Vim - config

Breakindent

Update: патч принят в основную ветку vim'а

Update: проблема с перемещением по тексту при совместной работе showbreak и breakindent. Подробности по ссылке.

Проблема: vim при переносе длинных строк не учитывает первоначальный отступ строки, например, при наличии вложенных многоуровневых структур в исходном коде (*if ... end*, *for ... end*, *\begin{center} ... \end{center}* и т.д.), что приводит к образованию горизонтальных блоков, ухудшающих визуальное восприятие текста.

Так выглядит текст по-умолчанию:

```
3  sub myfunc() {
4      for () {
5  Строка в которой нет "\n". Строка в которой нет "\n". Строка в которой нет "\n". Строка в которой нет
   "\n". Строка в которой нет "\n". Строка в которой нет "\n". Строка в которой нет "\n". Строка в которой
   нет "\n". Строка в которой нет "\n". Строка в которой нет "\n". Строка в которой нет "\n". Строка в
   которой нет "\n". Строка в которой нет "\n". Строка в которой нет "\n". Строка в которой нет "\n". Строка
   в которой нет "\n". Строка в которой нет "\n". Строка в которой нет "\n". Строка в которой нет "\n".
   Строка в которой нет "\n".
6      }
7  }
```

Таким должен быть результат «умного» переноса строк:

```
3  sub·myfunc()·{
4  ····for·()·{
5  ········Строка·в·которой·нет·"\n".·Строка·в·которой·нет·"\n".·Строка·в·которой·нет·"\n".·Строка·в·которой·
           нет·"\n".·Строка·в·которой·нет·"\n".·Строка·в·которой·нет·"\n".·Строка·в·которой·нет·"\n".·Строка·
           в·которой·нет·"\n".·Строка·в·которой·нет·"\n".·Строка·в·которой·нет·"\n".·Строка·в·которой·нет·
           "\n".·Строка·в·которой·нет·"\n".·Строка·в·которой·нет·"\n".·Строка·в·которой·нет·"\n".·Строка·в·
           которой·нет·"\n".·Строка·в·которой·нет·"\n".·Строка·в·которой·нет·"\n".·Строка·в·которой·нет·"\n".
           Строка·в·которой·нет·"\n".·Строка·в·которой·нет·"\n".
6  ····}
7  }
```

Решение: использовать breakindent-patch

Пересборка VIM'а в Debian

- **Установка инструментов сборки пакетов**

```
apt-get install dpkg-dev devscripts
```

- **Получение исходного кода**

```
# wheezy(stable) version 7.3.547
apt-get source vim
# or
# unstable version 7.4.253 (at 13.04.2014)
```

```
apt-get source vim -t unstable
```

- **Загрузка зависимостей необходимых для сборки пакета**

```
apt-get build-dep vim
# for unstable version may need update libperl, python
```

- **Загрузка breakindent патча с домашней страницы**

```
# for wheezy vim-7.3.547
wget https://retracile.net/raw-attachment/blog/2012/12/21/17.30/vim-7.3.682-breakindent.patch
# or
# for unstable vim-7.4.253
wget https://retracile.net/raw-attachment/blog/2013/09/12/23.00/vim-7.4.16-fc19-breakindent.patch
```

- **Помещение breakindent патча в каталог debian-заплаток**

```
# for wheezy
cp vim-7.3.682-breakindent.patch vim-7.3.547/debian/patches/debian/vim-breakindent.patch
# or
# for unstable
cp vim-7.4.16-fc19-breakindent.patch vim-7.4.253/debian/patches/debian/vim-breakindent.patch
```

- **Учёт нового патча при сборке пакета:** добавление записи «debian/vim-breakindent.patch» в конец файла «debian/patches/series»
- **Изменить changelog:** в файле «debian/changelog» произвести увеличение версии файла и добавить описание внесённых изменений.
- **Из каталога исходных данных запустить сборку проекта**

```
debuild -us -uc
```

- **Установить требуемые *.deb пакеты vim'а**

```
dpkg -i vim_pkg_name.deb
```

Локальные версии исходного кода патчей и собранных пакетов vim'а можно скачать здесь.

Пакеты собраны с libper5.18, но в репозитории Debian'а данная библиотека заменена на libper5.20. В итоге зависимости нарушены. Под libperl5.20 пакеты надо пересобирать...

