Morpher

API Documentation

November 28, 2011

Contents

1 Package morpher

Contains the modules used for the Morpher API-fuzzing utility

(GRAPH)

Morpher is a API fuzzing tool for Windows Dynamically Linked Libraries (DLLs). Morpher's methods are based around two major ideas:

1. Mutational fuzzing - Many fuzzers either generate valid data for the API calls, which requires the fuzzer to understand the API and how it is used, or generate completely random data for API calls, which is often invalid and does not exercise any of the DLL's code beyond preliminary input sanitization. Morpher takes a different route by running programs supplied by the user that use the DLL functions and "capturing" these function calls as they occur. The arguments to these function calls are then fuzzed individually and the function calls are replayed to the DLL. This method of "mutating" known valid calls is able to generate invalid calls that are more likely to pass input checking without requiring any knowledge of the API beforehand.
2. Understanding argument types - Some function parameters are more complex than others - for example, an argument might include a pointer to a structure in memory, which may contain more pointers, etc. If we don't acknowledge these relationships we may fail to fuzz values passed to the function that aren't the actual arguments. In addition, knowing the types associated with the data allows us to fuzz it intelligently - for example if we know a value is an integer, we can fuzz it mutationally by negating the value, or heuristically by replacing it with the maximum representable positive value.

Morpher accepts a target DLL, the header files for the DLL, and a list of programs that use the DLL. It parses the header files to create a model of the DLL's function prototypes, runs the given programs and records their function calls, then fuzzes and replays those function calls to the DLL and monitors the function call for signs of a crash or hang. Crashes and hangs are recorded with enough information that they can be inspected in detail by a reverse engineer to determine the cause of the problem.

Morpher is designed as a black-box package, so a Morpher object can be instantiated and used in an application, or the whole tool can be used from the command line by a minimal script wrapper. Most of Morpher's functionality is controlled by a central Config object, which is controlled in turn by the data in a configuration INI file.

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 21, 2011

1.1 Modules

- **collector**: Contains the modules used for Morpher's data collection phase.
  (Section 2, p. 19)
  - **collector**: Contains the `Collector` class for recording DLL function calls by a list of programs.
    (Section 3, p. 20)
  - **func_recorder**: Contains the `FuncRecorder` class for recording a `Snapshot` of a function call
    (Section 4, p. 22)
  - **range_union**: Contains the `RangeUnion` class for maintaining a "covering" set of ranges

2 Package morpher.collector

Contains the modules used for Morpher's data collection phase.

(GRAPH)

The overall purpose of these modules is to run a series of programs which use functions exported by a target DLL, hook those functions and take a "snapshot" of relevant parts of the stack at the moment of the function call, recording the data so the function call can be replayed later in its entirety.

`RangeUnion` is a utility class that implements a data structure for managing ranges - the idea is that after adding a large number of potentially overlapping ranges to the `RangeUnion`, it will return a minimal set of ranges that has the same coverage as the ranges it was given, no more or less, and with no overlapping members. This is used to make sure that no part of memory is copied twice when taking a `Snapshot`. The `SnapshotManager` is a class that uses the `RangeUnion` to make sure the `Snapshot` it creates doesn't copy more memory than necessary, and carries out the copying of memory at the moment the `Snapshot` is created. `FuncRecorder` is responsible for actually walking through the stack of a function call and identifying areas that need to be recorded, while `TraceRecorder` is responsible for setting up the program and hooking the function calls to be recorded. The whole process is coordinated by the top-level `Collector` object and is highly dependent on the information output in model.xml by the parser.

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 22, 2011

2.1 Modules

- **collector**: Contains the `Collector` class for recording DLL function calls by a list of programs.
  (Section 3, p. 20)
- **func_recorder**: Contains the `FuncRecorder` class for recording a `Snapshot` of a function call
  (Section 4, p. 22)
- **range_union**: Contains the `RangeUnion` class for maintaining a "covering" set of ranges
  (Section 5, p. 26)
- **snapshot_manager**: Contains the `SnapshotManager` class for creating a `Snapshot`
  (Section 6, p. 28)
- **trace_recorder**: Contains the `TraceRecorder` class for creating a `Trace` by observing a program using a DLL
  (Section 7, p. 32)

3 Module morpher.collector.collector

Contains the `Collector` class for recording DLL function calls by a list of programs.

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 23, 2011

3.1 Variables

Name	Description
__package__	**Value:** `'morpher.collector'`

3.2 Class Collector

object
└ **morpher.collector.collector.Collector**

Class documentation

3.2.1 Methods

__init__(*self*, *cfg*)

Stores the configuration object and initializes the internal data

Parameters

`cfg`: The configuration object to use

(type=`Config` object)

Overrides: object.__init__

collect(*self*)

The top-level collection routine.

