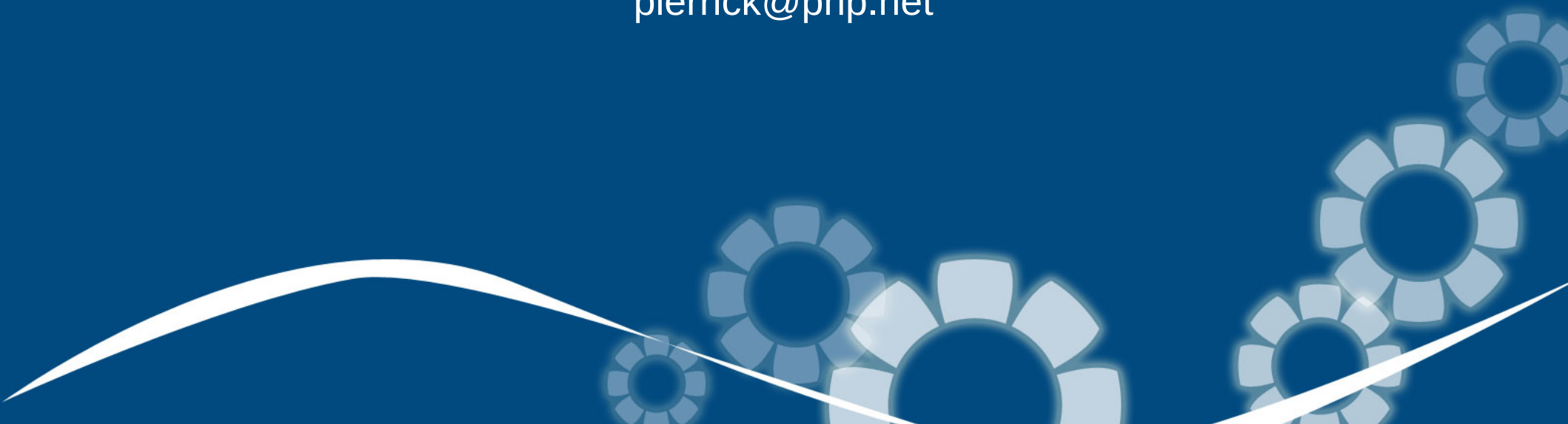
Premiers pas dans les extensions PHP

Pierrick Charron, Confoo, Mars 2011

pierrick@php.net



Qui suis-je ?

- Consultant TI à Montréal
- Auteur de l'extension PHP Stomp
- Contributeur PHP depuis 2009
  - Core
  - RFC (Request For Comment)
  - Documentation (Fr & En)

Une extension ?

Pourquoi créer une extension ?

- Permettre l'utilisation d'une librairie externe
  - Exemple : mysql, curl
- Améliorer les performances
- Opération impossible dans l'espace utilisateur
  - Exemple : Xdebug, operator, vld
- Satisfaire sa curiosité

Avant de commencer

- Présentation basée sur PHP >= 5.3
- Utilisation de Linux

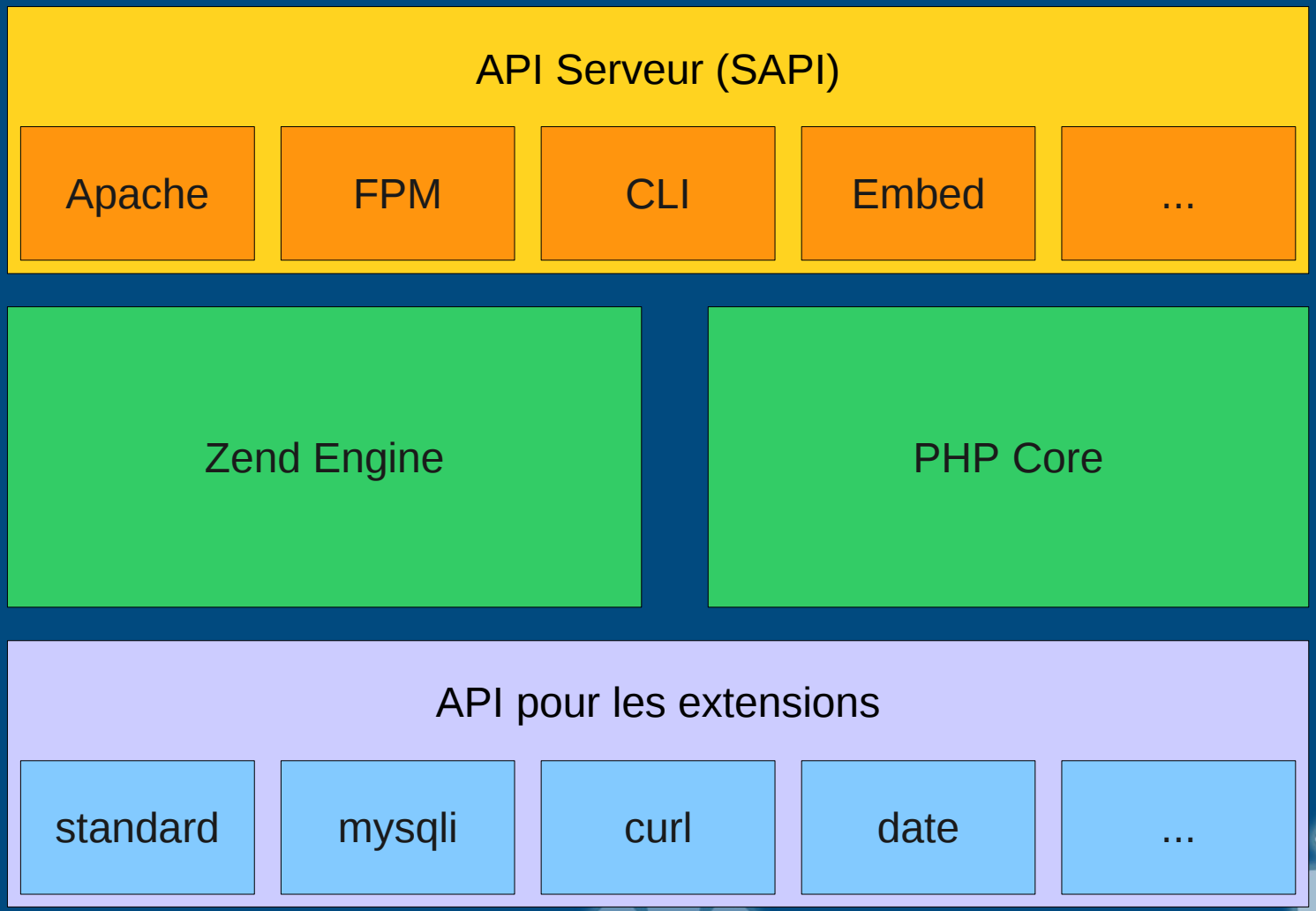
```c
#include <stdio.h>
void main() {
    printf("C Code");
}
```

```
$ echo "terminal linux"
terminal linux
```

