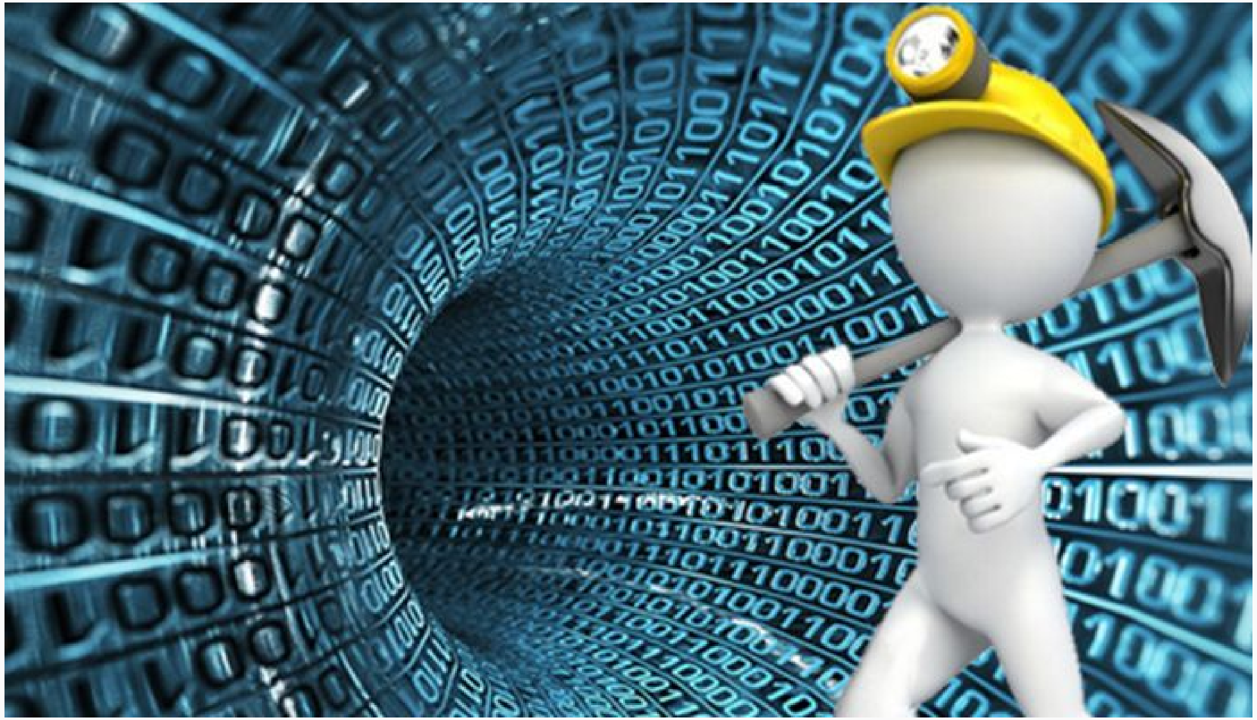
Threads em PHP: mito ou verdade?

Jonathan Szablevski
Conferência PHPRS 2017

Jonathan Szablevski

- Mora em Gramado-RS;
- Desenvolvedor Web há 6 anos;
- Cursa Ciência da Computação na Universidade Feevale.

Problema



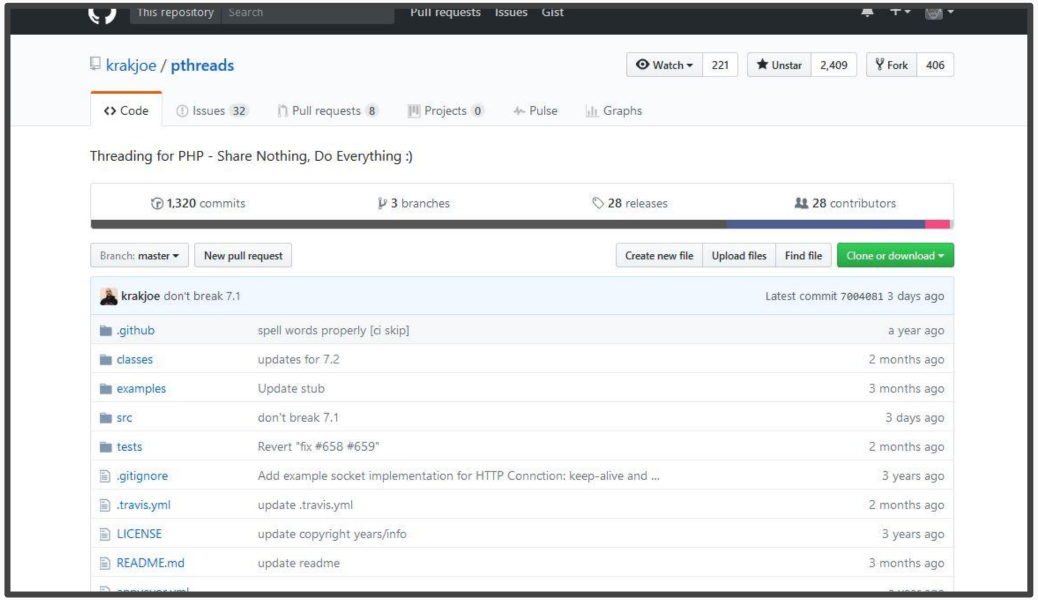
Fonte:
https://www.linkedin.com/pulse/what-web-scraping-everthing-you-wanted-know-more-jon-thralow?articleId=8989924668317778253

Problema

Fonte: https://plus.google.com/+GwenythCook/posts/HKvsDP8wiSe

pthreads

API que fornece diversas ferramentas necessárias para a execução de multithreading em PHP.



Threading for PHP - Share Nothing, Do Everything :)

krakjoe don't break 7.1 — Latest commit 7004081 3 days ago

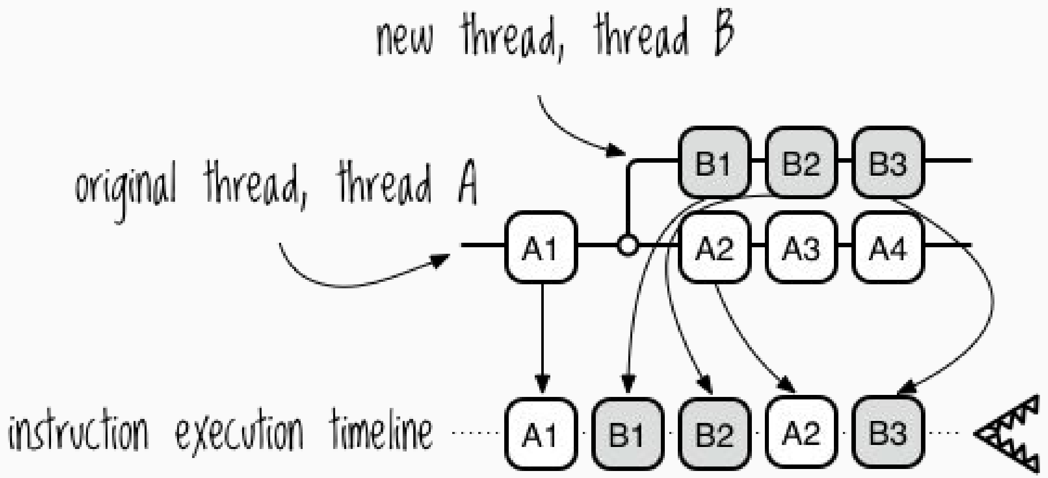
.github	spell words properly [ci skip]	a year ago
classes	updates for 7.2	2 months ago
examples	Update stub	3 months ago
src	don't break 7.1	3 days ago
tests	Revert "fix #658 #659"	2 months ago
.gitignore	Add example socket implementation for HTTP Connction: keep-alive and ...	3 years ago
.travis.yml	update .travis.yml	2 months ago
LICENSE	update copyright years/info	3 years ago
README.md	update readme	3 months ago

Threads

O que são threads

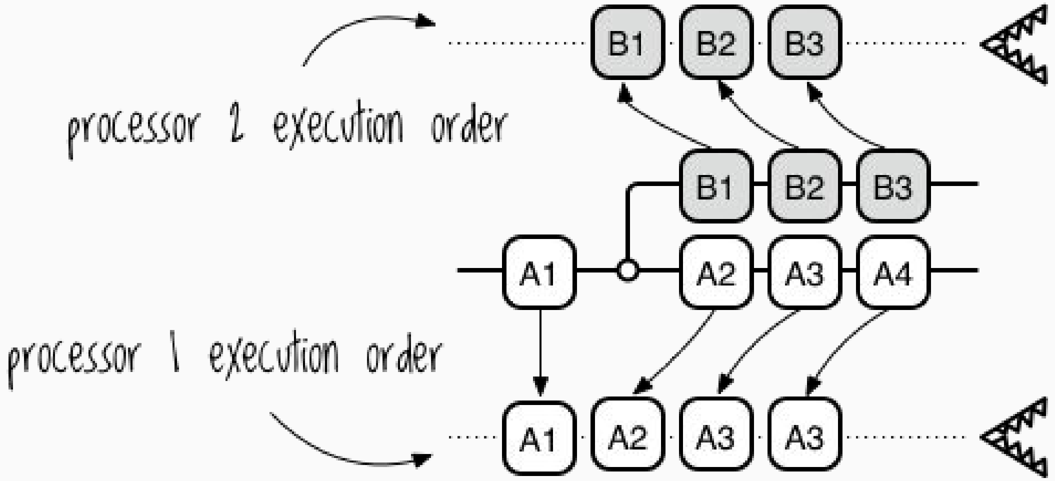
- Uma thread é um componente de um processo;
- Múltiplas threads podem existir dentro de um mesmo processo;
- Threads podem compartilhar recursos, como memória;
- Uma thread pode gerar outras threads;
- Threads podem ser executadas de maneira concorrente e paralela;
- ...

