Readme is now at:

https://github.com/ircmaxell/PHP-Internals-Book

Content outline

Let's make a clean part here, all comments from/to authors (us) should be noticeable, for example written highlighted

Prerequisites

Through this book we are going to write and compile C code, mainly under Unix/Linux. You then must have ideas of what we'll be talking about, thus this short chapter would refresh your mind about critical concepts.

- Introduction to OSes and Unix processes
  - What is an OS ? What is it for
  - Different OSes, Unix familly
  - Process loading under Linux and memory map
- Introduction to C
  - C as a second-level language
  - Quick thoughts about assembly
  - C types, C rules : just a matter of bytes in memory
    - Pointers
    - Memory areas : process heap and stack, mem leaks
    - Different allocation types (auto, static, global, extern) and sizes
    - structs and enums
  - POSIX
  - Preprocessor and macros management
  - Header files
- Introduction to code compiling and linking
  - Build tools, what are they ? Why are they ?
  - Quick view of GCC, Autotools, libtool, make ...
- Introduction to debugging and profiling
  - Gdb
  - Valgrind
  - strace / ltrace / mtrace . ioprofile
  - /proc FS

A first look at PHP source code

Let's now open the source code and oversee it. We'll here give you hints about the main components, the extensions and how all this stuff is linked together to make PHP what it is.

- What is PHP (from an inside look)
  - The different components
    - Zend Engine
    - SAPI Layer
    - PHP Core
- Extensions
  - What is an extension ?
    - Everything in PHP is extension !
  - How to enable/disable an extension ?

    - Difference between zend_extension and extension
    - Extensions inter-dependencies and incompatibilities
- What is a VirtualMachine ?
    - Concepts
    - Steps : setup- lexing+parsing - compiling - execution - teardown (no need to go too much deep here)
- Comparison of PHP internal-design to other languages
    - Python / Ruby and Java VM comparison would be a great +1

PHP ecosystem : Preparing PHP source code

Now that we know main tools to develop and build a C project, let's meet some more accurate tools dealing with PHP source code. They are mainly based on the general-purpose tools we introduced you, but you have to know them as you'll have to deal with them when building PHP

- Difference between OS packages and raw source code
- Compiling PHP
    - Configuring the sources (configure script)
        - Debug mode
        - External libraries dependencies
    - Adding/removing extensions
        - Statically
        - Dynamically (DSOs)
        - Impact on the memory footprint
        - checking for extensions
    - Compilation problems
    - Managing several PHP versions
- List of PHP extensions
    - PECL introduction
    - phpize tool

First step into code : Zend Memory Manager

Nobody can start programming inside PHP without knowing this layer. So we'll start by introducing how to manage memory within PHP using the Zend Memory Manager component

- What is ZMM and why we need that layer ?
    - Common Memory Heap problems
    - memleaks reporting
    - Memory consumption with/without ZendMM
- Enabling/disabling ZendMM. Configuration
    - ZEND_MM_SEG_SIZE
    - ZEND_MM_MEM_TYPE
    - USE_ZEND_ALLOC
- Exported functions (emalloc() etc...)
    - Memory leak checking
    - Invalid write detection
    - Using Valgrind with ZMM
- Persistent VS non persistent allocations
- ZendMM internals
    - structures

    - internals stuff
        - alignment
        - data storage classes
        - pointer reuse
        - watchdog and cookies
- Interacting with ZendMM through PHP land
    - memory_usage() and peak usage
    - memory limit
- PHP extensions dedicated for memory management
    - ext/memtrack

Zvals

Now that we know how to manage the memory into PHP source code, let's see how we can create PHP variables from an internals point of view. zvals are C structures responsible of PHP variables, but not only. You really need to master that special type management in order to play with PHP internals