Prérequis

- Familier avec PHP
  - Configuration de PHP
  - Installation de modules
- Connaissances en C
- Patience
  - API très complet
  - Beaucoup, beaucoup, beaucoup... de macros

Architecture de PHP

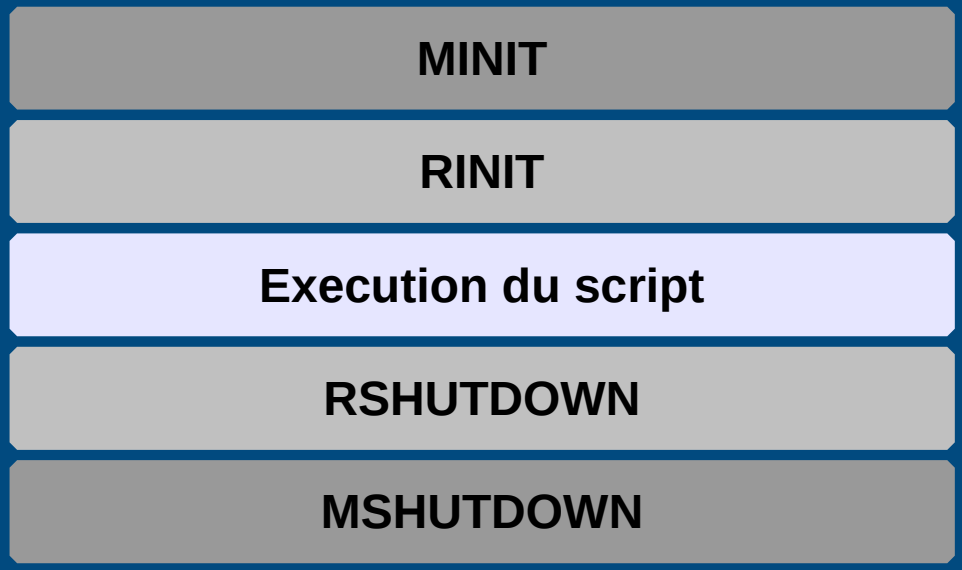


Cycle de vie d'une extension

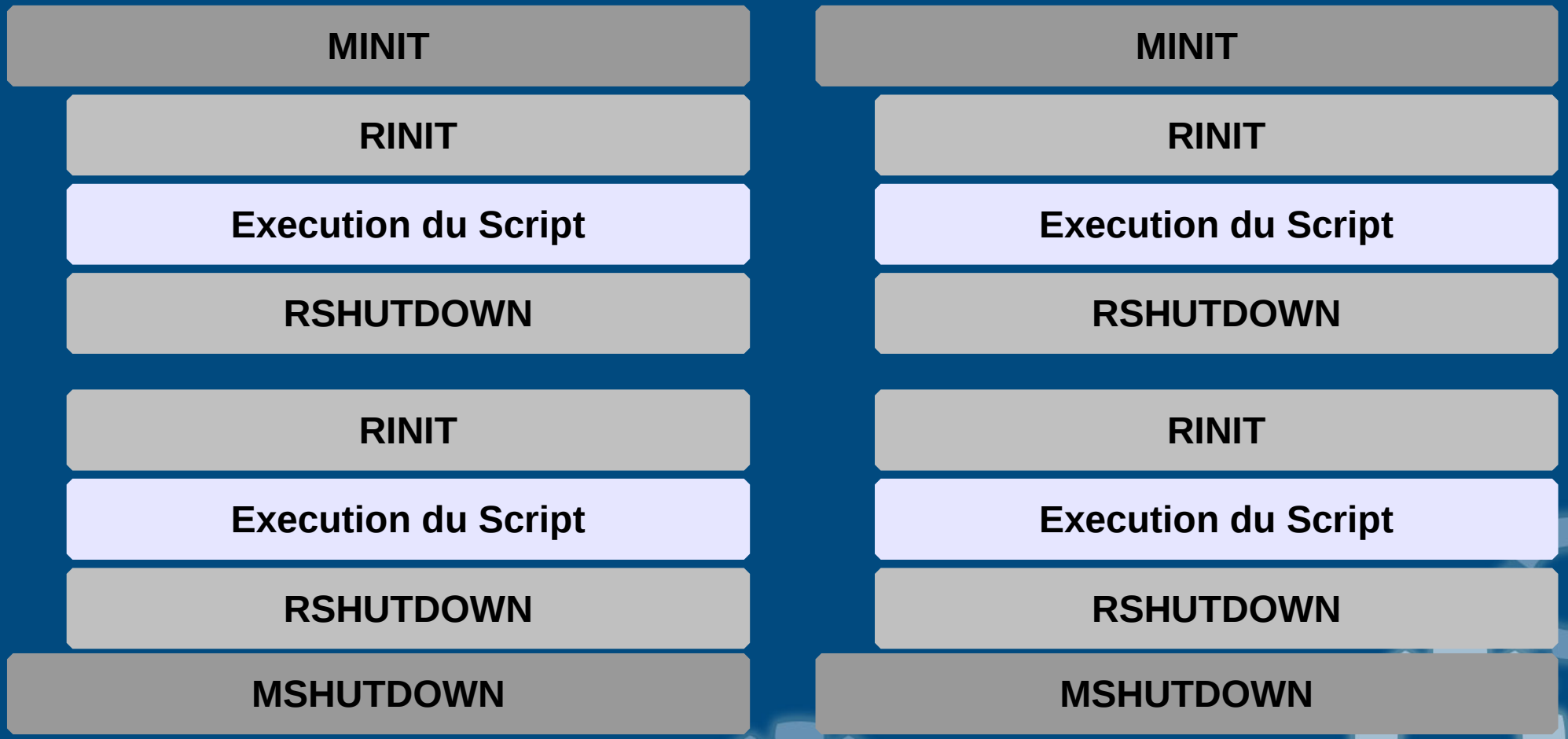
- Différentes étapes
  - MINIT : Initialisation du module
  - RINIT : Initialisation de la requête
  - RSHUTDOWN : Arrêt de la requête
  - MSHUTDOWN : Arrêt du module
- Moment et nombre d'appels dépend du SAPI