Concorrência



Fonte: http://www.braveclojure.com/concurrency/

Paralelismo



Fonte: http://www.braveclojure.com/concurrency/

Threads na prática

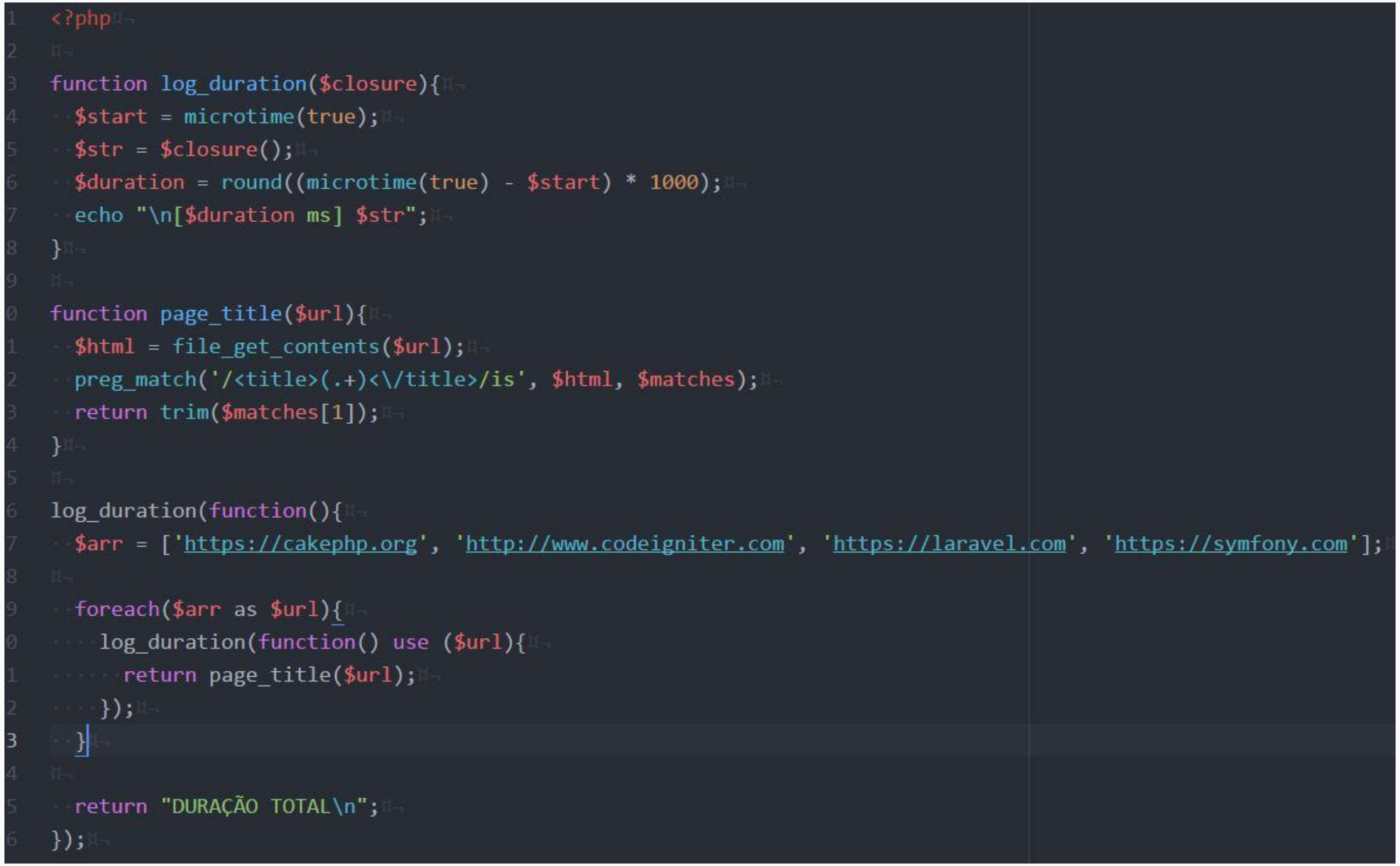
Fonte: https://www.reddit.com/r/aww/comments/2oagj8/multithreaded_programming_theory_and_practice/

Alternativas em PHP

- Funcionalidade *async* do Hack;
- Função *pcntl_fork*;
- Forking *curl/popen/exec*...

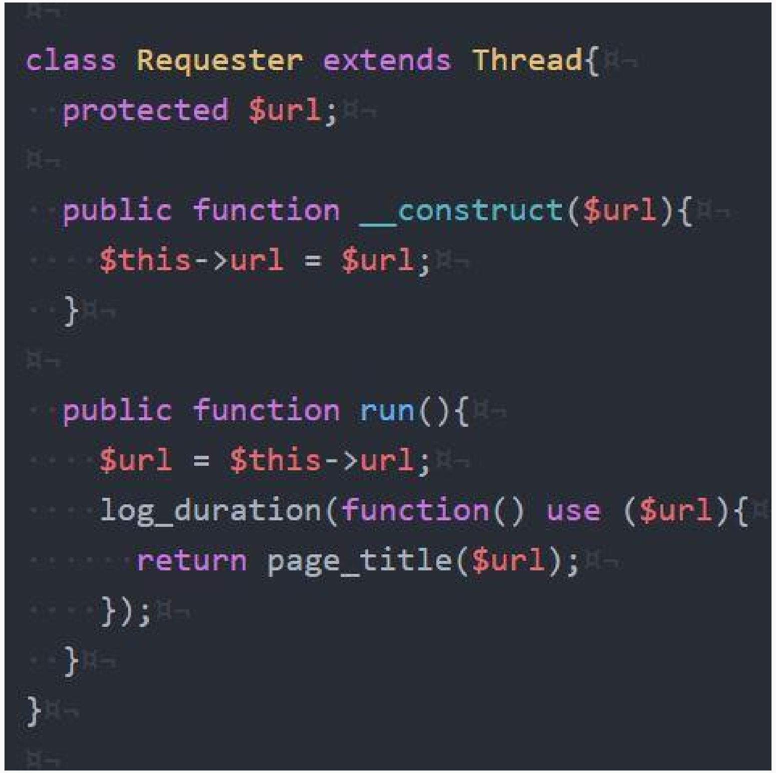
pthreads

Exemplo prático - Sem pthreads

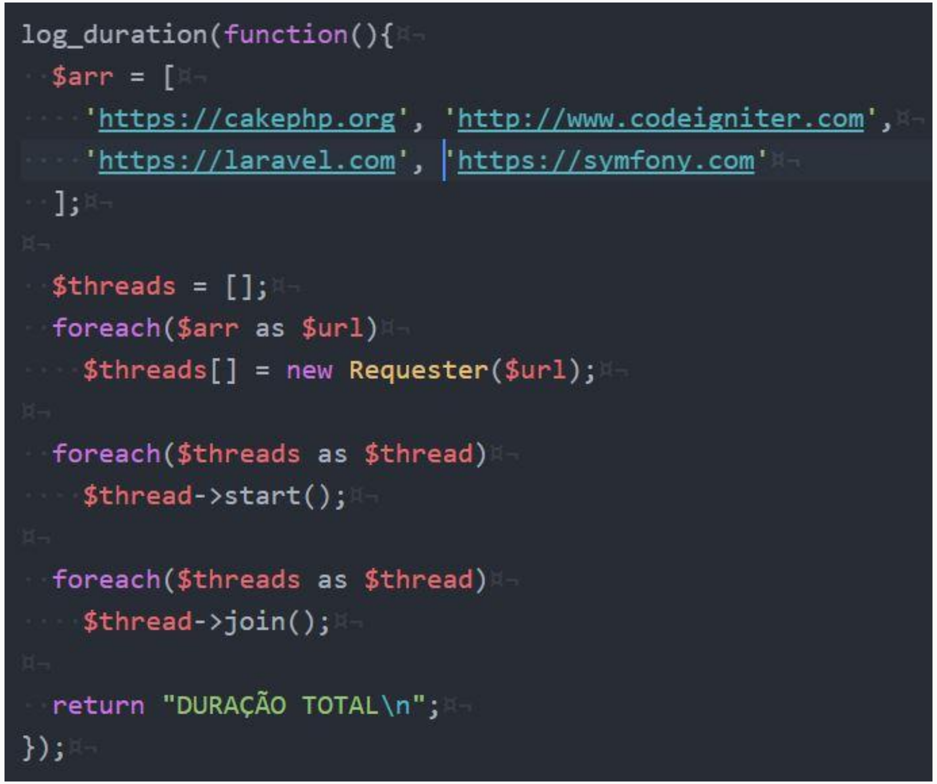


```php
<?php

function log_duration($closure){
  $start = microtime(true);
  $str = $closure();
  $duration = round((microtime(true) - $start) * 1000);
  echo "\n[$duration ms] $str";
}

function page_title($url){
  $html = file_get_contents($url);
  preg_match('/<title>(.+)<\/title>/is', $html, $matches);
  return trim($matches[1]);
}

log_duration(function(){
  $arr = ['https://cakephp.org', 'http://www.codeigniter.com', 'https://laravel.com', 'https://symfony.com'];

  foreach($arr as $url){
    log_duration(function() use ($url){
      return page_title($url);
    });
  }

  return "DURAÇÃO TOTAL\n";
});
```

Exemplo prático - Com pthreads



```php
class Requester extends Thread{
  protected $url;

  public function __construct($url){
    $this->url = $url;
  }

  public function run(){
    $url = $this->url;
    log_duration(function() use ($url){
      return page_title($url);
    });
  }
}
```