If collection is disabled according to the configuration object, a message saying so is printed to the console and the method exits. Otherwise the directory for storing traces ("data races") is cleared out and the specified list file and model file are read from the filesystem. The collector reads in each line of the listfile, parses it, then uses a `TraceRecorder` object to launch the specified program and create a `Trace` object with the contents of the function calls executed by that program.Each `Trace` is pickled and stored to the trace directory.

parseline(*self*, *line*)

Given a command line string, returns a pair of strings (path, args) where path is a path to a valid file and args is the rest of the string.

Takes a line such as "C:\Program Files\Test est.exe -v -f myfile" and tries to parse it into a tuple consisting of the file being executed and the arguments - in this case the correct return value would be ("C:\Program Files\Test est.exe", "-v -f myfile"). The parsing is performed by tokenizing the string using whitespace, then concatenating each token to the beginning token and testing the result to see if it is a path to a valid file.

Parameters

`line`: The string to parse

(type=string)

Return Value

(pathtoexe, args) or (None, None) if line couldn't be parsed

(type=(string, string) pair)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

3.2.2 Properties

Name	Description
Inherited from object	
__class__	

3.2.3 Instance Variables

Name	Description
cfg	The `Config` object
counter	The number of traces recorded so far
log	The `logging` object
model	The XML root `Node` for the DLL model
modelpath	The path to the XML model file
tracedir	The path to the directory to store `Trace` files in

4 Module morpher.collector.func_recorder

Contains the `FuncRecorder` class for recording a `Snapshot` of a function call

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: November 14, 2011

4.1 Variables

Name	Description
__package__	**Value:** `'morpher.collector'`

4.2 Class FuncRecorder

object
└ **morpher.collector.func_recorder.FuncRecorder**

Used to capture the state of a function call using a supplied debugger.

The `FuncRecorder` class contains enough information to be able to interpret and traverse the stack of another process, captured at the moment of a function call, and pinpoint areas of the stack that need to be captured and recorded in order to reproduce the function call at a later date. The tagging process used by the `FuncRecorder` is designed to minimize the amount of memory that is copied for the capture, while handling complex cases such as structures/unions and capturing the content referenced by pointers. The end goal is to produce a `Snapshot` object that reproduces the exact same function call and contains all the available information about the types of the objects captured (so they can be intelligently fuzzed).

4.2.1 Methods

__init__(*self*, *cfg*, *model*)

Stores the given config object for local configuration information and initializes the instance variables. The local type manager is initialized using the supplied model data, which is also used to traverse the stack of the function being recorded.

Parameters

`cfg:` The configuration object to use

*(type=**`Config`** object)*

`model:` The root node of the XML DLL model

*(type=**`Node`** object)*

Overrides: object.__init__

record(*self*, *dbg*, *name*)

Activated upon a function call and used to record the stack.

Should be called when the target process is paused by a debugger at the beginning of a function call. Uses the supplied debugger object to start the snapshot process and accesses the target process's memory space to capture the function arguments.

Parameters

`dbg:` The debugger that should be used to access memory

*(type=**`pydbg`** object)*

`name:` The name of the function we are recording

(type=string)

Return Value

The filled snapshot containing the image of this function call

*(type=**`Snapshot`** object)*

tagArgs(*self*, *addr*, *funcnode*)

Starts the recursive tag process for this function's args.

Given the function XML's model and the address of the arguments, walks through the arguments and tags each one using this object's snapshot manager.

Parameters

`addr:` Address the function arguments start at on the stack

(type=integer)

`funcnode:` XML `Node` for the function

(type=`Node` object)

Note: We can't rely on the arguments being properly aligned - they only need to be aligned to the stack requirements.

tag(*self*, *addr*, *paramtype*)

Given an address and a type. either basic (ex. "i") or user-defined (ex. "1"), tag the object for collection and recursively tag any member objects or objects it points to.

If the type is user-defined (for example, "1" indicates a user-defined type such as a struct), the type's definition is looked up using the model and the fields of the type are individually tagged. If the type is a basic type, the type is tagged. If the type is a pointer type, such as "PPI", a pointer tag ("P") is added and the tagging is recursively performed on the type pointed to ("PI") at the address contained in the pointer type.

Parameters

`addr:` The address of the object to tag

(type=integer)

`paramtype:` The format string representing the object's type

(type=string)

Note: The tagging algorithm is designed to only record a tag once, and handle pointer loops, pointers to the same object but as different types, and other complications.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

4.2.2 Properties

Name	Description
Inherited from object	
__class__	

4.2.3 Instance Variables

Name	Description
cfg	The `Config` object
dbg	The `pydbg` debugger
log	The `logging` object
model	The XML root `Node` for the DLL model
sm	`SnapshotManager` object for creating image
stack_align	The alignment requirement for the stack
type_manager	The `TypeManager` used for type information

5 Module morpher.collector.range_union

Contains the `RangeUnion` class for maintaining a "covering" set of ranges

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 26, 2011

5.1 Variables

Name	Description
__package__	**Value:** `'morpher.collector'`

5.2 Class RangeUnion

object
└ **morpher.collector.range_union.RangeUnion**

Used to maintain a list of ranges

The class is designed to solve the problem where a list of ranges is given and needs to be "simplified" to an equivalent list with the minimum possible number of ranges and no overlaps. The range list is maintained as the instance variable rlist, and rlist is updated each time a new range is added with the `add` method.