Cycle de vie : CLI

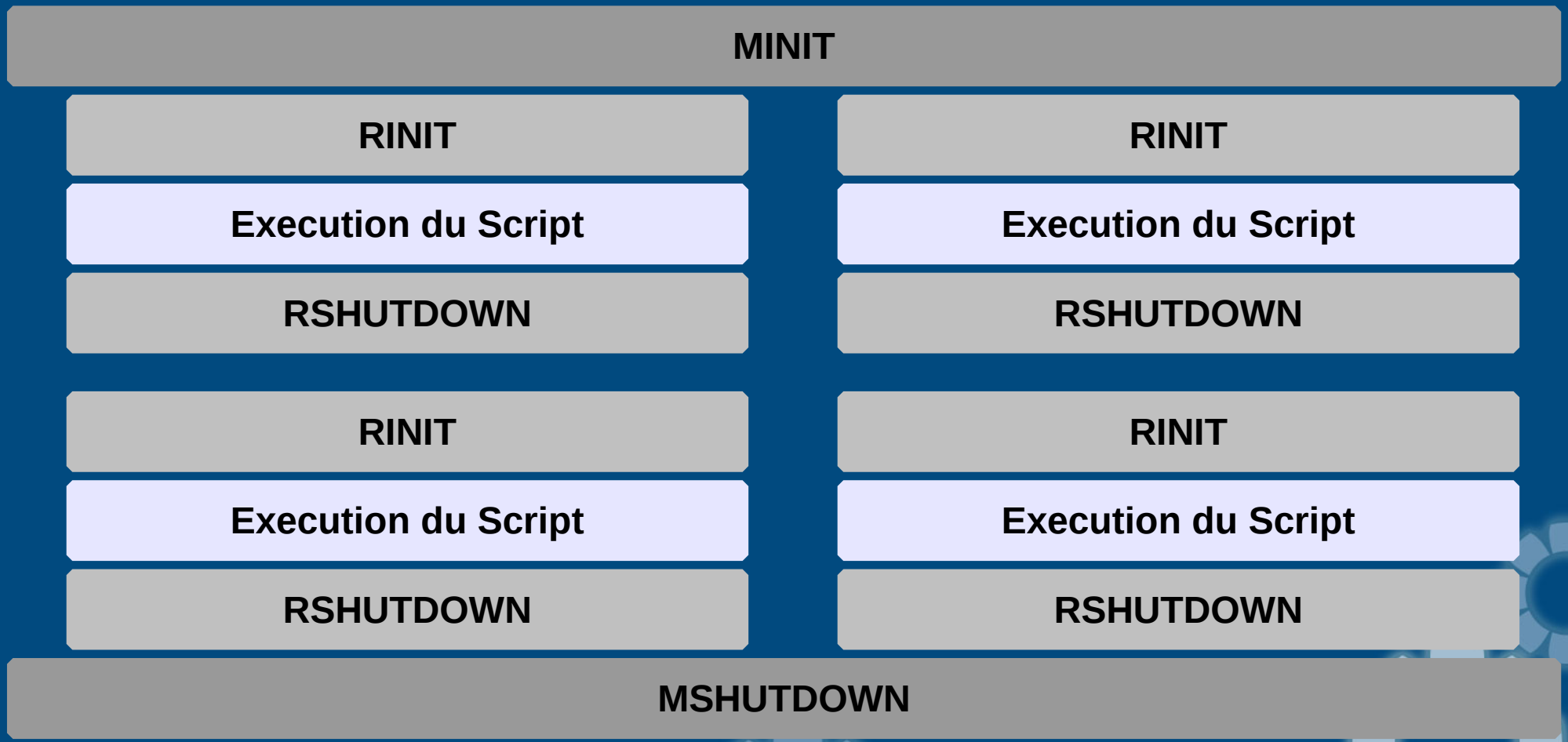
```
$ php monscript.php
```



Cycle de vie : Multiprocess



Cycle de vie : Multithreaded



Zend Thread Safety / TSRM

- Utilisé avec des "threaded webservers"
- Chaque thread a son propre stockage local
  - Evite les accès concurrents

Les macros ZTS

```
#define TSRMLS_D        void ***tsrmls
#define TSRMLS_DC       , TSRMLS_D
#define TSRMLS_C        tsrmls
#define TSRMLS_CC       , TSRMLS_C
```

```
static void ma_fonction(int i TSRMLS_DC);
ma_function(10 TSRMLS_CC);
```

```
static void ma_fonction(int i);
ma_function(10);
```

```
static void ma_fonction(int i, void ***tsrmls);
ma_function(10, tsrmls);
```

Zend Value

ZVAL

- Zval : valeur dans espace utilisateur
  - Attention valeur != variable
- Conversion de type
- Compteur de référence

ZVAL

```
typedef struct _zval_struct {
    zvalue_value value;
    zend_uint refcount__gc;
    zend_uchar type;
    zend_uchar is_ref__gc;
} zval;
```

```
IS_NULL
IS_LONG
IS_DOUBLE
IS_BOOL
IS_ARRAY
IS_OBJECT
IS_STRING
IS_RESOURCE
```

```
typedef union _zvalue_value {
    long lval;
    double dval;
    struct {
        char *val;
        int len;
    } str;
    HashTable *ht;
    zend_object_value *obj;
} zvalue_value;
```