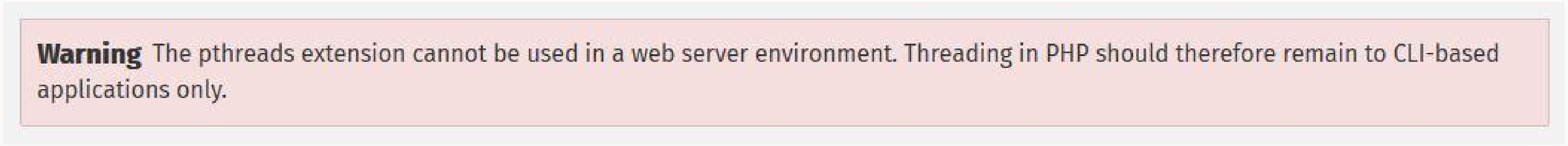
```php
log_duration(function(){
  $arr = [
    'https://cakephp.org', 'http://www.codeigniter.com',
    'https://laravel.com', 'https://symfony.com'
  ];

  $threads = [];
  foreach($arr as $url)
    $threads[] = new Requester($url);

  foreach($threads as $thread)
    $thread->start();

  foreach($threads as $thread)
    $thread->join();

  return "DURAÇÃO TOTAL\n";
});
```

pthreads

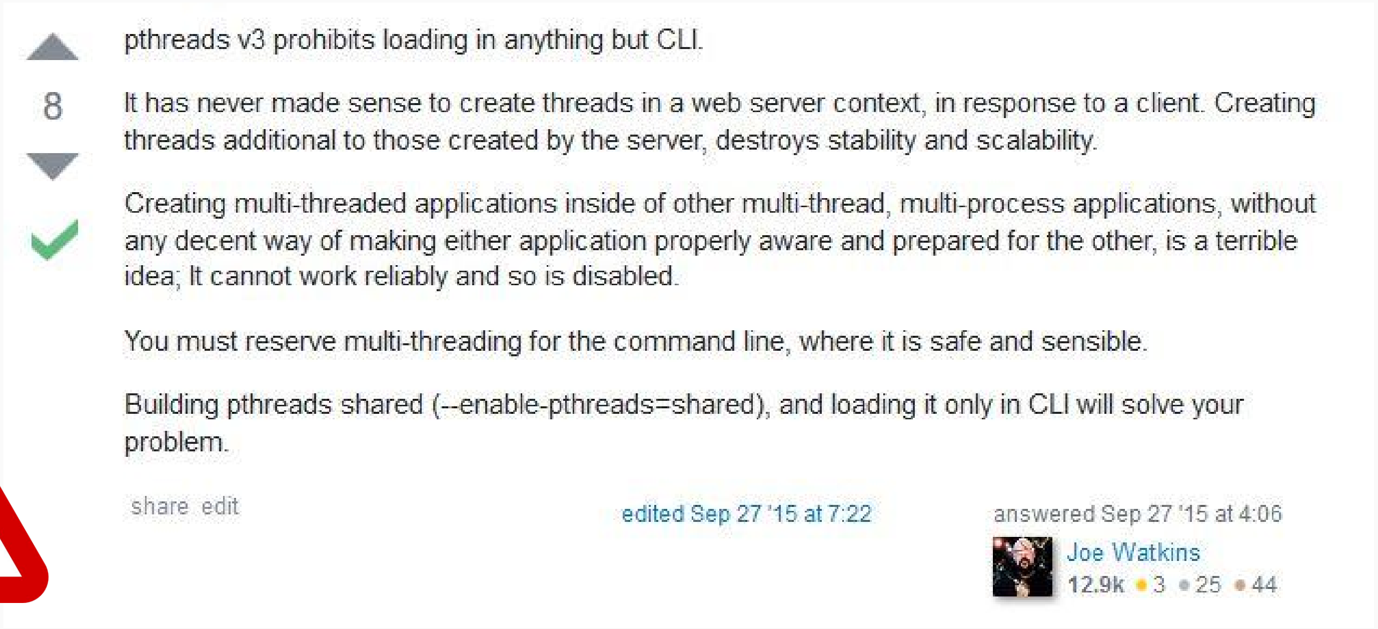
- Provê funcionalidades que permitem multithreading em PHP;
- Projeto criado em 2013 por Joe Watkins;
- Baseado no padrão POSIX Threads;
- API orientada a objetos;
- Compatível com PHP7 a partir da versão 3.

Fonte: https://github.com/krakjoe

pthreads



Warning The pthreads extension cannot be used in a web server environment. Threading in PHP should therefore remain to CLI-based applications only.



pthreads v3 prohibits loading in anything but CLI.

It has never made sense to create threads in a web server context, in response to a client. Creating threads additional to those created by the server, destroys stability and scalability.

Creating multi-threaded applications inside of other multi-thread, multi-process applications, without any decent way of making either application properly aware and prepared for the other, is a terrible idea; It cannot work reliably and so is disabled.

You must reserve multi-threading for the command line, where it is safe and sensible.

Building pthreads shared (--enable-pthreads=shared), and loading it only in CLI will solve your problem.

Fonte: http://stackoverflow.com/a/32804464/6794792

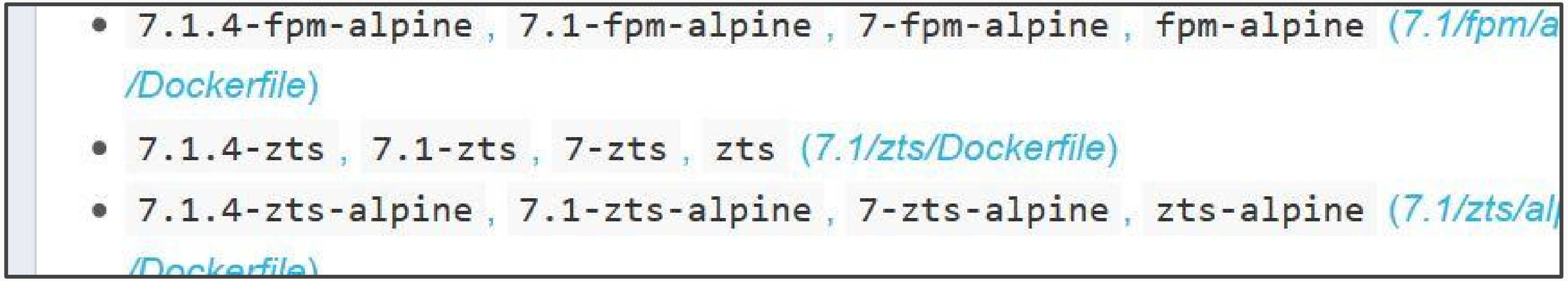
Instalação

Instalação

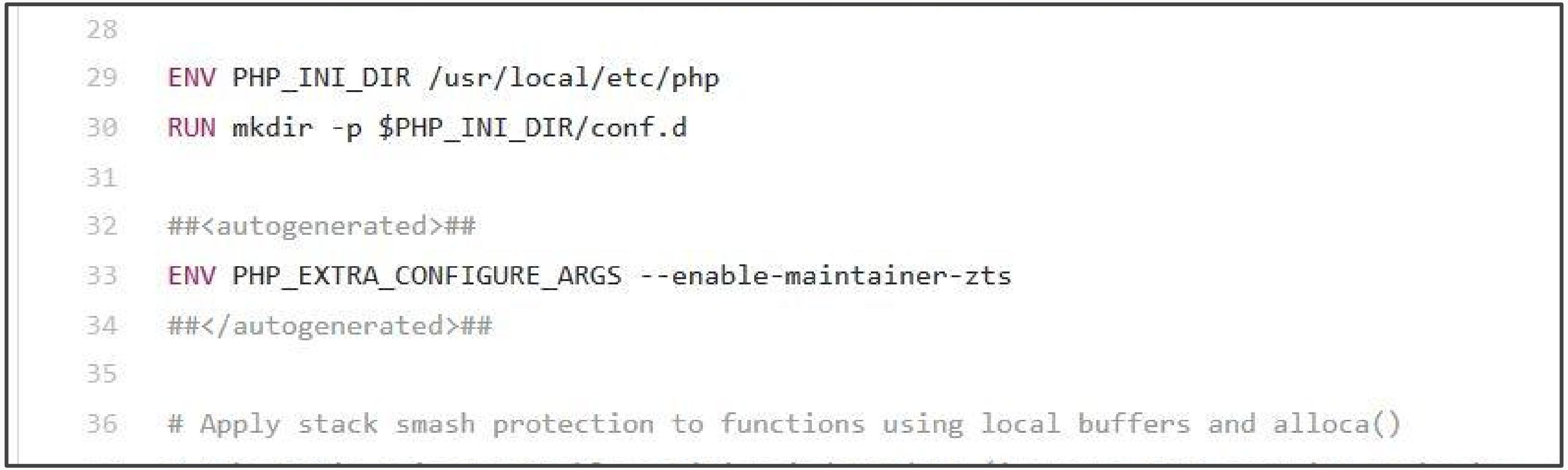
Requisitos

- Suporte para POSIX Threads;
- PHP7;
- ZTS Habilitado (Zend Thread Safety).

Instalação

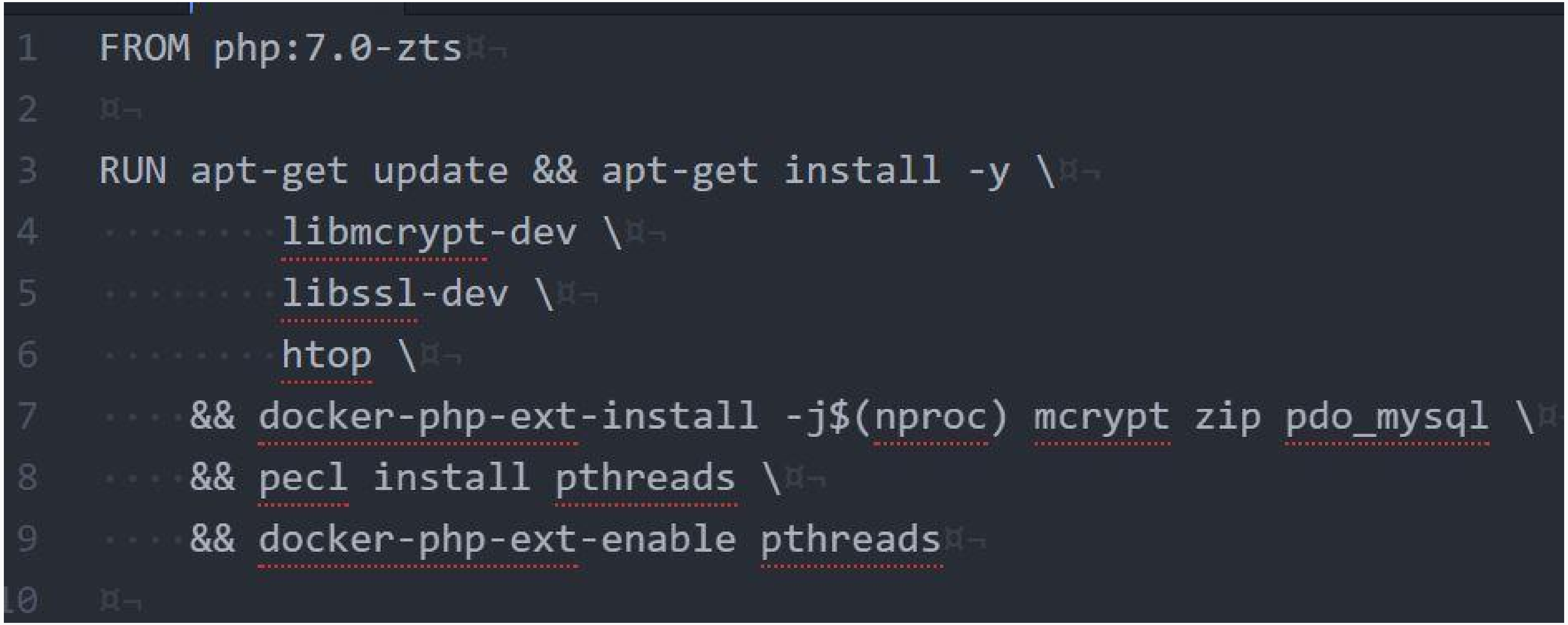


- `7.1.4-fpm-alpine` , `7.1-fpm-alpine` , `7-fpm-alpine` , `fpm-alpine` (7.1/fpm/a /Dockerfile)
- `7.1.4-zts` , `7.1-zts` , `7-zts` , `zts` (7.1/zts/Dockerfile)
- `7.1.4-zts-alpine` , `7.1-zts-alpine` , `7-zts-alpine` , `zts-alpine` (7.1/zts/al /Dockerfile)



```

ENV PHP_INI_DIR /usr/local/etc/php
RUN mkdir -p $PHP_INI_DIR/conf.d

##<autogenerated>##
ENV PHP_EXTRA_CONFIGURE_ARGS --enable-maintainer-zts
##</autogenerated>##

# Apply stack smash protection to functions using local buffers and alloca()
```