Ranges are represented by the "Range" `namedtuple`

Invariant: Intervals in the range list do not overlap, are in sorted order from lowest address to highest, and for any two consecutive ranges in the list there is a separation of at least 1 between the ending address of the first and the beginning address of the second.

To Do: Improve performance of the RangeUnion - use a tree structure?

5.2.1 Methods

__init__(*self*, *startlist*=`None`)

Takes an optional argument that allows this `RangeUnion` to be initialized from an existing range list, otherwise empty.

Parameters

`startlist`: The list of ranges to be initialized from

(type=Range object list)

Overrides: object.__init__

add(*self*, *c*)

Given a Range c, adds c to the list of ranges, then merges any overlapping members so the list retains the equivalent range information but remains sorted and non-overlapping. Note that the ranges are at integer granularity, so the range (1, 4) and range (5, 7) can be merged to range (1, 7)

Parameters

`c`: Range to add

(type=Range object)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

5.2.2 Properties

Name	Description
Inherited from object	
__class__	

5.2.3 Instance Variables

Name	Description
rlist	A list of Range objects

6 Module morpher.collector.snapshot_manager

Contains the `SnapshotManager` class for creating a `Snapshot`

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 26, 2011

6.1 Variables

Name	Description
__package__	**Value:** `'morpher.collector'`

6.2 Class SnapshotManager

object
└
morpher.collector.snapshot_manager.SnapshotManager

Designed to simplify the process of properly creating a `Snapshot` object.

The `SnapshotManager` provides a simple interface consisting of functions like `addArg` and `addObject` and handles the more complex issues in the background, such as using sets to ensure the uniqueness of `Tag`s added to the `Snapshot` and `RangeUnion` objects to ensure that the minimal amount of memory is copied for the `Snapshot`. The contents of the target process memory are not copied until the `snapshot` method is called, which uses all the information accumulated to record areas of memory using the debugger and create the requested `Snapshot`

6.2.1 Methods

__init__(*self*, *cfg*, *dbg*, *name*)

Stores the configuration object, a pydbg debugger attached to the target process, and the name of the function being captured.

Parameters

`cfg`: The configuration object to use

(type=`Config` object)

`dbg`: The debugger used to access the target process

(type=`pydbg` object)

`name`: The name of the function being called

(type=string)

Overrides: object.__init__

addArg(*self*, *addr*, *fmt*)

Add a tag (start, fmt) to our list of argument tags

Parameters

`addr`: Address of the argument being added

(type=integer)

`fmt`: Format string representing the argument type

(type=string)

checkObject(*self*, *addr*, *fmt*)

Returns True if the tag (addr, fmt) is already in the manager

Parameters

`addr`: Address of the object being checked

(type=integer)

`fmt`: Format string representing the object type

(type=string)

Return Value

True if tag already registered, *False* otherwise

(type=Boolean)

addObject(*self*, *start*, *size*, *fmt*)

Adds the memory range from start to start + (size of fmt) -1 to the list of areas to capture. A (start, fmt) tag is added for the object.

Parameters

`start`: Address of the object being added

(type=integer)

`size`: Size of the object being added

(type=integer)

`fmt`: Format string representing the object type

(type=string)

snapshot(*self*)

Uses the debugger to record the requested areas of the process's memory and returns the contents as a new `Snapshot` object. The `Snapshot` is populated using the tags registered using `addObject` and the arguments added using `addArg`.

Return Value

The newly created `Snapshot`

(type=`Snapshot` object)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

6.2.2 Properties

Name	Description
Inherited from object	
__class__	

6.2.3 Instance Variables

Name	Description
args	Ordered list of `Tag`s corresponding to the function arguments
cfg	The `Config` object
dbg	The `pydbg` debugger for capturing memory
log	The `logging` object

continued on next page

Name	Description
name	Name of the function we are recording
ru	`RangeUnion` object used for ensuring that the minimum necessary amount of process memory is captured
tset	The set of `Tag` objects for this capture

7 Module morpher.collector.trace_recorder

Contains the `TraceRecorder` class for creating a `Trace` by observing a program using a DLL

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 28, 2011

7.1 Variables

Name	Description
__package__	**Value:** `'morpher.collector'`

7.2 Class TraceRecorder

object
└ **morpher.collector.trace_recorder.TraceRecorder**

Used to run a program and produce a `Trace` object that can replay all function calls to the target DLL observed during the program run. Stores enough information to hook function calls using the supplied model and configuration objects and uses a `FuncRecorder` to do the actual stack recording.

7.2.1 Methods

__init__(*self*, *cfg*, *model*)

Stores the configuration object and model for local use and initializes other instance variables

Parameters

`cfg`: The configuration object to use

(type=`Config` object)

`model`: The root node of the XML DLL model

(type=`Node` object)

Overrides: object.__init__

record(*self*, *exe*, *arg*)

Given an application that uses the target DLL, runs the program and captures a `Trace` object thats capable of replaying all the function calls made by the application to the DLL.

