平成時代に憧れてた
あのエディタを実装してみる

Implement the text editor I admired in Heisei era

新元号決定！平成最後のLT大会＆PARTY

2019年4月30日 #engineers_lt

お前誰よ

- うさみけんた（@tadsan）/ Zonu.EXE
  - GitHub/Packagistでは id: zonuexe
- ピクシブ株式会社 サービスプラットフォーム事業部
- Emacs Lisper, PHPer
  - Emacs PHP Modeのメンテナ引き継ぎました
  - 好きなリスプはEmacs Lispです
- Qiitaに記事を書いたり変なコメントしてるよ

pixiv

時は遡る

Laravel JP Conference 直後

4月
30

新元号決定！平成最後のLT大会＆PARTY

LTと共に新元号になった私

ハッシュタグ ： #engineers_lt

フォロー参加者

フォローブックマーク

募集内容	**前売券** 3000円（前払い）	先着順 9/40人
	スタッフ券 3000円（前払い）	先着順 10/10人

グループ　メンバーになる

エンジニアの登壇を応援する会

イベント数　12回
メンバー数　634人

開催前

2019/04/30(火)
20:00 ～ 2019/05/01(水) 02:00

Googleカレンダー　icsファイル

このイベントに参加できます

受付票を見る

※受付や入場方法は主催者の案内に従ってください。

申し込みキャンセル

開催日時が重複しているイベントに申し込んでいる場合、このイベントには申し込むことができません

募集期間

何か話さな

いと……

5月
8

東京Emacs勉強会 端午の節句

Emacs 26.2リリース記念

フォロー参加者

フォローブックマーク

募集内容		
	発表する (15分程度) 無料	先着順 6/6人
	設営・受付スタッフ 無料	先着順 3/4人
	一般参加 無料	先着順 24/40人

グループ

メンバーです

東京Emacs勉強会

イベント数 2回
メンバー数 63人

開催前

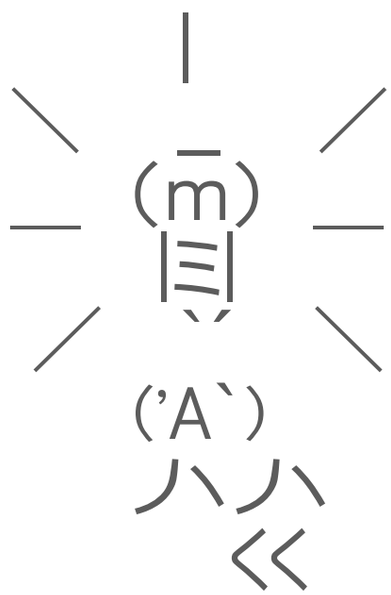
2019/05/08(水)
19:00 ～ 22:00

Googleカレンダー　icsファイル

このイベントに参加できます

受付票を見る

※受付や入場方法は主催者の案内に従ってください。



そうだ、テキストエディタ作ろう

最初に考えた内容

秀丸上にEmacsを実現しよう

先行実装

秀Macs

秀丸エディタ for Windows を Emacs ライクなキー・アサインに変える

ユーザーの評価(0人)：
コメント：0件
コメントを見る

■関連キーワード
エディタ ,キー ,ライク ,アサイン ,Emacs

ダウンロード

ソフト名： 秀Macs　2.01

ファイル： hmacs201.lzh / 18,425Bytes / 1996.06.22

昨晩手を
つけてみた

もう20年アップデートされてないし、現代風に書き直してみたら楽しいのでは？？？

めちゃくちゃ手間が
かかるし予想以上に
おもしろくない

はてさて

どうしよう

第2案発動
（本当に手をつける
つもりはなかった）

平成時代に憧れてた
あのエディタを実装してみる

Implement the text editor I admired in Heisei era

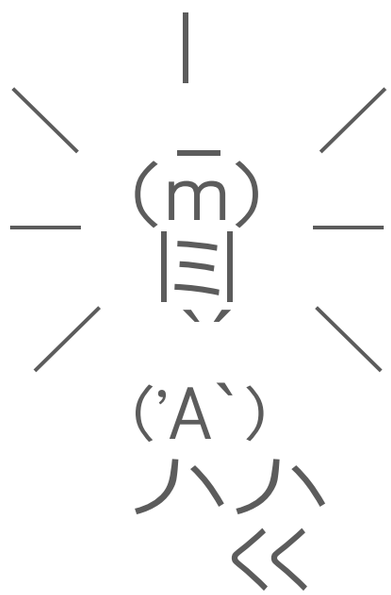
新元号決定！平成最後のLT大会＆PARTY

2019年4月30日 #engineers_lt

平成時代に憧れてた
~~あの~~エディタを実装してみる

Implement a text editor I admired in Heisei era

新元号決定！平成最後のLT大会＆PARTY
2019年4月30日 #engineers_lt



そうだ、PHPでエディタ作ろう

ということで今朝
起きてから実装を
始めました

そもそもテキストエディタとは何か

歴史的にはed, ex,
MS-DOSのEDLINのような
ラインエディタがあった

ファイルを部分的にプリントして編集していくスタイル

（今日「プリント」は画面上に文字を出力することを意味するが、1970年代前半ではテレタイプ端末で本当に印刷されてたらしい）

話がそれた

本質的な
仕事

（ファイルを読み込む）

↓

ユーザにごによごによさせる

↓

ファイルに書き戻す

僕にも

できそう

どうやって
「こだよこだよ」
させるかが問題

方針

PHPでやる

端末でやる

定石

ncursesを使う

やったことない

簡単な方法

画面を全部消す
↓
内容を全部出力