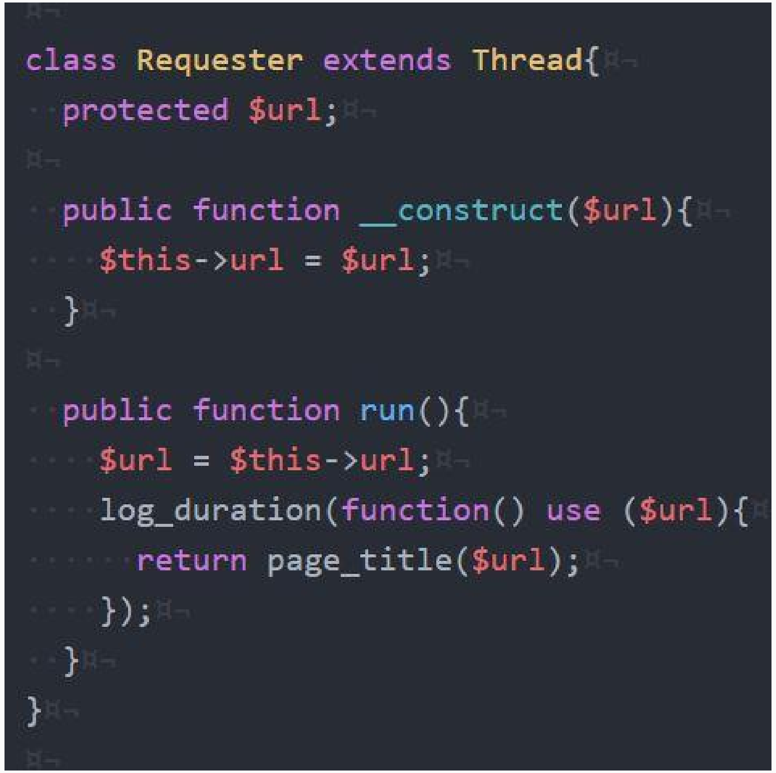
Instalação



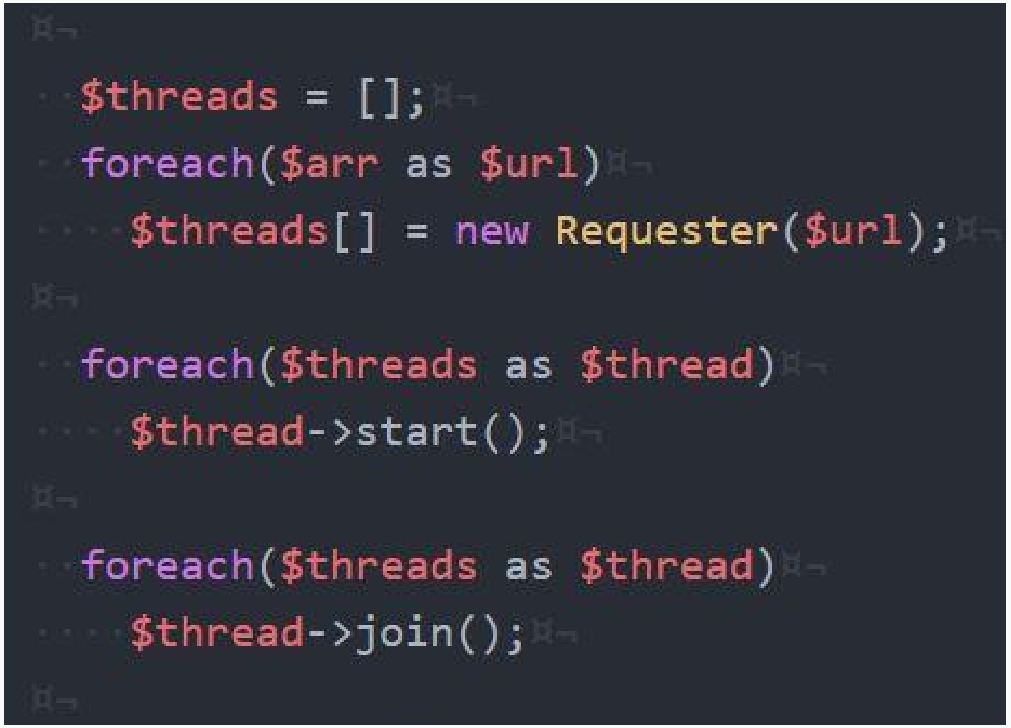
```
FROM php:7.0-zts

RUN apt-get update && apt-get install -y \
        libmcrypt-dev \
        libssl-dev \
        htop \
    && docker-php-ext-install -j$(nproc) mcrypt zip pdo_mysql \
    && pecl install pthreads \
    && docker-php-ext-enable pthreads

```

Funcionalidades

Classe Thread

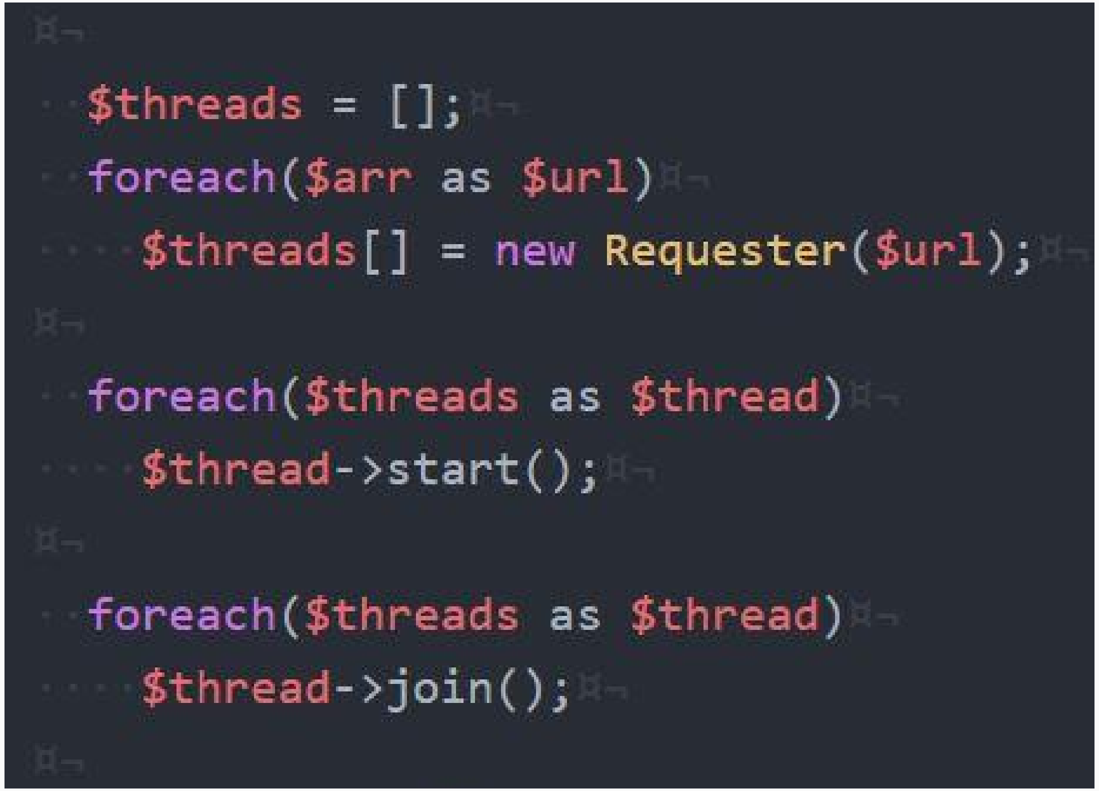


```php
class Requester extends Thread{
  protected $url;

  public function __construct($url){
    $this->url = $url;
  }

  public function run(){
    $url = $this->url;
    log_duration(function() use ($url){
      return page_title($url);
    });
  }
}
```



