

grass.elのぺえじ

```
(´・`・´) んも~
={=}O,
|:~::~:~::~:\, ', '
....し....((. @) w v ww WWw v ww Ww w v www WWWww Ww
w WWWWWWw w w w W w w v w WWw W w w v w WWW
```

何これw w w w w w w

Emacs上で、[ちょっと草植えときますね型言語Grass](#)のプログラムを実行や作成するためのプログラムです。元々は[ひげぼん氏のgrass.scm](#)を参考にして処理系だけ作るつもりだったのが、いつの間にか巨大化w w w w w ラムダ計算の学習にどうぞ。
([スクリーンショット](#))

GNU Emacs 21, 22, 23用です。たぶん、XEmacsでは動作確認していません。Meadowでも大丈夫かもしれませんが、Meadow(牧草地)に草を植えるとかw w w w w w w

質問・ツッコミはこちらのページへどうぞ [Softwareのぺえじ](#)

特徴w w w w w w w

- Grassのプログラムを実行できます(範囲を選択して実行するとかもできます)
- 分かりやすい形式でコードを書いて、それを普通のGrassコードに変換できます(一応、逆変換も可能)
- メジャーモード(grass-mode)とマイナーモード(grass-minor-mode)が使えるようになります
- 豊富なサンプルコード付き
- プログラム実行時の日本語出力に対応(シフトJISコードのみ)

ダウンロードw w w w w w w

 [grass.el-0.1.8.tar.gz](#)(2008.09.06 更新)

version 0.1.7 からの変更点

- オプションgrass-allow-consecutive-v-separatorsおよびgrass-conversion-buffer-restorableを追加
 - サンプルコードのgrass-sample-hello (Hello, world!) を差し替え
 - 評価ウィンドウや変換ウィンドウを開く時に既存のバッファをkillせず内容を消去するように変更
 - 文字列中からリスト形式のコード先頭を検索するのがうまくいかない事がある不具合を修正
- ([更新履歴はこちら](#))

つかいかたw w w w w w w

目次

- [チュートリアル](#)
- [使用できるコマンド](#)
- [メジャーモードとマイナーモード](#)
- [サンプルコードの簡単な説明](#)

チュートリアル

準備

まずEmacsでgrass.elを開き、M-x eval-current-bufferします。

Emacsの起動時にロードしたい場合は、grass.elをEmacsのロードパスの通ったディレクトリに保存し、.emacsファイルに以下の内

容を追加しておきます。

```
(require 'grass)
```

サンプルを実行してみる

scratchバッファで

```
grass-sample-hello
```

とタイプし、C-jを押してみてください。Grassのコードが表示されますね(中身は[これ](#))。

まず、このコードの入った文字列変数を直接実行してみます。

```
(grass-eval-string grass-sample-hello)
```

とタイプしてC-jを押すと、ウィンドウが分割されて「Hello, world!」と表示されます。

今度は、マイナーモードを使って実行してみましょう。

「C-c g m」または「M-x grass-minor-mode」とタイプすると、grass-minor-modeがオンになります。grassのコードが色分け表示されるはずですが、

カレントバッファにgrass-sample-helloの内容が表示された状態で「C-c g e」または「M-x grass-eval」とタイプすると、先程と同様に「Hello, world」と表示されます。

ただし、バッファ内の他の部分に「w」「W」「v」などの文字があると、それらもGrassのコードとして認識してしまい、うまくいかないかもしれません。

その場合は、実行したい部分を領域選択してから「C-c g e」とタイプしてください。

例えば、このマイナーモードをw3m-modeと一緒に使うと、インターネット上にあるGrassのプログラムを直接実行できます。

w3mの再描画などで文字の色が消えてしまった場合は、「C-c g f」で色付けをやり直して下さい。

Grassのコードを生成する

先程のgrass-sample-helloの内容が表示された状態のバッファで「C-c g l」または「M-x grass-listify」とタイプすると、grass-sample-helloがリストに変換されたものが表示されます。grass.elがコードを実行する時には、パーサでこのような内部形式のリストに変換してから評価します。人間にとってもこの方が元のGrassコードより理解しやすいと思います。grass.elでは、これを少し拡張した独自形式でコードを記述してから、Grassのコードに変換することができます。

次のファイルをダウンロードして開いてみてください。

 [grass-sample-hello.el](#)

関数、引数、関数適用に名前を付けて参照しているのが分かると思います。Emacs Lispにシンボルとして見なされるような名前を付ける必要があります。既に定義されている名前を再定義した場合、後から定義したものが優先されます。

ここで、「C-c g p」または「M-x grass-plant」とタイプすると、Grassのコードに変換され、他のウィンドウに表示されます。この時、バッファはgrass-modeというメジャーモードになり、色分けして表示されます。「C-c g e」で実行できます。