↓
無限に繰り返し…

簡単じゃん

PHPでやるには

どうすればいいか

概ねいい

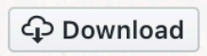
感じにできる

基本はWebアプリケーションフレームワークの実装と同じ

HTTPじゃなくて
端末の入力に対し
て処理していく

Hack book of Hoa\Console



The terminal is a very powerful interface that is based on multiple concepts. Hoa\Console allows to write tools that are adapted to this kind of environment.

Table of contents

1 Introduction
2 Window
    1 Size and position
    2 Title and label
    3 Interact with the content
3 Cursor
    1 Moving
    2 Content
    3 Style
    4 Sound
4 Readline
    1 Basic usage
    2 Shortcuts
    3 Auto-completion

画面を消す

||

Hoa\Console\Console::clear()

文字列を貯める バッファの実装は とことん楽をする

余談:エディタはテキストファイルの途中に文字を消したり追加したりするので、文字列（＝文字の配列）と相性が悪い（途中挿入・削除コスト高）

そんなわけで実装していきましょう

```php
$keymap = Zned\Keymap::make([
    "\x03" => new Zned\Command\ForceExit,
    "\x14" => new Zned\Command\ScreenRefresh,
]);
```

```php
$ui_terminal = new Zned\DumbTerminal(
    new Zned\Hoa\Cursor,
    new Zned\Hoa\Console
);

Zned\Application::run(STDIN, $argv, $keymap, $ui_terminal);
```

```
public static function run(
    $stdin, array $args, Keymap $keymap,
    UserInterfaceInterface $ui
): void
{

    $self = array_shift($args);
    $file = array_shift($args);
```

```php
$buffer = new StreamBuffer(fopen('php://temp', 'rw'));

if ($file !== null) {
    $buffer->insert(file_get_contents($file));
    $buffer = new FileRelatedBuffer($buffer, $file);
}

$app = new static;
$app->keymap = $keymap;
$app->ui = $ui;
$app->registerBuffer($buffer);
$app->display();
```

```php
private function enableTtyRawMode()
{
    $this->last_tty_mode = trim(`stty -g`);

    register_shutdown_function([$this, 'restoreTtyMode']);

    system('stty raw -echo');
}

public function restoreTtyMode()
{
    if ($this->last_tty_mode !== null) {
        system(sprintf('stty %f', escapeshellarg($this->last_tty_mode)));
    }
}
```

```php
$read = [$stdin];
$write = null;
$except = null;

while (true) {
    if (stream_select($read, $write, $except, 3600)) {
        $input = fread($stdin, 1024);

        $app->dispatchKeys($input);
    }

    $app->display();
}
```