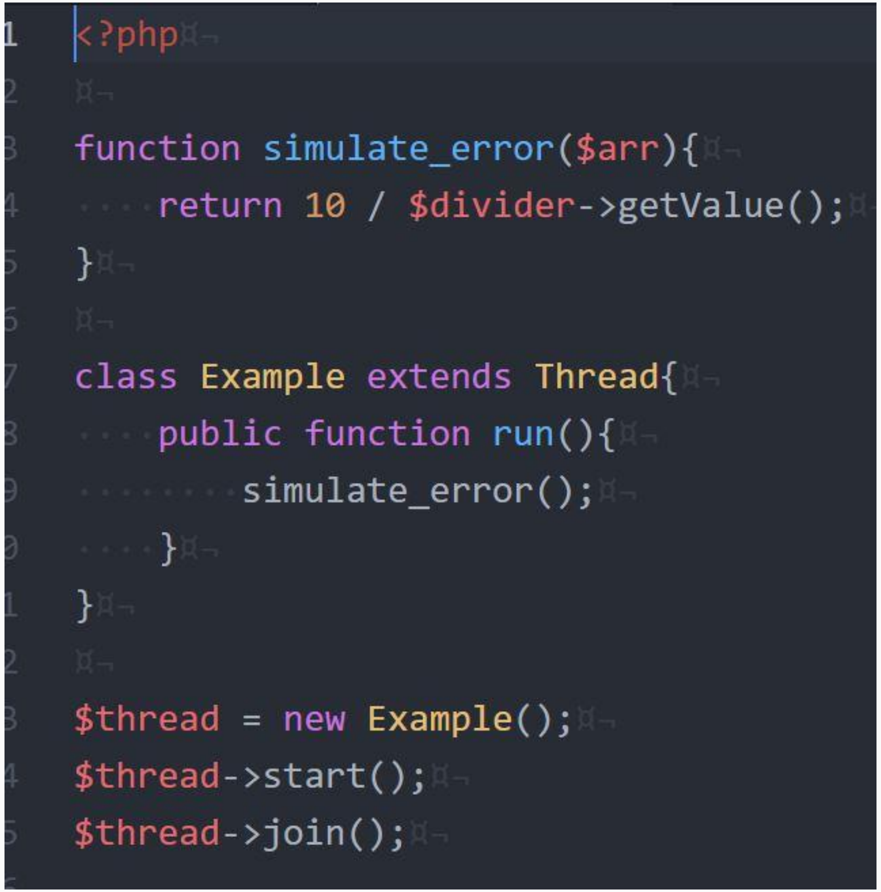
```php
  $threads = [];
  foreach($arr as $url)
    $threads[] = new Requester($url);

  foreach($threads as $thread)
    $thread->start();

  foreach($threads as $thread)
    $thread->join();
```

Método *join*

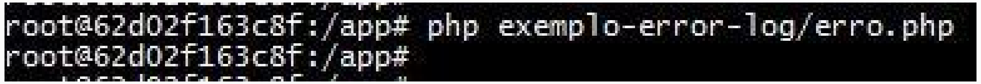


```
  $threads = [];
  foreach($arr as $url)
    $threads[] = new Requester($url);

  foreach($threads as $thread)
    $thread->start();

  foreach($threads as $thread)
    $thread->join();

```

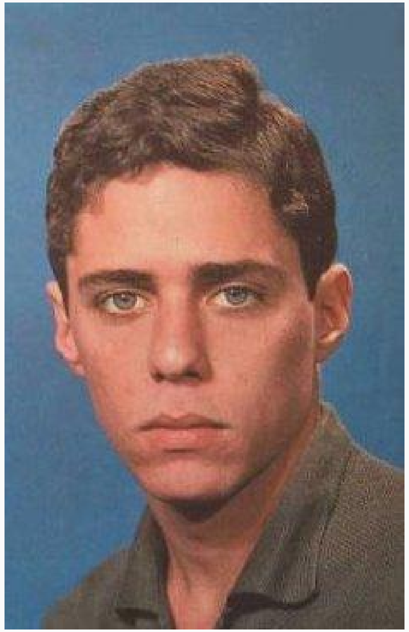
Log de erros no método *run*



```php
<?php

function simulate_error($arr){
    return 10 / $divider->getValue();
}

class Example extends Thread{
    public function run(){
        simulate_error();
    }
}

$thread = new Example();
$thread->start();
$thread->join();
```

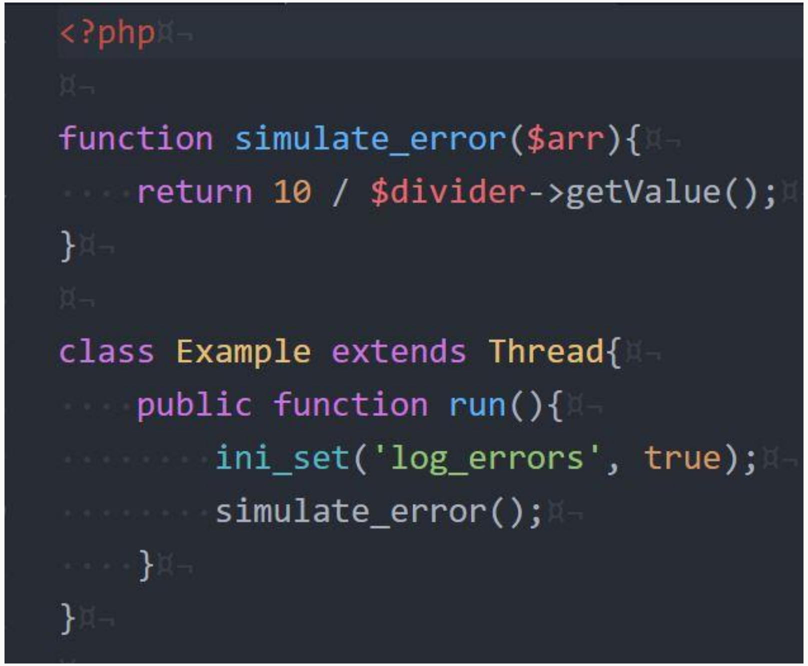


```
root@62d02f163c8f:/app# php exemplo-error-log/erro.php
root@62d02f163c8f:/app#
```

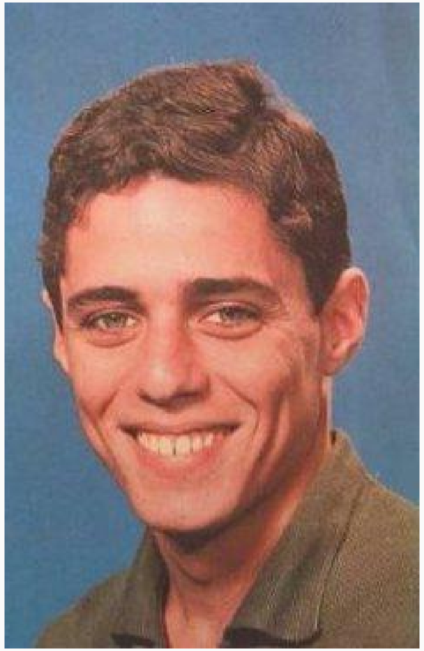


Fonte: https://memesuper.com/categories/view/4eeceefb0993e55aa00e74414f43a035ec14ea4a/meme-feliz-e-triste-ao-mesmo-tempo.html

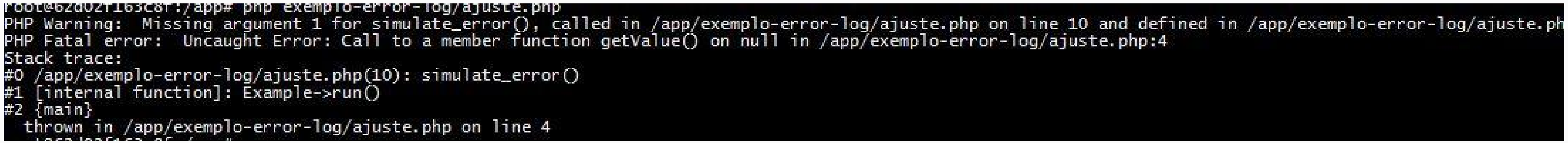
Log de erros no método *run*



```php
<?php

function simulate_error($arr){
    return 10 / $divider->getValue();
}

class Example extends Thread{
    public function run(){
        ini_set('log_errors', true);
        simulate_error();
    }
}
```

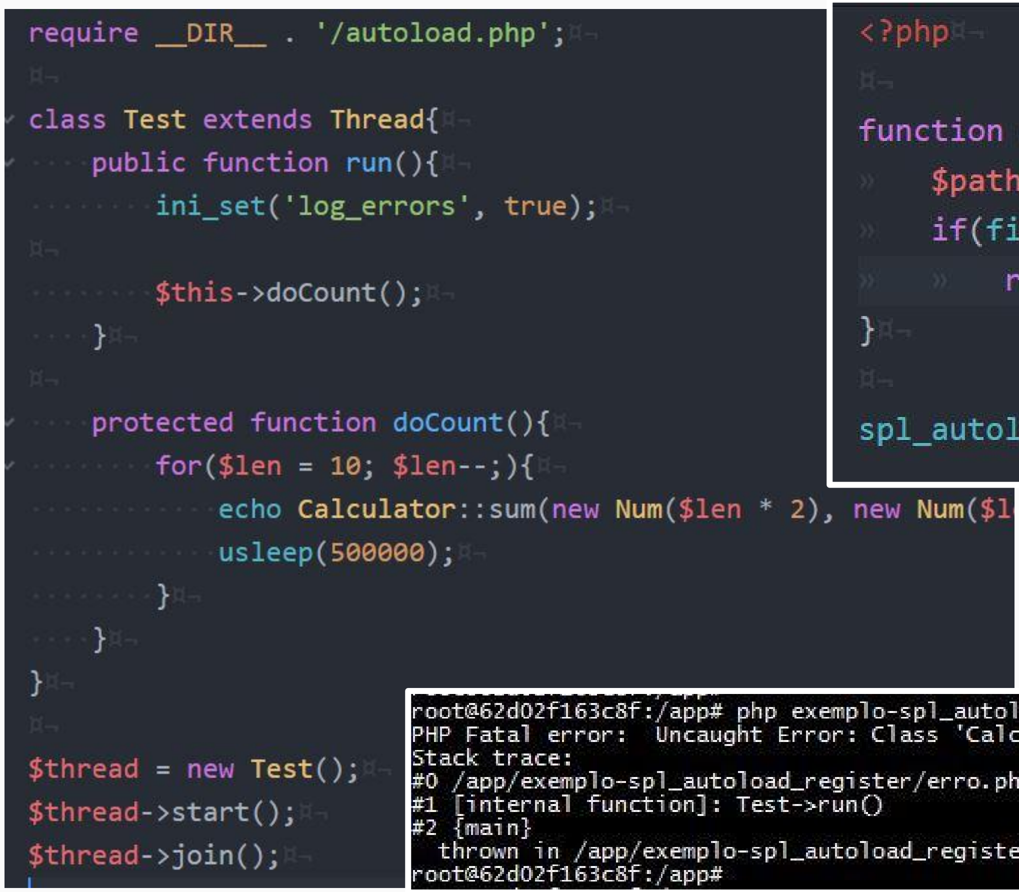


Fonte: https://memesuper.com/categories/view/4eeceefb0993e55aa00e74414f43a035ec14ea4a/meme-feliz-e-triste-ao-mesmo-tempo.html