The `Trace` is captured by launching the application in a second process and setting breakpoints at the beginning of each of the functions in the DLL. The application is allowed to run and if any of the breakpoints are tripped, a `FuncRecorder` is used along with the debugger to capture all relevant areas of the stack. Each `Snapshot` is stored in the created `Trace` in the same order that they were captured in.

Parameters

`exe`: The path to the application to record.

(type=string)

`arg`: List of command-line arguments for the program

(type=string)

Return Value

A `Trace` containing the captured function calls

(type=`Trace` object)

checkTimeout(*self*, *dbg*)

Checks for timeouts, in which case it logs the hang and terminates the process. This function should be set as the handler for the debugger's event loop (called at least every 100ms)

Parameters

`dbg`: The debugger that should be used to access memory

(type=`pydbg` object)

timeoutHandler(*self*)

Sets the timed_out flag. This function should be automatically called after (self.limit) seconds have elapsed since the beginning of the currently running program.

loadHandler(*self*, *dbg*)

Goes through the functions listed by the xml model and sets breakpoints at each function's entry point.

This function should be set as the handler for DLL load events detected by the debugger. It sets a breakpoint for each function whose handler is designated as the `funcHandler` function, and the breakpoint description is set to the ordinal of the function so `funcHandler` can identify it.

Parameters

`dbg`: The debugger that should be used to access memory

(type=`pydbg` object)

Return Value

Handler return code from `defines` module

(type=integer)

funcHandler(*self*, *dbg*)

Activated upon a function call. Determines which function was called and starts the snapshot process to capture the function arguments using the `FuncRecorder` object. The created `Snapshot` is appended to the end of the self.trace list.

Parameters

`dbg`: The debugger that should be used to access memory

(type=`pydbg` object)

Return Value

Handler return code from `defines` module

(type=integer)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

7.2.2 Properties

Name	Description
Inherited from object	
__class__	

7.2.3 Instance Variables

Name	Description
cfg	The `Config` object
dllpath	Path to the target DLL
func_recorder	`FuncRecorder` object used for stack capture
limit	The number of seconds a program can run before its considered to have timed out
log	The `logging` object
model	The XML root `Node` for the DLL model
trace	The list of `Snapshot` objects to turn into a `Trace`

8 Package morpher.fuzzer

Contains the modules used for Morpher's fuzzing phase.

(GRAPH)

The overall purpose of these modules is to take the `Trace` files generated by the collection phase, modify the recorded values in the traces according to their types, then replay those function calls in a new process and monitor the process for signs of hangs or crashes. If a failure is detected, the appropriate crash information is stored along with the modifed trace so it can be reproduced as needed.

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 22, 2011

8.1 Modules

- **fuzzer**: Contains the `Fuzzer` class for controlling Morpher fuzzing phase
  (Section 9, p. 37)
- **generator**: Contains the `Generator` class for building collections of fuzzed values
  (Section 10, p. 40)
- **harness**: Contains the `Harness` class for replaying function calls from a `Trace` object in a controlled environment.
  (Section 11, p. 43)
- **monitor**: Contains the `Monitor` class for launching and monitoring `Harness` tasks
  (Section 12, p. 46)

9 Module morpher.fuzzer.fuzzer

Contains the `Fuzzer` class for controlling Morpher fuzzing phase

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 28, 2011

9.1 Variables

Name	Description
__package__	**Value:** `'morpher.fuzzer'`

9.2 Class Fuzzer

object
└
morpher.fuzzer.fuzzer.Fuzzer

Top-level class in charge of reading in stored `Trace`s, fuzzing their contents, replaying them back and recording the results.

Most of this class's functionality is reading in `Trace` files from the appropriate directory, then iterating through each `Tag` for each `Trace`. An internal `Generator` object is used to get a list of fuzzed value for each `Tag`, and each fuzzed value is used to overwrite the original value in turn. Each changed version of the `Trace` is given to a `Monitor` object for playback, and after all versions have been replayed the original value is restored and the entire process is repeated for the next tag.

To Do:

- Possibly expand fuzzing to multiple tags at once
- Possibly generate `Trace` for functions we didn't actually collect any data for.
- Possibly look at fuzzing global variables
- Possibly fuzz ORDER of function calls, not just data
- Special fuzzing for arrays and buffers?

9.2.1 Methods

__init__(*self*, *cfg*)

Store the configuration object, create a `Generator` object, and set the internal trace number to 0.

Parameters

`cfg`: The configuration object to use

(type=`Config` object)

Overrides: object.__init__

fuzz(*self*)

Runs the entire fuzzing process.

If fuzzing is disabled according to the configuration object, this function prints a message saying so and returns. Otherwise the data races directory is searched for `Trace` files, and each one is read into memory.

For each `Snapshot` in each `Trace`, the list of `Tag`s is extracted. For each `Tag`, the original value is saved and used to create a list of fuzzed values from the `Generator`. For each fuzzed value, the fuzzed value is used to overwrite the original value, and the modified `Trace` is fed to the `Monitor` for replay. After every fuzzed version has been run, the original value is restored and fuzzing moves on to the next tag.