ZVAL

```
typedef struct _zval_struct {
    zvalue_value value;
    zend_uint refcount__gc;
    zend_uchar type;
    zend_uchar is_ref__gc;
} zval;
```

Nombre de variables qui pointent vers cette valeur

Définit si oui ou non une valeur est une référence

ZVAL

```
typedef struct _zval_struct {
    zvalue_value value;
    zend_uint refcount__gc;
    zend_uchar type;
    zend_uchar is_ref__gc;
} zval;
```

Nombre de variables qui pointent vers cette valeur

Définit si oui ou non une valeur est une référence


```
$a = 10;
$b = $a;
$b = 20;
```

```
value.lval = 10
refcount = 2
type = IS_LONG
is_ref = 0
```

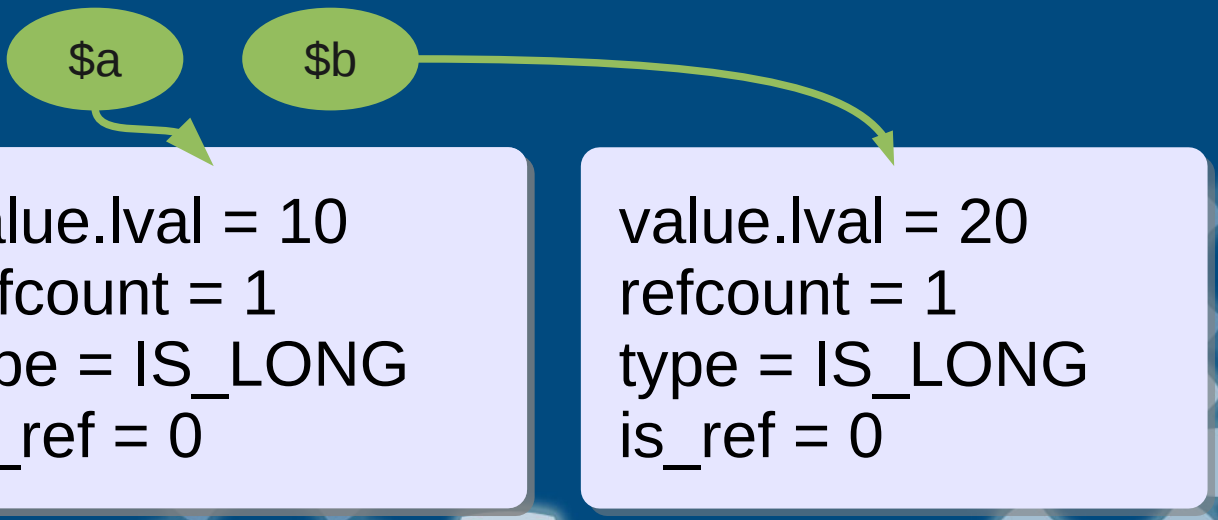
ZVAL

```
typedef struct _zval_struct {
    zvalue_value value;
    zend_uint refcount__gc;
    zend_uchar type;
    zend_uchar is_ref__gc;
} zval;
```

Nombre de variables qui pointent vers cette valeur

Définit si oui ou non une valeur est une référence

```
$a = 10;
$b = $a;
$b = 20;
```



ZVAL

```
typedef struct _zval_struct {
    zvalue_value value;
    zend_uint refcount__gc;
    zend_uchar type;
    zend_uchar is_ref__gc;
} zval;
```

Nombre de variables qui pointent vers cette valeur

Définit si oui ou non une valeur est une référence


```
$a = 10;
$b = &$a;
$b = 20;
```

```
value.lval = 10
refcount = 2
type = IS_LONG
is_ref = 1
```

ZVAL

```
typedef struct _zval_struct {
    zvalue_value value;
    zend_uint refcount__gc;
    zend_uchar type;
    zend_uchar is_ref__gc;
} zval;
```

Nombre de variables qui pointent vers cette valeur

Définit si oui ou non une valeur est une référence


```
$a = 10;
$b = &$a;
$b = 20;
```

```
value.lval = 20
refcount = 2
type = IS_LONG
is_ref = 1
```

ZVAL : Récuperer le type

```
#define Z_TYPE(zval)          (zval).type
#define Z_TYPE_P(zval_p)      Z_TYPE(*zval_p)
#define Z_TYPE_PP(zval_pp)    Z_TYPE(**zval_pp)
```

```
zval foo;
Z_TYPE(foo) == IS_LONG

zval *bar;
Z_TYPE_P(bar) == IS_BOOL
```

ZVAL : Récuperer une valeur

```
#define Z_LVAL(zval)        (zval).value.lval
#define Z_BVAL(zval)        ((zend_bool)(zval).value.lval)
#define Z_DVAL(zval)        (zval).value.dval
#define Z_STRVAL(zval)      (zval).value.str.val
#define Z_STRLEN(zval)      (zval).value.str.len
#define Z_ARRVAL(zval)      (zval).value.ht
#define Z_OBJVAL(zval)      (zval).value.obj
```

```
#define Z_*_P(zval_p)       (*zval).
#define Z_*_PP(zval_pp)     (**zval).
```

ZVAL : Assigner une valeur

```
ZVAL_NULL(zval);                    Z_TYPE_P(zval) = IS_NULL;

ZVAL_LONG(zval,l);                  Z_TYPE_P(zval) = IS_LONG;
                                    Z_LVAL_P(zval) = l;

ZVAL_DOUBLE(zval,d);                Z_TYPE_P(zval) = IS_DOUBLE;
                                    Z_DVAL_P(zval) = d;

ZVAL_STRINGL(zval,str,len,dup)      Z_TYPE_P(zval) = IS_STRING;
                                    Z_STRLEN_P(zval) = len;
                                    if (dup) {
                                        Z_STRVAL_P(zval) = estrndup(str, len+1);
                                    } else {
                                        Z_STRVAL_P(zval) = str;
                                    }

ZVAL_BOOL(zval, b)                  Z_TYPE_P(zval) = IS_BOOLEAN;
                                    Z_BVAL_P(zval) = b ? 1 : 0;
```