リスト形式のコードを直接実行したい場合は、「C-c g E」または「M-x grass-eval-list」とタイプします。

使用できるコマンド

基本的なコマンドは、grass-eval、grass-eval-list、grass-plant、grass-listifyの4つです。他に、これらのコマンドの変種と、メジャーモードとマイナーモード関連のコマンドがあります。

grass-eval

Grassプログラムを実行します。カレントバッファが対象になります。インタラクティブな呼び出しで範囲選択している場合は、その範囲のみを評価します。結果は他のウィンドウに表示します。ウィンドウがなければ、フレームを分割して表示します。

grass-evalには以下の派生コマンドがあります。

- grass-eval-string string : 文字列を評価します
- grass-eval-region beg end : 指定範囲を評価します
- grass-eval-buffer buffer : バッファを評価します

また、これらの出力を文字列として返す関数もあります。

- grass-eval-string-to-string string
- grass-eval-region-to-string beg end

grass-eval-buffer-to-string buffer

■

grass-eval-list

リスト形式のGrassのプログラムを実行します。Emacs Lispのリストの形で記述するものです。カレントバッファが対象になります。インタラクティブな呼び出しで範囲選択している場合は、その範囲のみを評価します。結果は他のウインドウに表示します。ウインドウがなければ、フレームを分割して表示します。

grass-eval-listには以下の派生コマンドがあります。

- grass-eval-list-string string : 文字列を評価します
- grass-eval-list-in-region beg end : 指定範囲を評価します
- grass-eval-list-in-buffer buffer : バッファを評価します
- grass-eval-code list : リストを評価します

また、これらの出力を文字列として返す関数もあります。

- grass-eval-list-string-to-string string
- grass-eval-list-in-region-to-string beg end
- grass-eval-list-in-buffer-to-string buffer
- grass-eval-code-to-string list

grass-plant

リスト形式で記述したプログラムから標準のGrassプログラムを生成します。カレントバッファが対象になります。インタラクティブな呼び出しで範囲選択している場合は、その範囲のみを変換します。結果は他のウインドウに表示します。ウインドウがなければ、フレームを分割して表示します。

grass-plantには以下の派生コマンドがあります。

- grass-plant-string string : 文字列を変換します
- grass-plant-region beg end : 指定範囲を変換します

- grass-plant-buffer buffer : バッファを変換します
- grass-plant-code list : リストを変換します

また、これらの出力を文字列として返す関数もあります。

- grass-plant-string-to-string string
- grass-plant-region-to-string beg end
- grass-plant-buffer-to-string buffer
- grass-plant-code-to-string list

grass-listify

grass-plantの逆変換、つまりGrassプログラムからリスト形式のプログラムに変換します。カレントバッファが対象になります。インタラクティブな呼び出しで範囲選択している場合は、その範囲のみを変換します。結果は他のウインドウに表示します。ウインドウがなければ、フレームを分割して表示します。

grass-listifyには以下の派生コマンドがあります。

- grass-listify-string string : 文字列を変換します
- grass-listify-region beg end : 指定範囲を変換します
- grass-listify-buffer buffer : バッファを変換します

また、これらの出力を文字列として返す関数もあります。

- grass-listify-string-to-string string
- grass-listify-region-to-string beg end
- grass-listify-buffer-to-string buffer

メジャーモードとマイナーモード

たいした機能はありませんが、メジャーモードとマイナーモードを作ってみました。

「M-x grass-mode」でメジャーモードを開始、「C-c g m」または「M-x grass-minor-mode」でマイナーモードをトグルできます。

grass-sample-infinity

無限に草植えときますねwWWwwwwWWww

Grassの公式サンプルです。無限に「w」を出力し続けます。放置するとEmacsがクラッシュするかもしれないので、C-gで止めて下さいw

更新履歴

2008.09.06 version 0.1.8

- オプションgrass-allow-consecutive-v-separatorsおよびgrass-conversion-buffer-restorableを追加
- サンプルコードのgrass-sample-hello (Hello, world!) を差し替え
- 評価ウィンドウや変換ウィンドウを開く時に既存のバッファをkillせず内容を消去するように変更
- 文字列中からリスト形式のコード先頭を検索するのがうまくいかない事がある不具合を修正

2008.09.01 version 0.1.7

- 関数適用が関数定義の中にしか書けなかったバグを修正

2008.08.03 version 0.1.6

- OutプリミティブでCRをLFに変換していたのを、Inプリミティブで変換するように変更
- マイナーモード変数をバッファローカルにしていなかったバグを修正

2008.08.02 version 0.1.5

- grass-plantやgrass-eval-listでコードの先頭位置を探す時、初めの空行とコメント行を無視するようにした

2008.07.27 version 0.1.4

- emacs-lisp-modeのキーマップを上書きせずgrass-mode-mapを使うように変更

2008.07.25 version 0.1.3

- 色付けをやり直すキーバインドを追加
- Emacs21, 22の文字出力で、不正なシフトJISコードの時にエラーで止まらないように修正

2008.07.24 version 0.1.2

- Emacs21, 22でシフトJISコードの文字出力ができなかったのを修正

2008.07.24 version 0.1.1

- OutプリミティブをシフトJISコードの文字出力に対応
- リスト形式のコードの構文チェックを修正

2008.07.23 version 0.1.0

- 初公開
-