Notes:

- For any fuzzed `Trace`, only one value is changed from the original version.
- A `SectionReporter` object is instantiated and used to track the overall progress for the user.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

9.2.2 Properties

Name	Description
Inherited from object	
__class__	

9.2.3 Instance Variables

Name	Description
cfg	The `Config` configuration object for this `Fuzzer`
generator	The `Generator` object used for fuzzing `Trace` values
log	The `logging` object for this `Fuzzer`
monitor	The `Monitor` object used for replaying `Trace`s.
tracenum	The number identifying the current `Trace`

10 Module morpher.fuzzer.generator

Contains the `Generator` class for building collections of fuzzed values

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 30, 2011

10.1 Variables

Name	Description
__package__	**Value:** `'morpher.fuzzer'`

10.2 Class Generator

object
└
morpher.fuzzer.generator.Generator

Used to generate lists of values created by mutating a given value, using lists of heuristic values based on type, and values chosen at random.

This class operates by accepting a `struct` style format character and an original value. A map is used to match the format string type to the appropriate generator function - for example, unsigned integer types of all sizes are fuzzed using the `_getUints` function. The individual generator functions return the values as a set, ensuring that no value is repeated, which is turned into a list and returned to the caller. Values are generated according to methods specified in the configuration.

To Do:

- Use memoization to increase performance
- Examine fuzzing algorithms for possible improvement

10.2.1 Methods

__init__(*self*, *cfg*)

Store the configuration object, initializes instance variables using configuration data, and sets up the generator map.

Parameters

`cfg`: The configuration object to use

(type=`Config` object)

Overrides: object.__init__

generate(*self*, *fmt*, *orig*)

Takes a type and a value and returns a list of fuzzed values, which is created using some or all of the methods listed:

1. Mutational: the original value is modified to return values that are similar to the original, such as by adding or subtracting small amounts (for integers)
2. Heuristic: a predetermined list of values is used, where the values are chosen based on a high likelihood of creating problems for programs that do not adequately check their inputs. For example, if a floating-point type is being fuzzed, NaN and Inf values might be used, since programs that accepts floats may not correctly handle these special and uncommon values.
3. Random: several values are chosen at random from the set of legal values for this type.

Parameters

`fmt`: The format string of the type we are fuzzing

(type=string)

`orig`: The original value of the object we are fuzzing

(type=basic Python value (string, integer, etc))

Return Value

List of fuzzed values

(type=basic Python value (string, integer, etc) list)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

10.2.2 Properties

Name	Description
Inherited from object	
__class__	

10.2.3 Instance Variables

Name	Description
cfg	The `Config` configuration object for this `Monitor`
generators	Map of format strings to appropriate generator function
heuristic	Boolean indicating if heuristic values should be used
log	The `logging` object for this `Monitor`
mutaterange	Integer indicating the range of values around the original value should be produced for mutational fuzzing
mutational	Boolean indicating if mutational values should be used
randcases	Number of values chosen at random to produce
random	Boolean indicating if random values should be used

11 Module morpher.fuzzer.harness

Contains the `Harness` class for replaying function calls from a `Trace` object in a controlled environment.

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 26, 2011

11.1 Variables

Name	Description
__package__	**Value:** `'morpher.fuzzer'`

11.2 Class Harness

object

multiprocessing.process.Process

morpher.fuzzer.harness.Harness

Works as a seperate process which accepts a `Trace`, loads a specified DLL, and uses the trace to replay a function call to the DLL.

A fuzzer can't replay a `Trace` in the same process because if the process crashes it will take down the fuzzer as well. Using the `Harness` allows the trace to be replayed in a seperate process, and if the process crashes a debugger can observe the crash and log any debug data. The `Harness` is deliberately kept as simple as possible - it merely accepts a `Trace` from a pipe, replays each `Snapshot` in the trace and reports successful completion for each using the pipe, then exits.

Just before each `Snapshot` is replayed, the harness sends a *True* value over the pipe it was given. This allows the process that owns the other end of the pipe to keep track of the `Trace` replay's progress, so the exact `Snapshot` that triggers a crash or hang can be pinpointed.

Note: `_kill_output` is used to suppress any output to standard output or standard error streams by the DLL during replay.

11.2.1 Methods

__init__(*self*, *cfg*, *pipe*)

Sets up the given input/output pipes and stores the config, which needs to be serializable.

Parameters

`cfg`: The configuration object with target and logging info

(type=`Config` object)

`pipe`: A pair of `multiprocessing` connections (input, output)

(type=(Connection, Connection) tuple)

Overrides: object.__init__

Warning: This code is still in the same process as the object creator

run(*self*)

Sets the seperate process running, waits for a `Trace`, and replays the trace for the specified DLL.

A seperate logging root is set up, with logging going to a different file, so two processes won't be trying to write to the same file. The target DLL is loaded using information in the `Config`, then the `Harness` waits for a `Trace` to be received over the pipe.