- zval struct as tagged union
- Memory management of PHP variables
    - reference counting
    - copy on write
    - references
    - garbage collector
- access macros
    - Z_TYPE*, Z_*VAL*
- zval allocation and initialization
    - ZVAL_ALLOC(_REL), INIT_ZVAL, ALLOC_INIT_ZVAL, MAKE_STD_ZVAL
- setting a typed value
    - ZVAL_(NULL|LONG|...)
- reference count macros, destruction, deallocation
    - Z_ADDREF*, Z_DELREF*, Z_(SET_)REFCOUNT*, Z_*ISREF*
    - zval_ptr_dtor, zval_dtor
- copying and zval separation
    - zval_copy_ctor, MAKE_COPY_ZVAL
    - SEPARATE_ZVAL(_IF_NOT_REF|_TO_MAKE_IS_REF)?
- casts
    - is_numeric_string
    - convert_to_*, convert_to_*_ex

Creating PHP extensions

You have know enough knowledge to begin creating your first extension. We'll reuse all you've learnt so far and put it in a practical case. We'll then start writing new PHP function in C and then cover the PHP functions scope from an internal point of view.

- zend_extension VS extension
    - What are the differences ?(zend_extension VS zend_module_entry structs)
    - Overview of an extension skeleton
- extension loading mechanism

    - extension dependencies
    - dl()
    - API and binary compatibility checking
- Automatic tools
    - Building a skeleton with ext_skel
- building / activating the extension
    - config.m4
    - phpize
    - php_config.h
- Extension lifecycle
    - MINIT, RINIT, RSHUTDOWN, MSHUTDOWN, MINFO
    - globals management and TSRM
- My first usefull function

Functions

You've just built your first extension, and it works ! That's great ! Now we will dive deeper into a very important concept : PHP functions. We'll cover the subject showing how to declare functions, how to accept parameters from them and how to make them return values.

- zend_function_entry
    - Anatomy of a PHP function from internal point of view
    - PHP_FUNCTION definition
    - PHP_FE registration
- Argument information
    - Accepting parameters
    - zend_arg_info structure
    - ZEND_*_ARG_INFO*
- Parameter parsing
    - zend_parse_parameters() and family
- Returning values from your functions
    - RETURN_*, RETVAL_* macros
- Error handling
    - Error reporting, php_error_docref*
    - correct freeing in case of errors

Hash tables and arrays

- Hash table basics
- Bucket and HashTable structures
- Allocation and initialization, cleaning, destruction, deallocation
    - ALLOC_HASHTABLE, zend_hash_init, array_init
    - zend_hash_clean, zend_hash*_destroy, FREE_HASHTABLE
- Basic zend_hash_* operations
    - zend_hash_num_elements
    - update, add, insert, del, find (with quick and index variations)
- Array APIs
    - add_assoc_*, add_index_*, add_next_index_*, add_get_index_*
- Iteration
    - HashPosition, zend_hash_move_*, zend_hash_get_current_*, zend_hash_internal_pointer_*

    - Avoiding segfaults due to modification-during-iteration, internal vs external array pointer
    - Application with zend_hash_apply*
- Use of HashTable for other purposes
    - Symtables

Classes + Objects

- Class entry registration
    - INIT_CLASS_ENTRY, zend_register_internal_class
    - Extending classes
    - Implementing interfaces, zend_class_implements
    - ce_flags
- PHP_METHOD definition, PHP_ME registration, ZEND_ACC_*, getThis()
- Constant declarations
    - zend_declare_class_constant*
- Properties
    - Declaration, zend_declare_property*
    - Updating, zend_update_property*, zend_update_static_property*
    - Reading, zend_read_property, zend_read_static_property
    - ??? zend_add_property*
- Zend object store and custom object structs
    - create_object handler
        - zend_object_std_init
        - zend_objects_store_put
        - dtor, free and clone storage handlers
    - zend_object_store_get_object
- Object handlers, class handlers
    - Cloning
        - clone_obj
        - zend_objects_store_clone_obj
    - Serialization
        - get_properties
        - serialize, unserialize class handlers
            - zend_class_serialize_deny, zend_class_unserialize_deny
        - __wakeup, __sleep
    - Iteration
        - get_iterator handler
        - iteration functions, iterator_funcs.funcs
    - var_dump etc debugging information
        - get_properties
        - get_debug_info
    - Overloads
        - Property access: read_property, write_property, has_property, unset_property, get_property_ptr_ptr
        - Array access: read_dimension, write_dimension, has_dimension, unset_dimension
        - Methods: get_method, call_method, get_constructor, get_static_method
        - Casts: cast_object
        - count(): count_elements
        - Comparison: compare_objects