Conversion

```
void convert_to_string(zval *);
void convert_to_long(zval *);
void convert_to_double(zval *);
void convert_to_null(zval *);
void convert_to_boolean(zval *);
void convert_to_array(zval *);
void convert_to_object(zval *);
```

Gestion de la mémoire

- Zend dispose d'un memory manager
  - évite les fuites de mémoire
  - signature des fonctions similaires aux fonctions natives C

Les fonctions d'allocation

Fonction C traditionnelle	Fonction spécifique à PHP
void *malloc(size_t count);	void *emalloc(size_t count);
void *calloc(size_t nmemb, size_t size);	void *ecalloc(size_t nmemb, size_t size);
void *realloc(void *ptr, size_t size);	void *erealloc(void *ptr, size_t size);
char *strdup(const char *s);	char *estrdup(const char *s);
char *strndup(const char *s, size_t n);	char *estrndup(const char *s, size_t n);
void free(void *ptr);	void efree(void *ptr);

Et les extensions alors ?

./ext_skel

- Outil de génération de code
- Génère le squelette de l'extension

```
$ ./ext_skel --extname=monextension
```

Les fichiers de *monextension*

```
$ tree monextension

monextension/
|-- config.m4
|-- config.w32
|-- CREDITS
|-- EXPERIMENTAL
|-- monextension.c
|-- monextension.php
|-- php_monextension.h
`-- tests
    `-- 001.phpt

1 directory, 8 files
```

- Fichiers de configuration
  - Config.m4
  - Config.w32
- Fichiers de code
  - monextension.c
  - php_monextension.h

Config.m4

- Script de configuration sous Linux
  - Utilisé par ./buildconf et phpize

```
dnl If your extension references something external, use with:

dnl PHP_ARG_WITH(monextension, for monextension support,
dnl Make sure that the comment is aligned:
dnl [  --with-monextension           Include monextension support])

dnl Otherwise use enable:

PHP_ARG_ENABLE(monextension, whether to enable monextension support,
Make sure that the comment is aligned:
[  --enable-monextension          Enable monextension support])
```

Compiler *monextension*

- Dynamique

```
$ phpize
$ ./configure
$ make
$ make test
```

- Statique

```
$ ./buildconf --force
$ ./configure --enable-monextension
$ make
$ make test
```

PHPT

Avant même de commencer à programmer,
pensez à écrire vos tests !!!!

http://qa.php.net/write-test.php

php_monextension.h

- Déclaration des données pour la liaison statique

monextension.c

- #includes
- Structures statiques
- Fonctions PHP
- MINFO
- MINIT, MSHUTDOWN, RINIT, RSHUTDOWN
- Table des fonctions
- Définition du module

Définition du module

- Structure permettant à PHP d'initialiser ou de désinitialiser le module

```
zend_module_entry monextension_module_entry = {
    STANDARD_MODULE_HEADER,
    "monextension",
    monextension_functions,
    PHP_MINIT(monextension),
    PHP_MSHUTDOWN(monextension),
    PHP_RINIT(monextension),
    PHP_RSHUTDOWN(monextension),
    PHP_MINFO(monextension),
    "0.1",
    STANDARD_MODULE_PROPERTIES
};
```

Liste des fonctions

```
const zend_function_entry monextension_functions[] = {
        PHP_FE(confirm_monextension_compiled,   NULL)
        {NULL, NULL, NULL}
};
```

- PHP_FE(nom, signature);
- PHP_FALIAS(nom, nom_original, signature);

Première fonction

monextension.c

```
const zend_function_entry monextension_functions[] = {
        PHP_FE(monextension_helloworld,   NULL)
        {NULL, NULL, NULL}
};

/** ... **/

PHP_FUNCTION(monextension_helloworld)
{
      php_printf("Hello world !!!");
};
```

php_monextension.h

```
PHP_FUNCTION(monextension_helloworld);
```

PHP_FUNCTION

```
PHP_FUNCTION(monextension_helloworld)
{
	php_printf("Hello world !!!");
}
```

```
void zif_monextension_helloworld(int ht, zval *return_value, zval
**return_value_ptr, zval *this_ptr, int return_value_used TSRMLS_DC)
{
	php_printf("Hello world !!!");
}
```

Valeur de retour

- Modifier la variable return_value (zval *)
- Par défaut Z_TYPE_P(return_value) = IS_NULL
- **Attention** ne pas retourner la valeur !

```
PHP_FUNCTION(monextension_helloworld)
{
    ZVAL_STRING(return_value, "Hello world !!!", 1);
    return;
};
```

RETVAL_*

ZVAL Macro	RETVAL_*
ZVAL_NULL(return_value);	RETVAL_NULL();
ZVAL_BOOL(return_value, bval);	RETVAL_BOOL(bval);
ZVAL_TRUE(return_value);	RETVAL_TRUE();
ZVAL_FALSE(return_value);	RETVAL_FALSE();
ZVAL_LONG(return_value, lval);	RETVAL_LONG(lval);
ZVAL_DOUBLE(return_value, dval);	RETVAL_DOUBLE(dval);
ZVAL_STRING(return_value, str, dup);	RETVAL_STRING(str, dup);
ZVAL_STRINGL(return_value, str, len, dup);	RETVAL_STRINGL(str, len, dup);
ZVAL_RESOURCE(return_value, rval);	RETVAL_RESOURCE(rval);

```
PHP_FUNCTION(monextension_helloworld)
{
    RETVAL_STRING("Hello world !!!", 1);
    /* Code executé */
    return;
};
```

RETURN_*

- RETVAL_* n'interrompt pas l'exécution du code
- RETURN_* = RETVAL_*; return;

```
PHP_FUNCTION(monextension_helloworld)
{
    RETURN_STRING("Hello world !!!", 1);
    /* Code jamais executé */
};
```

Paramètres de fonction

```
PHP_FUNCTION(monextension_helloworld)
{
	char *name;
	int name_len;
	if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
		"s", &name, &name_len) == FAILURE) {
		return;
	}
	php_printf("Hello %s !", name);
};
```

Paramètres de fonction

```
int zend_parse_parameters (int num_args TSRMLS_DC,  char *type_spec, ...)
```