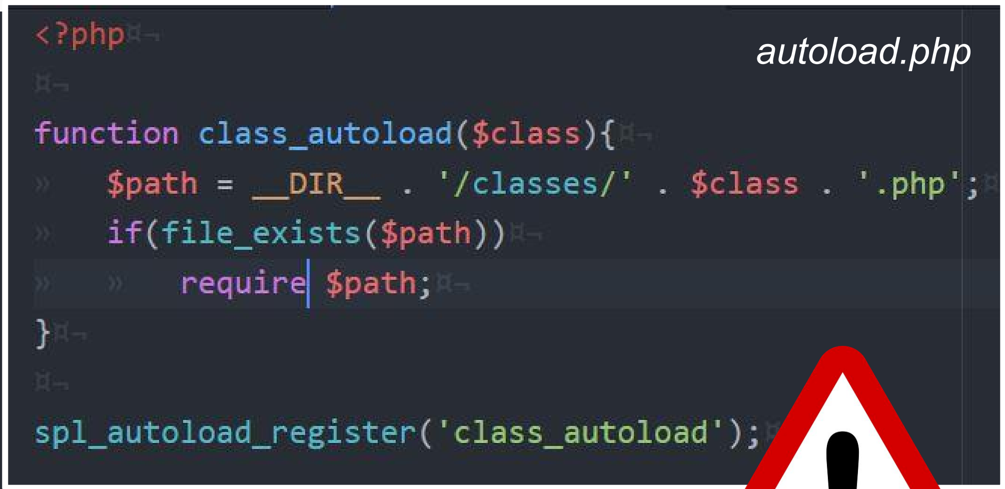


```
root@62d02f163c8f:/app# php exemplo-error-log/ajuste.php
PHP Warning:  Missing argument 1 for simulate_error(), called in /app/exemplo-error-log/ajuste.php on line 10 and defined in /app/exemplo-error-log/ajuste.ph
PHP Fatal error:  Uncaught Error: Call to a member function getValue() on null in /app/exemplo-error-log/ajuste.php:4
Stack trace:
#0 /app/exemplo-error-log/ajuste.php(10): simulate_error()
#1 [internal function]: Example->run()
#2 {main}
  thrown in /app/exemplo-error-log/ajuste.php on line 4
```

spl_autoload_register

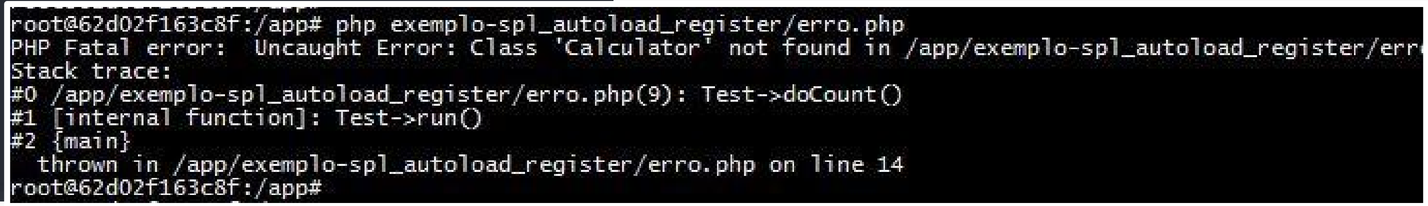


```php
require __DIR__ . '/autoload.php';

class Test extends Thread{
    public function run(){
        ini_set('log_errors', true);

        $this->doCount();
    }

    protected function doCount(){
        for($len = 10; $len--;){
            echo Calculator::sum(new Num($len * 2), new Num($l
            usleep(500000);
        }
    }
}

$thread = new Test();
$thread->start();
$thread->join();
```



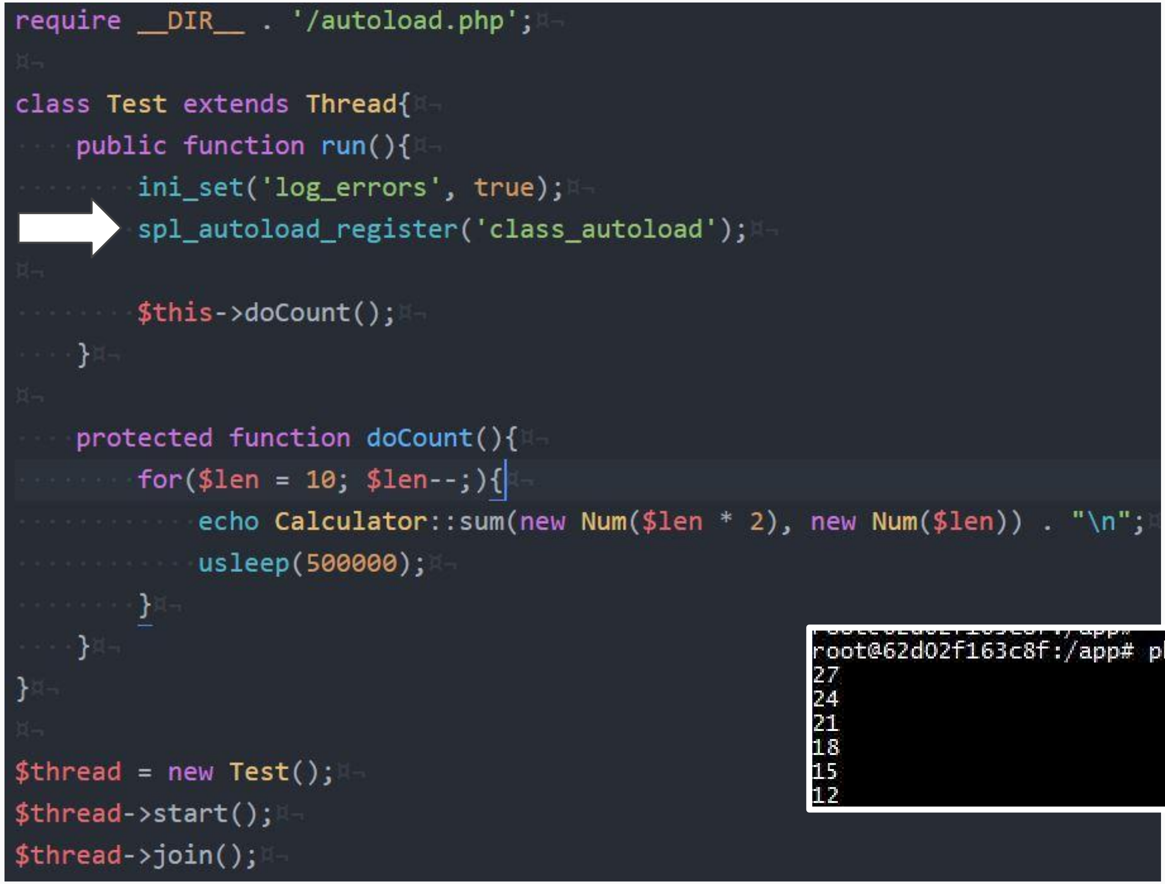
autoload.php

```php
<?php

function class_autoload($class){
    $path = __DIR__ . '/classes/' . $class . '.php';
    if(file_exists($path))
        require $path;
}

spl_autoload_register('class_autoload');
```

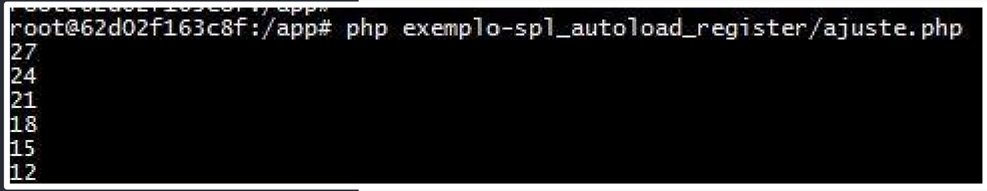


```
root@62d02f163c8f:/app# php exemplo-spl_autoload_register/erro.php
PHP Fatal error:  Uncaught Error: Class 'Calculator' not found in /app/exemplo-spl_autoload_register/err
Stack trace:
#0 /app/exemplo-spl_autoload_register/erro.php(9): Test->doCount()
#1 [internal function]: Test->run()
#2 {main}
  thrown in /app/exemplo-spl_autoload_register/erro.php on line 14
root@62d02f163c8f:/app#
```

spl_autoload_register

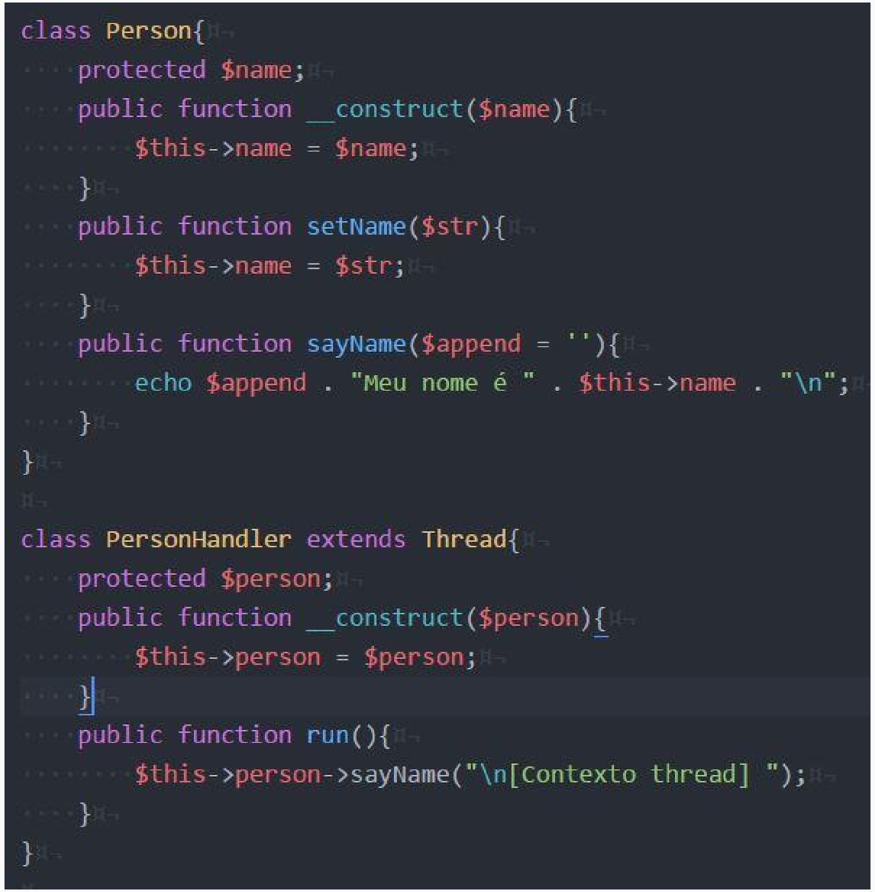


```php
require __DIR__ . '/autoload.php';

class Test extends Thread{
    public function run(){
        ini_set('log_errors', true);
        spl_autoload_register('class_autoload');

        $this->doCount();
    }

    protected function doCount(){
        for($len = 10; $len--;){
            echo Calculator::sum(new Num($len * 2), new Num($len)) . "\n";
            usleep(500000);
        }
    }
}

$thread = new Test();
$thread->start();
$thread->join();
```