    - Other:
        - add_ref, del_ref, get, set, get_class_entry, get_class_name, get_closure_get_gc
    - All handlers: http://lxr.php.net/xref/PHP_TRUNK/Zend/zend_object_handlers.h#29
    - How to respect magic methods provided by user inheritance
- Preventing crashes from from missing ctor call

Resources

- Recall on what resources are in PHP
- zend_list structure
- API to create/remove resources
- persistent resources VS non persistent

Network streams

Constants

- zend_constant structure
- manipulating constants

INI and configuration management

- How the engine gets its configuration
    - Difference between configuration and ini params
    - ini files loading
    - ini modification levels
    - Keeping track of original ini value
- Reading and displaying ini directives
    - ini types and conversions
    - binding ini directive to a global
    - ini displayers
- Adding ini entries to your extension
    - PHP_INI_BEGIN() and other macros to help
- Quick look at the ini parser

SAPIs

- Recall on SAPI : PHP entry point
- List of different existing SAPI
- SAPI main structures
    - sapi_module_struct, sapi_headers_struct, sapi_global_struct, sapi_request_info
- SAPI integration into PHP
    - sapi_startup(), sapi_activate() and shutdown
- output buffering
- Creating a SAPI
    - Example with embed SAPI
    - Linking PHP to another program

Zend Engine

- Lexer
  - lexing and re2c basics, tokenizer ext
  - semantic values, zendlval
  - YY stuff
    - yytext, yyleng, YYCURSOR, YYLIMIT, YYMARKER
    - yy_pop_state, yy_push_state, BEGIN, YYSTATE, YY_SET_CONDITION, YY_GET_CONDITION
    - yyless, yymore
- Parser
  - parsing and bison basics, LALR(1)
  - token declarations, precedence, associativity
  - semantic actions, magic $* variables
  - znode structure
- Compiler (zend_do_* functions)
  - Structures: zend_op, zend_op_array, znode_op
  - get_next_op, get_temporary_variable
  - SET_NODE, GET_NODE, SET_UNUSED
- Zend VM
  - VM basics, VM kinds, zend_vm_gen.php
  - Vulcan Logic Disassembler
  - ZEND_VM_HANDLER definitions
  - Node types: IS_CONST, IS_CV, IS_VAR, IS_TMP_VAR, IS_UNUSED
  - The VM macros and pseudo-macros
    - USE_OPLINE, SAVE_OPLINE
    - Fetching vars: zend_free_op*, GET_OP*_ZVAL_PTR(_PTR), FREE_OP*
    - ZEND_VM_INC_OPCODE, ZEND_VM_NEXT_OPCODE, ZEND_VM_SET_OPCODE
    - CHECK_EXCEPTION, HANDLE_EXCEPTION
    - ZEND_VM_CONTINUE, ZEND_VM_RETURN, ZEND_VM_LEAVE
    - … … ...
  - (VM helpers, ZEND_VM_DISPATCH_TO_HELPER)
- Zend VM Implementation details
  - executor globals
  - zend_execute_data structure
  - CV and T storage, symbol table optimizations
  - VM stack
  - function calls, argument pushing
  - … … …

- function/method calls
  - zend_call_function(), zend_fcall_info, zend_fcall_info_cache
  - zend_internal_function VS zend_user_function
  - scopes
  -

Final Thoughts

- The PHP History
    - From 1995 to nowadays
- The PHP People
    - Main contributors
- Where to look for help
    - php.net resources (Zend API Reference)
    - IRC and mailing lists
    - Other resources
-