Spécificateur du type	Type côté PHP	Type de donnée côté C
b	Boolean	zend_bool *
l	Entier	long *
d	Virgule floatante	double *
s	Chaine de caractère	char **, int *
r	Ressource	zval **
a	Tableau	zval **
o	Objet	zval **
O	Objet d'un type spécifique	zval **, zend_class_entry *
z	zval	zval **

Paramètres optionnels

- Utilisation du modificateur "|"

```
PHP_FUNCTION(monextension_helloworld)
{
	char *name = "world";
	int name_len = sizeof("world")-1;
	if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
		"|s", &name, &name_len) == FAILURE) {
		return;
	}
	php_printf("Hello %s !", name);
};
```

Les tableaux (HashTable *)

- Stockage clef valeur
- Très rapide
  - Combinaison de tableau C et de listes chaînées
- Utilisés presque partout dans le corp PHP
- Recherche par valeur impossible

Initialiser un tableau

```
int array_init(zval *z); /* $array = array(); */
```

```
PHP_FUNCTION(monextension_emptyarray)
{
    array_init(return_value);
};
```

Ajout d'éléments

Tableau numérique

Syntaxe PHP	Syntaxe C
$arr[1] = null;	add_index_null(arr, 1);
$arr[2] = "foo";	add_index_string(arr, 2, "foo", 1);
$arr[3] = true;	add_index_bool(arr, 3, 1);
$arr[4] = 10;	add_index_long(arr, 4, 10);
$arr[5] = 3.1416;	add_index_double(arr, 5, 3.1416);
$arr[6] = $foo;	add_index_zval(arr, 6, foo);
$arr[] = null;	add_next_index_null(arr);
$arr[] = "foo";	add_next_index_string(arr, "foo", 1);
$arr[] = true;	add_next_index_bool(arr, 1);
$arr[] = 10;	add_next_index_long(arr, 10);
$arr[] = 3.1416;	add_next_index_double(arr, 3.1416);
$arr[] = $foo;	add_next_index_zval(arr, foo);

Ajout d'éléments

Tableau associatif

Syntaxe PHP	Syntaxe C
$arr["foo"] = null;	add_assoc_null(arr, "foo"); add_assoc_null_ex(arr, "foo", sizeof("foo"));
$arr["bar"] = "foo";	add_assoc_string(arr, "bar", "foo", 1); add_assoc_string_ex(arr, "bar", sizeof("bar"), "foo", 1);
$arr["baz"] = true;	add_assoc_bool(arr, "baz", 1); add_assoc_bool_ex(arr, "baz", sizeof("baz"), 1);
$arr["boz"] = 10;	add_assoc_long(arr, "boz", 10); add_assoc_long_ex(arr, "boz", sizeof("boz"), 10);
$arr["biz"] = 3.1416;	add_assoc_double(arr, "biz", 3.1416); add_assoc_double_ex(arr, "biz", sizeof("biz"), 3.1416);
$arr["buz"] = $foo;	add_assoc_zval(arr, "buz", foo); add_assoc_zval(arr, "buz", sizeof("buz"), foo);

Recherche d'éléments

```
int zend_hash_find(const HashTable *ht,
      const char *arKey, uint nKeyLength, void **pData)

int zend_hash_quick_find(const HashTable *ht,
      const char *arKey, uint nKeyLength, ulong h, void **pData)

int zend_hash_index_find(const HashTable *ht, ulong h, void **pData)
```

```
zval **data;

if (zend_symtable_find(Z_ARRVAL(zval), "clef", sizeof("clef"), (void**)&data) ==
SUCCESS) {
      switch (Z_TYPE_PP(data)) {
            /* ... */
      }
}
```

Suppression d'éléments

```
int zend_hash_del(HashTable *ht, char *arKey, uint nKeyLen);
int zend_hash_quick_del(HashTable *ht, char *arKey, uint nKeyLen, ulong h);
int zend_hash_index_del(HashTable *ht, ulong h);
```

Création d'une classe

```
zend_class_entry *counter_ce;
PHP_MINIT_FUNCTION(monextension)
{
    zend_class_entry ce;
    INIT_CLASS_ENTRY(ce, "Counter", monextension_counter_methods);
    counter_ce = zend_register_internal_class(&ce TSRMLS_CC);
    zend_declare_property_long(counter_ce, "value", sizeof("value")-1, 0,
            ZEND_ACC_PUBLIC TSRMLS_CC);
    return SUCCESS;
}
```

```
static zend_function_entry monextension_counter_methods[] = {
      PHP_ME(Counter, reset, NULL, ZEND_ACC_PUBLIC)
      PHP_ME(Counter, getValue, NULL, ZEND_ACC_PUBLIC)
      {NULL, NULL, NULL}
};
```

Les méthodes

```
PHP_METHOD(Counter, getValue)
{
    zval *object = getThis();
    zval *details = NULL;
    details = zend_read_property(counter_ce, object, "value", sizeof("value")-1, 1
TSRMLS_CC);
    RETURN_LONG(Z_LVAL(value));
}

PHP_METHOD(Counter, reset)
{
      zend_update_property_long(counter_ce, getThis(), "value", sizeof("value")-1, 0
TSRMLS_DC)
      RETURN_TRUE;
}
```

Quelques outils bien utiles ;)

Code compatible ZTS ?

Compilez toujours votre PHP avec
--enable-maintainer-zts

Fuites de mémoire ?

--enable-debug

```
$ php -e
<?php leak();
[Tue Feb 22 20:12:17 2011]  Script:  '-'
/home/pierrick/php-src/trunk/Zend/zend_builtin_functions.c(1425) :  Freeing
0x7FA54892B708 (3 bytes), script=-
=== Total 1 memory leaks detected ===

$
```

Fuites de mémoire ?