Внешний вида

Зелёная цветовая схема spring, Шрифт - Droid Sans Mono 10:

```
if has("gui_running")
    set guioptions+=b
    colorscheme spring
    set guifont=Droid\ Sans\ Mono\ 10
else
    colorscheme spring
endif
```

Подсветка

```
syntax on       " Отображение подсветки `
set syntax=automatic " Автоматическое определение вида подсветки
```

Строки текста

```
set wrap " переносить строки, если они не помещаются на экране
set showbreak=\|-> " установить '|->', как символ продолжения предыдущей строки
set linebreak  " перенос строк по словам, а не по буквам
set number " включить номера строк
```

Установка параметров для разных типов файлов

Поставленная задача может быть реализована, пруф, посредством команды vim'а **autocmd**.

- **Проверка поддержки autocmd (выполнение в vim'е):**

  ```
  :version
  ```

  Доступные опции отмечены (+): «+autocmd».

- **Получение списка всех известных типов файлов (выполнение в vim'е):**

  ```
  :echo join(map(split(globpath(&rtp, 'ftplugin/*.vim'), '\n'),
  'fnamemodify(v:val, ":t:r")'), "\n")
  ```

  This gets all ftplugin scripts from the runtimepath, and then modifies the filespec via fnamemodify() to only list the root of the filename. split() converts into a List, and join() back to lines for :echo ing источник.

- **Узнать тип открытого файла:**

  ```
  :set filetype?
  ```

- **Пример установки параметров для различных типов файлов:**

  ```
  if has("autocmd")
    "Включаем определение типов файлов,
    "без этого не будут срабатывать события в  autocmd
  ```

```
    filetype on

    autocmd FileType php        setlocal ts=4 sts=4 sw=4 noet
    autocmd FileType python     setlocal ts=4 sts=4 sw=4 et
    autocmd FileType html       setlocal ts=2 sts=2 sw=2 noet

  " Ручная установка типа файла по расширению
    autocmd BufNewFile,BufRead *.tmpl,*.tpl setfiletype html
endif
```

Сворачивание блоков текста (Folding)

- **Включение режима фолдинга:**

```
:set foldenable
:set foldmethod=<method_name>
```

- **Методы формирования сворачиваемых блоков (method_name):**
  1. *manual* - формирование блоков происходит вручную с указанием границ блока;
  2. *indent* - блоки формируются на основе отступов, которые есть в тексте;
  3. *expr* - блоки формируются в зависимости от выражения в параметре foldexpr;
  4. *syntax* - блоки формируются на основе разбора синтаксических конструкций в файле, если таковые имеются;
  5. *diff* - применяется для сворачивания одинакового текста в режиме сравнения (diff) двух файлов;
  6. *marker* - метод похож на ручное формирование блоков, за тем исключением, что блоки формируются посредством вставки специальных маркеров в исходный текст. По-умолчанию началу блока соответствуют **{{{**, а концу **}}}**. Линия свёрнутого блока будет отображать текст, находящийся в строке перед началом блока.
- **По-умолчанию, начало и конец блока обрабляются символами комментария:**

```
:set commentstring=/*%s*/
" маркер обрамлён комментарием
/*{{{*/
/*}}}*/
```

  При необходимости это поведение можно изменить

```
:set commentstring=%s
" маркер без комментария
{{{
}}}
```

- **Основные комманды работы с блоками текста**
  - *Создание* **F**old creation
    - **zf#j** - создать блок от курсора вниз на #-строк;
    - **zf/string** - создать блок от курсора до первого вхождения строки «string»;
  - *Открытие* **O**pen a fold
    - **zo** - открыть верхний блок под курсором;
    - **zO** - открыть все блоки под курсором;
    - **zR** - открыть все блоки в документе;

- *Закрытие* **C**lose a fold
  - **zc** - закрыть текущий блок под курсором;
  - **zC** - закрыть все блоки под курсором;
  - **zM** - закрыть все блоки в документе;
- *Изменение* состояния блока
  - **za** - если верхний блок под курсором открыт - закрыть, и наоборот;
  - **zA** - тоже, но относится ко всем вложенным блокам под курсором;
- *Удаление*
  - **zd** - блока под курсором;
  - **zE** - всех блоков в документе;
- *Перемещение*
  - **zj** - к следующему блоку;
  - **zk** - к предыдущему блоку;
  - **[z** - к началу открытого блока;
  - **]z** - к концу открытого блока;

Источники: раз, два, три.

Переключение раскладки

http://habrahabr.ru/post/175709/

Работа с макросами

Если нужно набрать фразу или предложение несколько раз, существует эффективный подход - механизм записи макросов.

- Команда **qa**, **начнёт запись** в регистр 'a'.
- После набора необходимых действий, команда **'q' закончит запись** макроса.
- **Запуск макроса** через **@a** или **N@a**, для выполнения команд N раз.