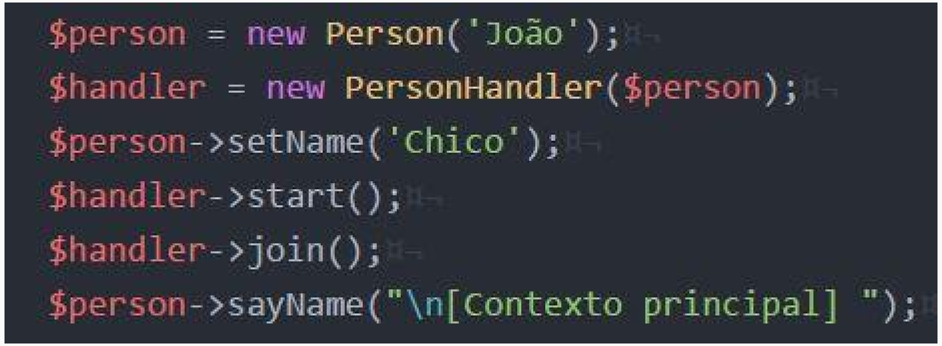


```
root@62d02f163c8f:/app# php exemplo-spl_autoload_register/ajuste.php
27
24
21
18
15
12
```

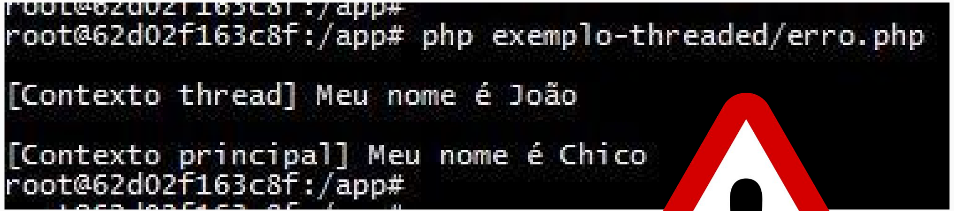
Classe Threaded



```php
class Person{
    protected $name;
    public function __construct($name){
        $this->name = $name;
    }
    public function setName($str){
        $this->name = $str;
    }
    public function sayName($append = ''){
        echo $append . "Meu nome é " . $this->name . "\n";
    }
}

class PersonHandler extends Thread{
    protected $person;
    public function __construct($person){
        $this->person = $person;
    }
    public function run(){
        $this->person->sayName("\n[Contexto thread] ");
    }
}
```

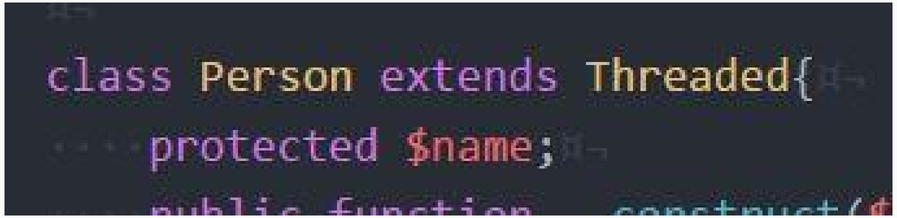


```php
$person = new Person('João');
$handler = new PersonHandler($person);
$person->setName('Chico');
$handler->start();
$handler->join();
$person->sayName("\n[Contexto principal] ");
```

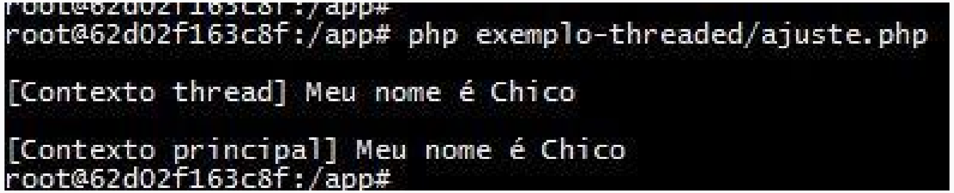


```
root@62d02f163c8f:/app# php exemplo-threaded/erro.php

[Contexto thread] Meu nome é João

[Contexto principal] Meu nome é Chico
root@62d02f163c8f:/app#
```

Classe Threaded



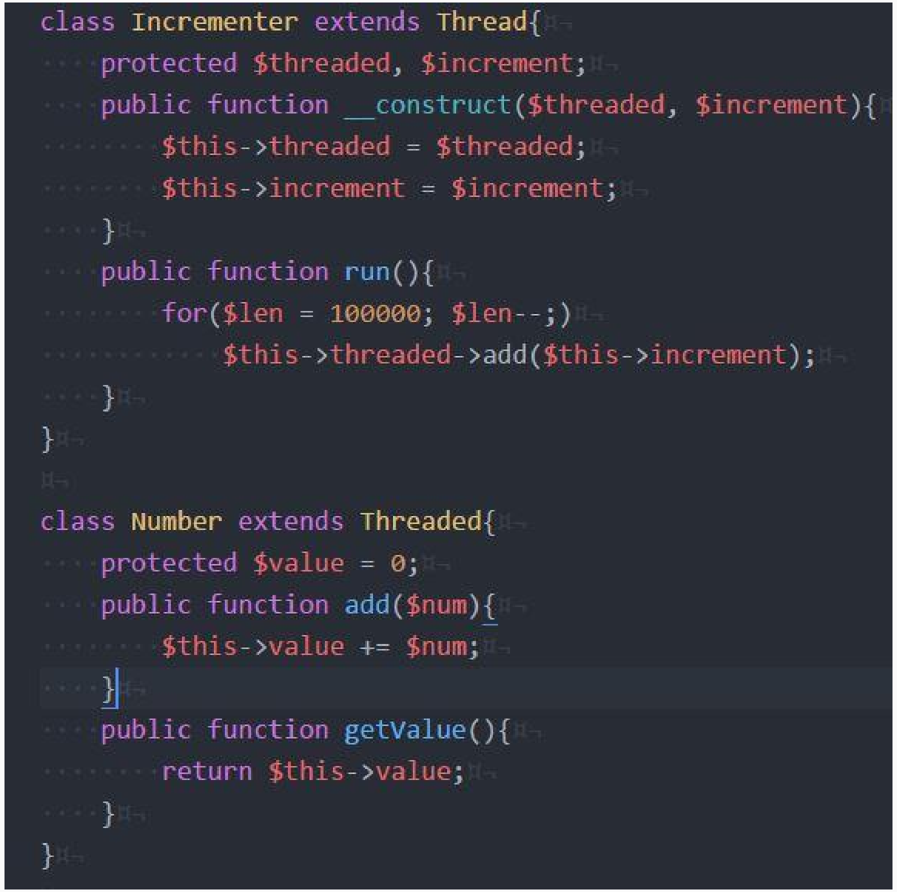
```
class Person extends Threaded{
    protected $name;
```



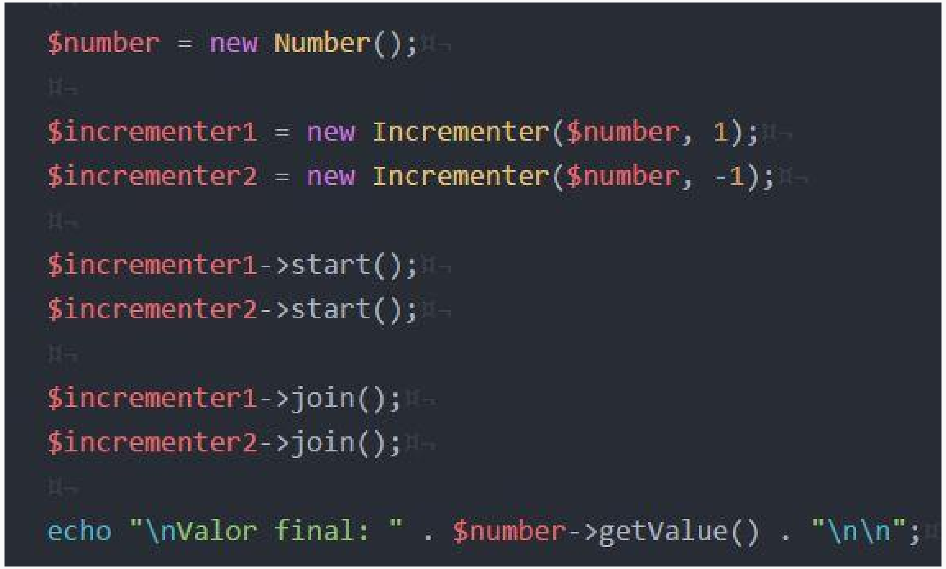
```
root@62d02f163c8f:/app# php exemplo-threaded/ajuste.php

[Contexto thread] Meu nome é Chico

[Contexto principal] Meu nome é Chico
root@62d02f163c8f:/app#
```

Fonte: https://giphy.com/search/ok

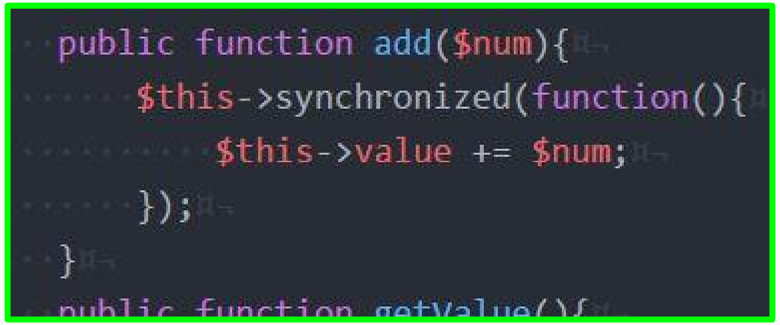
Sincronização



```php
class Incrementer extends Thread{
    protected $threaded, $increment;
    public function __construct($threaded, $increment){
        $this->threaded = $threaded;
        $this->increment = $increment;
    }
    public function run(){
        for($len = 100000; $len--;)
            $this->threaded->add($this->increment);
    }
}

class Number extends Threaded{
    protected $value = 0;
    public function add($num){
        $this->value += $num;
    }
    public function getValue(){
        return $this->value;
    }
}
```

