progn
              (insert "<htmlurl url=\"mailto:")
              (insert email-translator) (insert "\" name=\"")
              (insert email-translator)
              (insert "\"><newline>\n")))
        (if (> (length home-translator) 0)
            (progn
              (insert "<htmlurl url=\"http://")
              (insert home-translator) (insert "\" name=\"")
              (insert home-translator)
              (insert "\"><newline>\n")))))
  (insert "</author>\n<date>\n")
  (insert date)
  (insert "\n</date>\n\n<abstract>\n")
  (setq point-beginning (point))
(insert "\n</abstract>\n<toc>\n\n<sect>\n<p>\n\n\n</sect>\n\n</article>\n")
  (goto-char point-beginning)
  ))))
```