After receiving the `Trace`, the standard output is disabled and the trace is replayed one call at a time, with the value *True* being sent back over the pipe before each call to the DLL. Once the replay is complete the pipes are closed and the process exits.

Overrides: multiprocessing.process.Process.run

Inherited from multiprocessing.process.Process

__repr__(), is_alive(), join(), start(), terminate()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

11.2.2 Properties

Name	Description
Inherited from multiprocessing.process.Process	

continued on next page

Name	Description
authkey, daemon, exitcode, ident, name, pid	
Inherited from object __class__	

11.2.3 Instance Variables

Name	Description
cfg	The configuration object
inpipe	The connection used to receive a `Trace` object
outpipe	The connection used to send "pings" back to the parent

12 Module morpher.fuzzer.monitor

Contains the `Monitor` class for launching and monitoring `Harness` tasks

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 21, 2011

12.1 Variables

Name	Description
__package__	**Value:** `'morpher.fuzzer'`

12.2 Class Monitor

object
└ **morpher.fuzzer.monitor.Monitor**

Used for running `Trace` replays using a `Harness` and monitoring the `Harness` for crashes, etc.

Acts as a controller for `Harness` objects - each time `run` is called, a new `Harness` is spawned, a debugger is attached, and the `Trace` is sent to the `Harness` for replay. If a problem is noted by the debugger, relevant crash information is assembled and dumped to a predetermined directory. Right now two types of "problems" are detected - segmentation faults (access protection violation) and hangs over a certain time limit.

12.2.1 Methods

__init__(*self*, *cfg*)

Takes a config object and sets up Monitor. If data/crashers doesn't exist, the directory is created, otherwise all directories inside that start with "address-" are erased. If data/hangers doesn't exist, the directory is created, otherwise all file entries that start with "trace-" and end with ".txt" or ".pkl" are erased.

Parameters

`cfg`: The configuration object

(type=`Config` object)

Overrides: object.__init__

setTraceNum(*self*, *tracenum*)

Change the trace number used for naming dump files. Automatically sets the iteration number back to 0

Parameters

`tracenum`: The new number to use to identify this `Trace` batch

(type=integer)

run(*self*, *trace*)

Takes the `Trace` and runs it in a `Harness`, monitoring for crashes.

This function spawns a new process using a `Harness` object connected to this process by a pair of pipes. A debugger is attached to the `Harness` process and handlers are attached to monitor for crashes and hangs (defined as the harness not completing by a certain time limit). The given `Trace` is then sent over the pipe to the `Harness` for replay, and the `Harness` is watched for completion. If a crash or hang occurs, relevant information is collected and dumped to a file for inspection and possible reproduction.

Each `Trace` is identified as a certain run (the iteration) of a certain batch (the trace number), and this identification is reflected by the file name if a dump occurs. The scheme is based off the common fuzzing pattern of taking one "base" trace and fuzzing the values in it to create multiple fuzzed versions - so a batch is all traces that were generated by fuzzing the same base trace.

Parameters

`trace`: The trace to run and monitor

(type=`Trace` object)

timeout(*self*)

Sets the timed_out flag. This should be called with a timer after (self.limit) seconds

time_check(*self*, *dbg*)

Checks for timeouts, in which case it logs the `Trace` as a "hanger" and terminates the process.

This function should be set up as the handler for the debugger's event loop (which is called at least every 100ms). This function checks if a timeout has occurred and if so, drops any `Snapshot`s from the offending `Trace` that weren't called before the timeout occurred, and dumps the information to the "hangers" directory.

Two files are created: a text (.txt) file with the human-readable contents of the `Snapshot`s that lead to the hang, and a pickle (.pkl) file with the same name that contains a pickled version of the hanging `Trace`, which can be replayed in order to reproduce the hang.

Parameters

`dbg`: The debug object this was called from

(type=`pydbg` object)

crash_handler(*self*, *dbg*)

Handles crash events in the `Harness`, logs the information and terminates.

This function should be set up as the handler for segmentation fault events detected by the debugger. This function records the crash information using the `crash_binning` module, drops any `Snapshot`s from the offending `Trace` that weren't called before the crash occurred, and dumps the information to the "crashers" directory, under a sub-directory matching the address of the instruction the crash occurred at.

Two files are created: a text (.txt) file with the human-readable crash information and contents of the `Snapshot`s that lead to the crash, and a pickle (.pkl) file with the same name that contains a pickled version of the crashing `Trace`, which can be replayed in order to reproduce the crash.

Parameters

`dbg`: The debug object this was called from

(type=`pydbg` object)

Return Value

`pydbg.defines` DBG_EXCEPTION_NOT_HANDLED

(type=integer)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

12.2.2 Properties

Name	Description
Inherited from object	
__class__	

12.2.3 Instance Variables

Name	Description
cfg	The `Config` configuration object for this `Monitor`
crashpath	The path to the "crashers" directory
hangpath	The path to the "hangers" directory
iter	The number of `Trace`s run so far for this batch
last_trace	The last `Trace` object sent to a `Harness`

continued on next page

Name	Description
limit	Number of seconds to wait for `Harness` completion before declaring a timeout.
log	The `logging` object for this `Monitor`
tracenum	The number identifying the current batch of `Trace`s

13 Package morpher.misc

Contains various modules that are shared by more than one package or do not fall neatly into the scope of other packages.