実装しなかったけど
面倒くさかったところ

カーソル

UTF-8は

可変長

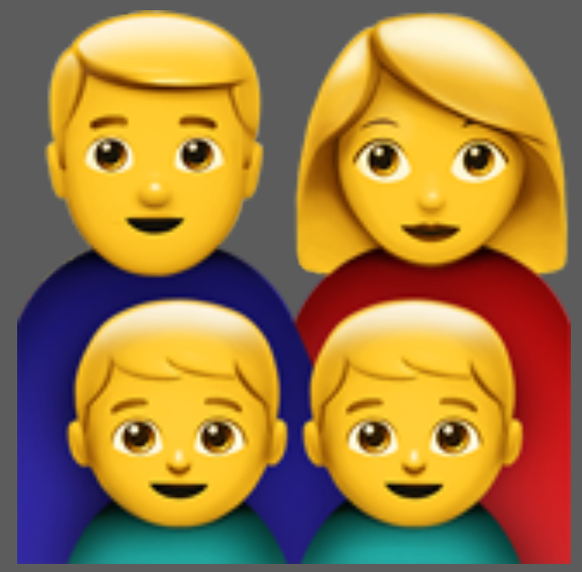
カーソルを移動させるためには一文字ずつ表示幅を調べないといけない

「カーソル」位置を
文字数で持つか
バイト数で持つか

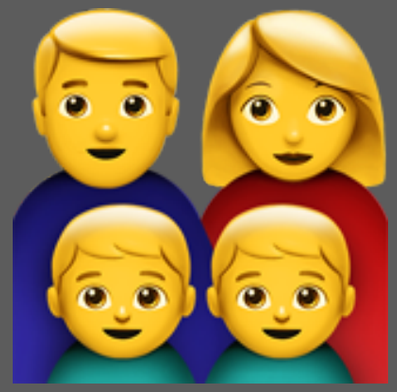
文字列をUTF-32で持てば固定長になるのでは？？？

だめです

Unicodeの書記素クラスタ



この絵文字のバイト数

strlen('  ');

この絵文字のバイト数

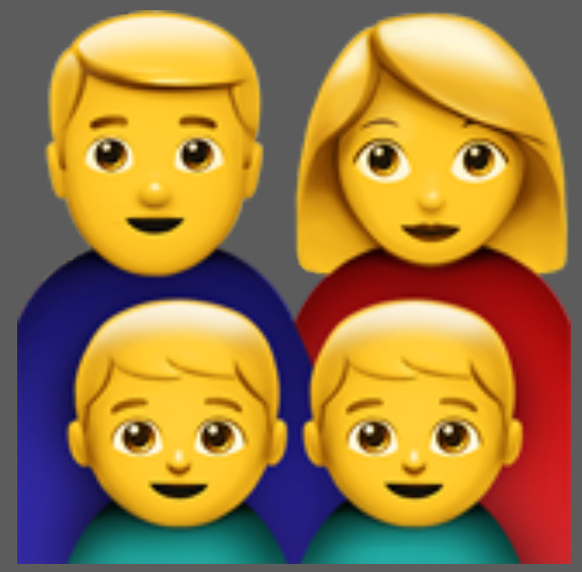
```
strlen('👨‍👩‍👦');
// => 25
```

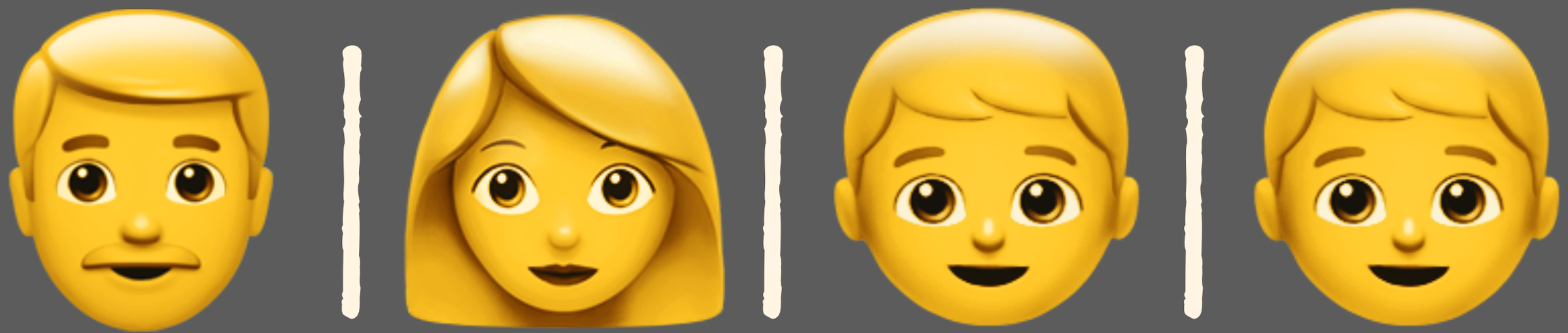
この絵文字のUTF-8文字数

```
mb_strlen('👨‍👩‍👦‍👦', 'UTF-8');
```

この絵文字のUTF-8文字数

```
mb_strlen('👨‍👩‍👦', 'UTF-8');
// => 7
```





Zero Width Joiner

Unicodeにおいて、
1バイト
1コードポイント
人間に見える文字

まったく

一致しません

エディタは完成
してないです！！

でも行きあたりばったり
で設計しながら
実装していくのは楽しい

私からは

異常です