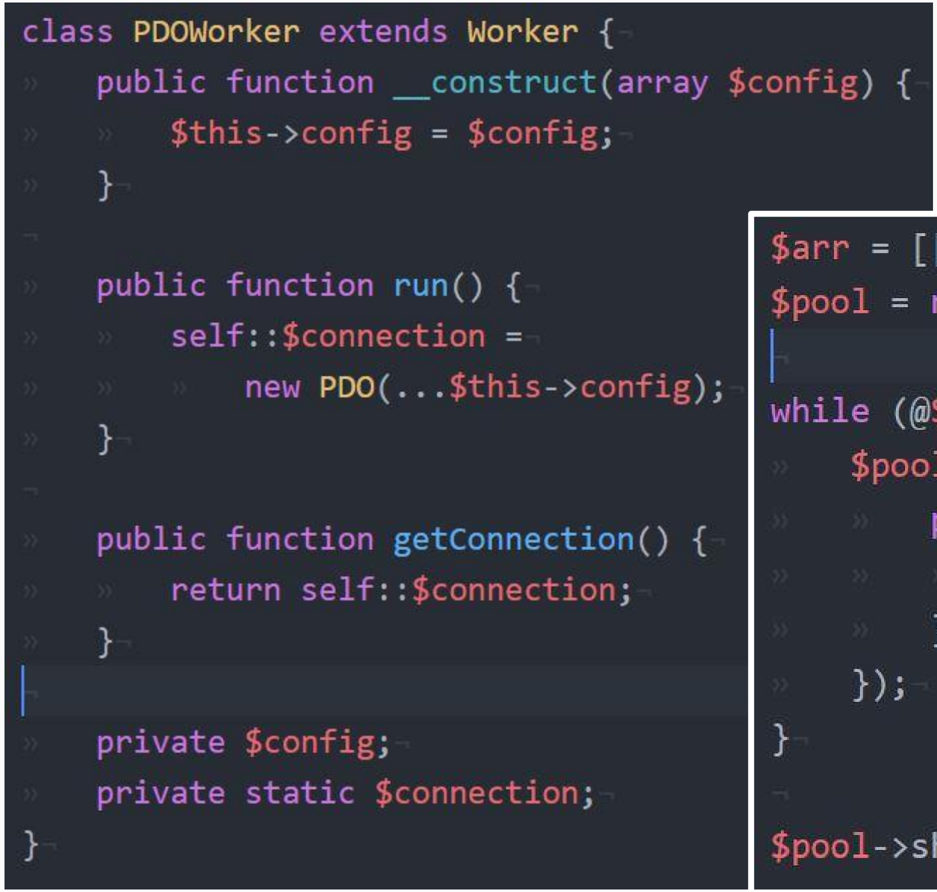


```php
$number = new Number();

$incrementer1 = new Incrementer($number, 1);
$incrementer2 = new Incrementer($number, -1);

$incrementer1->start();
$incrementer2->start();

$incrementer1->join();
$incrementer2->join();

echo "\nValor final: " . $number->getValue() . "\n\n";
```

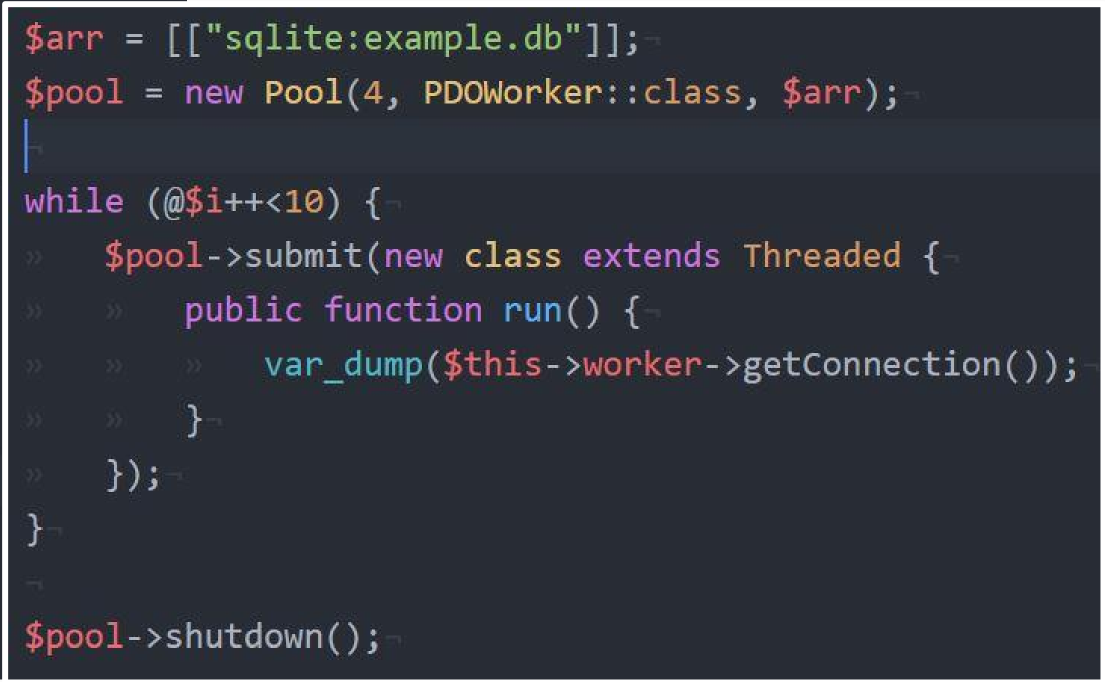


```php
    public function add($num){
        $this->synchronized(function(){
            $this->value += $num;
        });
    }
    public function getValue(){
```

Classes Worker e Pool



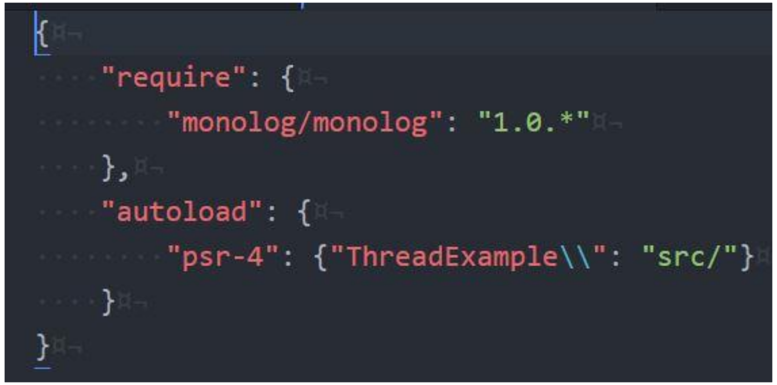
```php
class PDOWorker extends Worker {
    public function __construct(array $config) {
        $this->config = $config;
    }

    public function run() {
        self::$connection =
            new PDO(...$this->config);
    }

    public function getConnection() {
        return self::$connection;
    }

    private $config;
    private static $connection;
}
```



```php
$arr = [["sqlite:example.db"]];
$pool = new Pool(4, PDOWorker::class, $arr);

while (@$i++<10) {
    $pool->submit(new class extends Threaded {
        public function run() {
            var_dump($this->worker->getConnection());
        }
    });
}

$pool->shutdown();
```

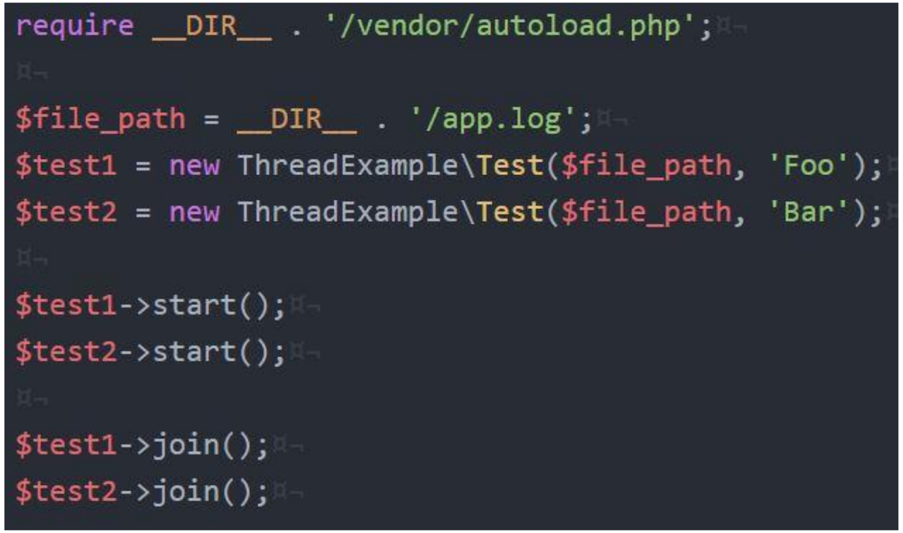
pthreads e Composer

composer.json



```
{
    "require": {
        "monolog/monolog": "1.0.*"
    },
    "autoload": {
        "psr-4": {"ThreadExample\\": "src/"}
    }
}
```

index.php



```
require __DIR__ . '/vendor/autoload.php';

$file_path = __DIR__ . '/app.log';
$test1 = new ThreadExample\Test($file_path, 'Foo');
$test2 = new ThreadExample\Test($file_path, 'Bar');

$test1->start();
$test2->start();

$test1->join();
$test2->join();
```

Obrigado!

github.com/krakjoe/pthreads

github.com/Jesm/conf-PHPRS-2017-pthreads

avalie.se/phprs

Referências

- http://eddmann.com/posts/compiling-php-5-5-with-zts-and-pthreads-support/
- https://gist.github.com/krakjoe/6437782
- https://gist.github.com/krakjoe/9384409
- https://en.wikipedia.org/wiki/Thread_(computing)
- https://www.mullie.eu/parallel-processing-multi-tasking-php/
- http://php.net/manual/en/class.worker.php
- https://github.com/krakjoe/pthreads-autoloading-composer