(GRAPH)

Currently contains `Config`, a class used to share configuration information between all components of a project; `log_setup`, which contains a simple method that initializes the project-wide logging system; `StatusReporter`, which contains a class for tracking and displaying progress in the form of a status bar; and `SectionReporter`, which builds off of `StatusReporter` to report the progress of a multi-part program.

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 22, 2011

13.1 Modules

- **config**: Contains the `Config` class definition
  (Section 14, p. 52)
- **log_setup**: Contains the `setupLogging` and `translateLevel` function definitions, used for interacting with the standard Python `logging` module
  (Section 15, p. 55)
- **section_reporter**: Contains the `SectionReporter` class definition for reporting progress updates
  (Section 16, p. 58)
- **status_reporter**: Contains the `StatusReporter` class definition for reporting progress updates
  (Section 17, p. 62)

14 Module morpher.misc.config

Contains the `Config` class definition

Author: Rob Waaser

Contact: robwaaser@gmail.com

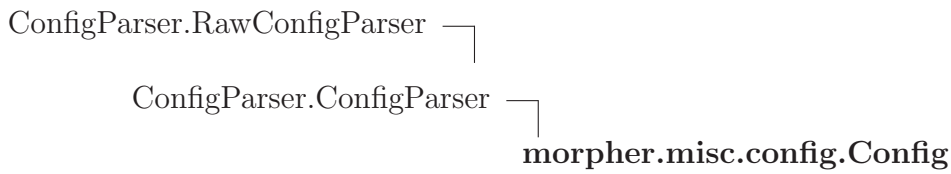
Organization: Carnegie Mellon University

Since: October 22, 2011

14.1 Variables

Name	Description
__package__	**Value:** `'morpher.misc'`

14.2 Class Config



A wrapper for Python's `ConfigParser` class which adds some project-specific configuration and a toString method.

Inherits from Pythons' standard `ConfigParser` class, which is used to read in files in the well-known INI format and parse them for configuration information. Config overrides the `__init__` method with it's own version, which does some project-specific configuration, and also adds a toString method, which returns a pretty-printed string useful for logging the state of this Config object.

Config is designed to be used as a central registry of configuration information for a project, and after it is initialized with the contents of a configuration file, it should be passed to every object in the project that needs to access configuration information. Each object can then use their individual reference to the global Config object to read and write key-value pairs as necessary, which can be seen by all other objects as well.

Note: Config is naturally pickleable as long as no key-value pairs are added that contain pickleable objects - meaning it can be used to store a programs' state to a file and used to later restore that state.

To Do: Add additional validation of parameters read from the config file

14.2.1 Methods

__init__(*self*, ***params*)

Parses a configuration file and any additional keyword parameters to create and initialize a new configuration object.

The __init__ method accepts a list of optional keyword arguments, reads in additional arguments from a configuration file, and also contains a list of default parameter values. The final value of a particular parameter is set to (in order of precedence):

1. The supplied keyword parameter, if one is given
2. The supplied value in the configuration file, if one is given
3. The built-in default value (if one exists for this parameter)

The initialization process does not use the logging system like the rest of Morpher, since the logging system is dependent on configuration information supplied here.

Refer to the documentation for `ConfigParser` for information on how config files are parsed and how key-value pairs can be read and written.

Parameters

`params:`	Override values for optional keyword arguments *(type=keyword options)*
`configfile:`	The path to the configuration file
`debug:`	A boolean value enabling debug mode if *True*
`dll:`	The path to the target dll (no default)
`listfile:`	The path to the collection listfile (no default)

Raises

`Exception` An exception is raised if a needed parameter is not found in the params, config file, or default values.

Overrides: ConfigParser.RawConfigParser.__init__

Note: Config defines the default option "basedir" as the path to the current working directory. Entries in the config file can use this option to refer to other directories relative to the current directory, for example: %(BASEDIR)s\data

toString(*self*)

Returns a pretty-printed string suitable for displaying or logging the contents of this Config object

Returns a string similar to the following:

```
Configuration dump:
[TEMP]
  basedir : C:\Users\Rob\workspace\ApiFuzzing
[directories]
  basedir : C:\Users\Rob\workspace\ApiFuzzing
  data : C:\Users\Rob\workspace\ApiFuzzing\data
  tools : C:\Users\Rob\workspace\ApiFuzzing ools
  logs : C:\Users\Rob\workspace\ApiFuzzing\logs
[logging]
  basedir : C:\Users\Rob\workspace\ApiFuzzing
  enabled : yes
  level : debug
```

Return Value

Nicely-formatted string containing contents of the Config object

(type=string)