Источники: moolenar.ru, habrahabr.ru.

Комментирование блока текста

Вертикальное выделение + множественная вставка: http://stackoverflow.com/a/15588798

Терминал во вкладке

- Открыть терминал в новой вкладке **:tab terminal**
- Нормальный режим в терминале: **Ctrl-w N**
- Возвращение в обычный режим терминала: **i**
Источник

Отладка в VIM'e (GDB)

Описание по ссылке

Менеджер плагинов Vundle

Материалы в сети:

- автоматическая установка менеджера плагинов Vundle (ссылка);
- Vim Vundle Tutorial (ссылка).

Список полезных плагинов:

- Анализатор синтаксических ошибок в программном коде, plugin 'dense-analysis/ale'.
- Работа с Git из-под Vim'a, plugin ' tpope / vim-fugitive'.
- Обзор файловой системы, plugin 'preservim/nerdtree', справка доступна по ссылке.

Полезное:

- Для удаления плагина plugin_name необходимо:
  - удалить строку с названием плагина в ~/.vimrc;
  - вычистить код удалённых плагинов :PluginClean.

ctags в VIM

Список основных команд по ссылке.

- **Ctrl-]** - переход по тегу на реализацию функции, класса и т.д.;
- **Ctrl-o**, **Ctrl-t** - возврат после перехода по тегу.

Автоматическое обновления файла с тегами:

- установка плагина vim-gutentags через Vundle:

```
Plugin 'ludovicchabant/vim-gutentags'
```

- после запуска редактора установка нового плагина:

```
:PluginInstall
```

- настройка плагина:

```
" Plugin 'ludovicchabant/vim-gutentags'
let g:gutentags_add_default_project_roots = 0 " Disable default root markers
let g:gutentags_project_root = ['.gutentags','.vimGutenTags']
let g:gutentags_ctags_exclude = [ '.git', 'build', 'depends', 'docs', '.md',
'.cache', 'tags', '.css', '.vim' ]
```

- использование. В корневом каталоге проекта создать файл с именем '.gutentags'.
- при наличии соответствующей настройки в .vimrc удобно по F2 отображать активный буфер в

NERDTree:

```
silent! map <F2> :NERDTreeFind<CR> " Find directory in NERDTree with current
file
```

Установка ctags:

- для работы плагинов в vim'e необходимо установить утилиту ctags в системе

```
apt install universal-ctags
```

Автодополнение ddc.vim

Настройка **Dark deno-powered completion framework**.
Домашняя страница проекта доступна по ссылке ddc.vim.

- установка окружения Deno:

```
$ curl -fsSL https://deno.land/x/install/install.sh | sh
```

Vim отобразит ошибку при отсутствии исполняемого файла deno:

```
[denops] A 'deno' (g:denops#deno) is not executable. Denops requires
executable Deno.
```

- установка базового плагина **ddc.vim** и зависимости **denops.vim**:

```
Plugin 'Shougo/ddc.vim'        "Dark deno-powered completion framework for
neovim/Vim
Plugin 'vim-denops/denops.vim' "Denops is ecosystem of Vim/Neovim which allows
developers to write plugins in Deno
```

- выбор и установка требуемых компонентов sources, filters и ui;
- установка ddc-ui-native:

```
Plugin 'Shougo/ddc-ui-native'   "/UI/ Native popup menu UI for ddc.vim
```

- установка ddc-source-around:

```
Plugin 'Shougo/ddc-source-around' "/Source/ Around completion for ddc.vim
```

- установка ddc-filter-matcher_head и ddc-filter-sorter_rank:

```
Plugin 'Shougo/ddc-filter-sorter_rank'   "/Filter/ Matched rank order sorter
for ddc.vim
Plugin 'Shougo/ddc-filter-matcher_head'  "/Filter/ Heading matcher for ddc.vim
```

- настройка ddc-ui-native:

```
call ddc#custom#patch_global('ui', 'native')
```

- настройка ddc.vim:

```
call ddc#custom#patch_global('sources', ['around'])
call ddc#custom#patch_global('sourceOptions', {
  \ 'around': { 'mark' : '[A]' },
  \ '_': {
  \   'matchers': ['matcher_head'],
  \   'sorters': ['sorter_rank']},
  \ })
call ddc#enable()
```

—
По материалам:

1. справочного руководства проекта (см. ссылку);
2. статьи «Плагин автодополнения нового поколения ddc.vim» автора ddc.vim (см. ссылку); машинный перевод статьи с Японского языка доступен по ссылке.