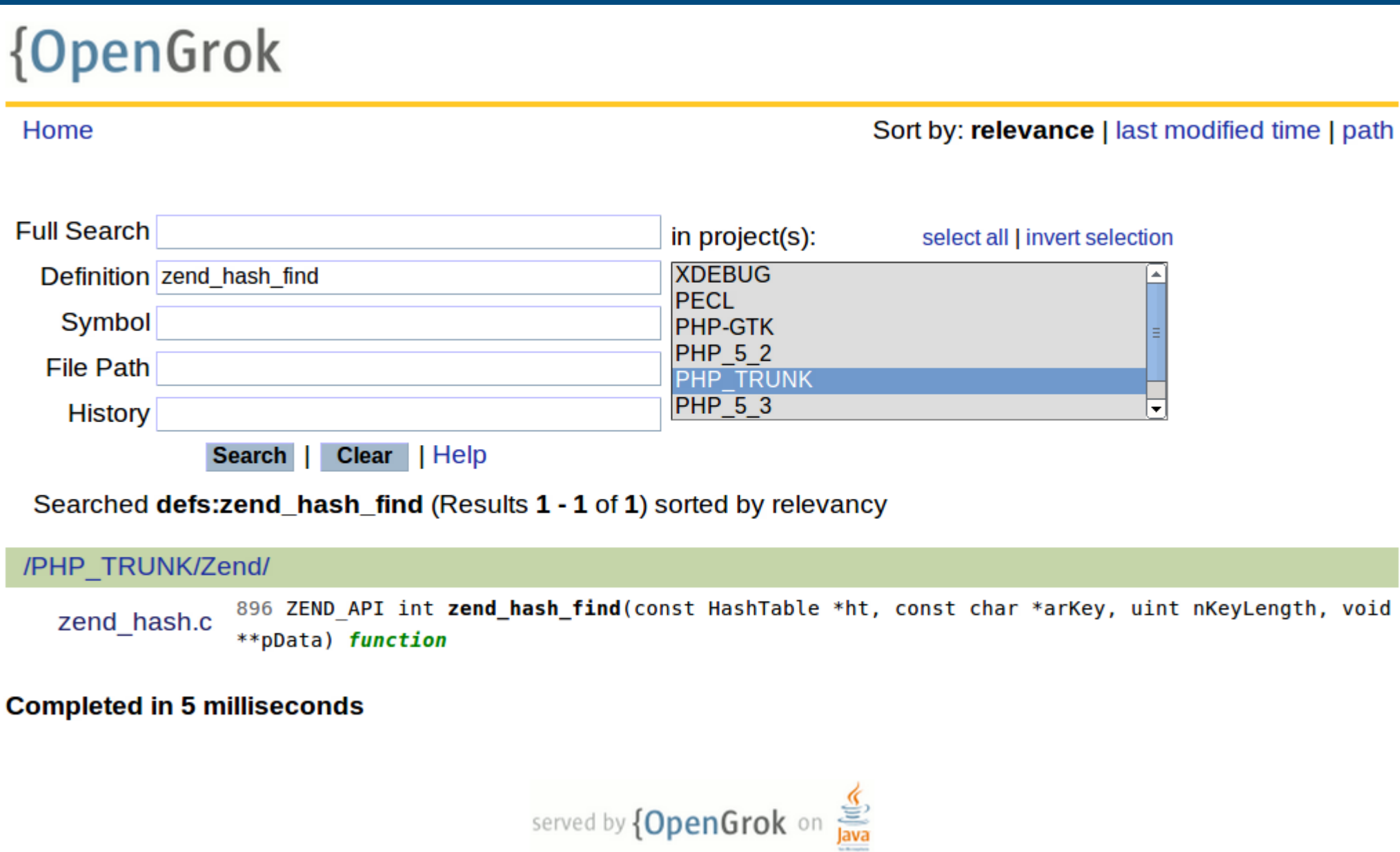
valgrind

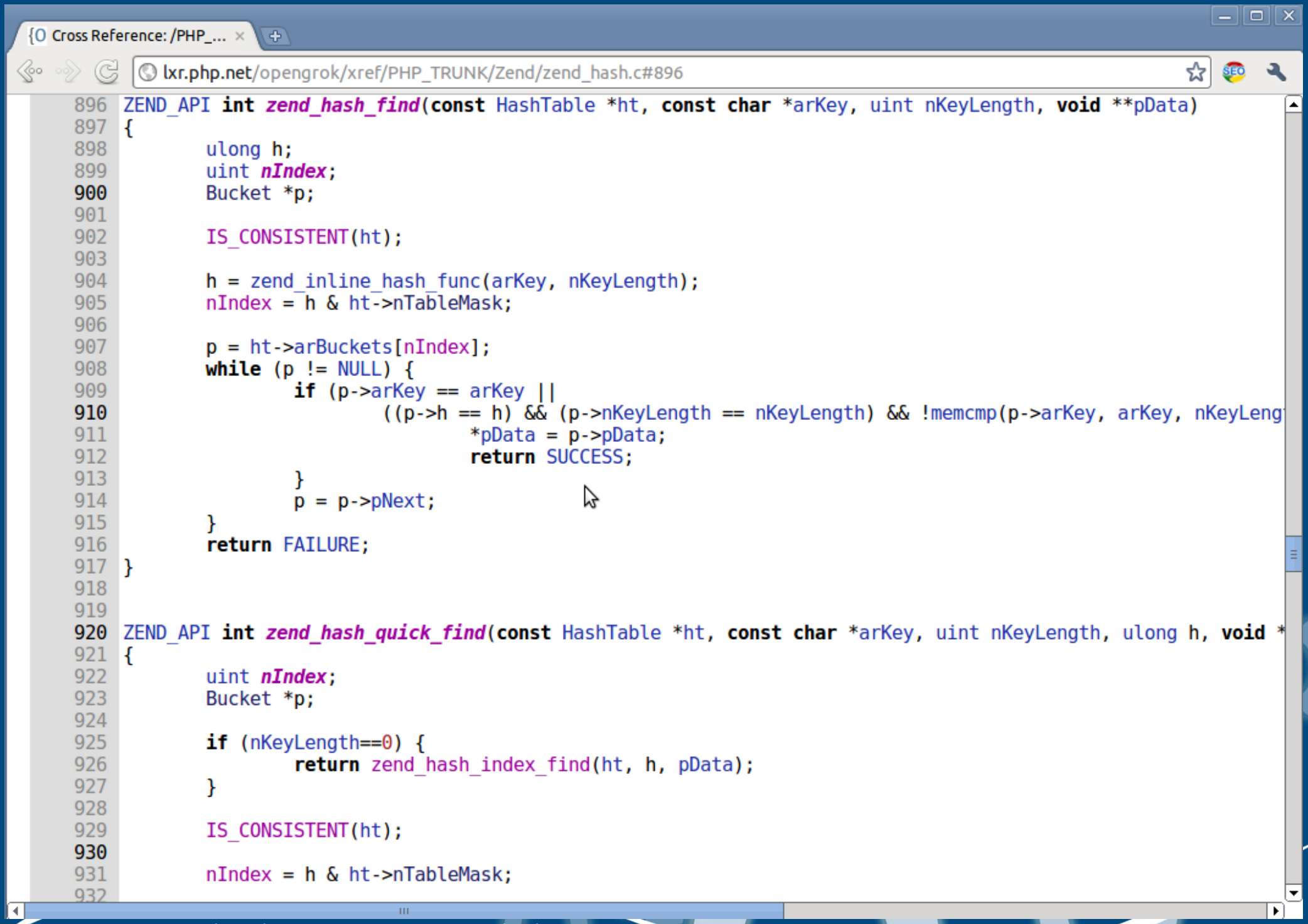
```
$ USE_ZEND_ALLOC=0 valgrind --tool=memcheck --leak-check=yes --num-
callers=30 --show-reachable=yes sapi/cli/php leak.php

....

==7164== LEAK SUMMARY:
==7164==    definitely lost: 0 bytes in 0 blocks
==7164==    indirectly lost: 0 bytes in 0 blocks
==7164==      possibly lost: 0 bytes in 0 blocks
==7164==    still reachable: 1,111,059 bytes in 8,728 blocks
==7164==         suppressed: 0 bytes in 0 blocks

....
```

Recherche de code ?

grep

http://lxr.php.net


{OpenGrok
Home
Sort by: relevance | last modified time | path
Full Search
Definition
zend_hash_find
Symbol
File Path
History
Search | Clear | Help
in project(s):
select all | invert selection
XDEBUG
PECL
PHP-GTK
PHP_5_2
PHP_TRUNK
PHP_5_3
Searched defs:zend_hash_find (Results 1 - 1 of 1) sorted by relevancy
/PHP_TRUNK/Zend/
zend_hash.c
896 ZEND_API int zend_hash_find(const HashTable *ht, const char *arKey, uint nKeyLength, void **pData) function
Completed in 5 milliseconds
served by {OpenGrok on Java


{O Cross Reference: /PHP_...
lxr.php.net/opengrok/xref/PHP_TRUNK/Zend/zend_hash.c#896
896 ZEND_API int zend_hash_find(const HashTable *ht, const char *arKey, uint nKeyLength, void **pData)
897 {
898 ulong h;
899 uint nIndex;
900 Bucket *p;
901
902 IS_CONSISTENT(ht);
903
904 h = zend_inline_hash_func(arKey, nKeyLength);
905 nIndex = h & ht->nTableMask;
906
907 p = ht->arBuckets[nIndex];
908 while (p != NULL) {
909 if (p->arKey == arKey ||
910 ((p->h == h) && (p->nKeyLength == nKeyLength) && !memcmp(p->arKey, arKey, nKeyLeng
911 *pData = p->pData;
912 return SUCCESS;
913 }
914 p = p->pNext;
915 }
916 return FAILURE;
917 }
918
919
920 ZEND_API int zend_hash_quick_find(const HashTable *ht, const char *arKey, uint nKeyLength, ulong h, void *
921 {
922 uint nIndex;
923 Bucket *p;
924
925 if (nKeyLength==0) {
926 return zend_hash_index_find(ht, h, pData);
927 }
928
929 IS_CONSISTENT(ht);
930
931 nIndex = h & ht->nTableMask;
932

Segfault ?

GDB (gnome debugger)

```
$ gdb --args php -d extension=monextension.so segfault.php
(gdb) r
Starting program: /usr/local/bin/php -d extension=monextension.so segfault.php
[Thread debugging using libthread_db enabled]
[New Thread 0x7fffefd1f700 (LWP 9457)]
[Thread 0x7fffefd1f700 (LWP 9457) exited]

Program received signal SIGSEGV, Segmentation fault.
__strcpy_chk () at ../sysdeps/x86_64/strcpy_chk.S:196
196   ../sysdeps/x86_64/strcpy_chk.S: No such file or directory.in ../sysdeps/x86_64/strcpy_chk.S

(gdb) bt

#0  __strcpy_chk () at ../sysdeps/x86_64/strcpy_chk.S:196
#1  0x00007fffefd20b0c in strcpy (ht=<value optimized out>, return_value=<value optimized out>,
return_value_ptr=<value optimized out>, this_ptr=0x6, return_value_used=0, tsrm_ls=0xfa70c0) at
/usr/include/bits/string3.h:107
#2  zif_monextension_helloworld (ht=<value optimized out>, return_value=<value optimized out>,
return_value_ptr=<value optimized out>, this_ptr=0x6, return_value_used=0, tsrm_ls=0xfa70c0) at
/home/pierrick/php-src/trunk/ext/monextension/monextension.c:162
#3  0x00007ffff2658b35 in xdebug_execute_internal (current_execute_data=0x7ffff7eb7050,
return_value_used=0, tsrm_ls=0xfa70c0)     at /home/pierrick/installs/xdebug-2.1.0/xdebug.c:1339
#4  0x000000000081b777 in zend_do_fcall_common_helper_SPEC (execute_data=0x7ffff7eb7050,
tsrm_ls=0xfa70c0) at /home/pierrick/php-
```

Besoin d'aide ?

Irc.EFNet.org #php.pecl

Mailing Lists
http://marc.info/?l=php-internals
http://marc.info/?l=pecl-dev

Envie de lecture ?

http://www.php.net/manual/en/internals2.php

Advanced PHP Programming
par George Schlossnagle

Extending and Embedding PHP
par Sara Golemon
ISBN#0-6723-2704-X

Questions ?

pierrick@php.net
Twitter: @adoyy

Commentaires et présentation :
http://joind.in/2869