Inherited from ConfigParser.ConfigParser

get(), items()

Inherited from ConfigParser.RawConfigParser

add_section(), defaults(), getboolean(), getfloat(), getint(), has_option(), has_section(), options(), optionxform(), read(), readfp(), remove_option(), remove_section(), sections(), set(), write()

14.2.2 Class Variables

Name	Description
Inherited from ConfigParser.RawConfigParser	
OPTCRE, OPTCRE_NV, SECTCRE	

15 Module morpher.misc.log_setup

Contains the `setupLogging` and `translateLevel` function definitions, used for interacting with the standard Python `logging` module

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: November 1, 2011

15.1 Functions

setupLogging(*cfg*, *root*=`None`)

When called with a Config object, uses the Config object to extract configuration information and sets up Python's standard `logging` system for project-wide use.

Initializes the log system provided by Python's `logging` module using information in a provided `Config` object. The log system defines the top-level package in the heirarchy of this module as the root logger by default, or a supplied root can be used instead.

The following actions are performed:

- The logging->enabled option in the cfg object is checked and used to either enable or disable logging globally
- Sets up a handler that prints logging messages of level logging.ERROR or higher to standard output
- Sets up a handler that prints all other logging messages to a log file, located in the directory specified in directories->logging
- Stops propagation of messages above the defined root logger
- Registers an `atexit` handler that ensures the logging system is properly flushed upon program exit.

Parameters

`cfg:` A `Config` object containing logging setup information

(type=`Config` object)

`root:` An optional string specifying the name of the root module

(type=string)

Requires:

- cfg must specify the logging->enabled option
- If logging->enabled is *True*, cfg must specify:
  - the logging->level option
  - the directories->logs option

translateLevel(*string*)

Parameters

`string`: The string to translate to a logging level

(type=string)

Return Value

A corresponding constant from the `logging` module

(type=integer)

Raises

`Exception` Throw an exception if the given string is not matched

Note: The given string is converted to lowercase and *strip()* is applied before any comparisons

15.2 Variables

Name	Description
__package__	**Value:** `'morpher.misc'`

16 Module morpher.misc.section_reporter

Contains the `SectionReporter` class definition for reporting progress updates

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: November 2, 2011

16.1 Variables

Name	Description
__package__	**Value:** `'morpher.misc'`

16.2 Class SectionReporter

object

morpher.misc.status_reporter.StatusReporter

morpher.misc.section_reporter.SectionReporter

Extends `StatusReporter` with additional functionality for multi-part status bars

`StatusReporter` requires that you know the number of events that will be tracked at the time the status bar is created. However in some cases this information is not completely known. For example, a program might be written to process ten batches of files, but the number of files in the each batch is not known until the previous batch is completed. SectionReporter allows the status bar to be divided into a known number of sections, but the number of events tracked in each section does not need to be known until that section is reached by the status bar. This allows the status bar to display quasi-accurate completion information and remaining time estimates even if the actual information is impossible to determine at that time.

SectionReporter objects can also be reused multiple times by using the `start` method, which essentially resets the counter. The usage pattern is:

```
rep = SectionReporter(2)
rep.start()
rep.startSection(1, 10)
...call rep.pulse() ten times....
```

```
rep.endSection()
rep.startSection(2, 3)
...call rep.pulse() three times
rep.endSection()
```

Warning: The status bar assumes that no other output is sent to the console in dynamic update mode and will not display correctly otherwise

See Also: `StatusReporter` is the base class for this class

16.2.1 Methods

__init__(*self*, *numsections*)

Initializes a new object with the underlying `StatusReporter` object using default settings

Parameters

`numsections`: The total number of sections tracked by the status bar

(type=integer)

Overrides: object.__init__

startSection(*self*, *section*, *numevents*)

Sets the current section to the given section number and sets the total number of events tracked by this section

The variable "curevents" is dynamically scaled at this time as if all previous sections had also tracked the same number of events

Parameters

`section`: The index of the section to start, beginning from 1.

(type=integer)

`numevents`: Total number of events tracked by this section.

(type=integer)

pulse(*self*, *events*=1)

Increments the number of events completed by the given amount, or 1 by default, then reprints the status bar.

Parameters

`events`: The number of events to increment the counter by, default is 1

(type=integer)

Overrides: morpher.misc.status_reporter.StatusReporter.pulse

Note: The status bar will not actually reflect this section as being 100 percent complete until `endSection` is called.

endSection(*self*)

Ends the current section, correcting the status bar to reflect exactly *(cursection/numsections)*100* percent completion

Inherited from morpher.misc.status_reporter.StatusReporter(Section 17.2)

correct(), done(), printBar(), start()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

16.2.2 Properties

Name	Description
Inherited from object	
__class__	

16.2.3 Instance Variables

Name	Description
curevents	The total number of events that have completed, across all sections - dynamically scaled when a new section is entered as if all previous sections were composed of the same number of total events
cursection	The current section index, starting from 1
curtotal	The total number of events tracked by the current section

continued on next page

Name	Description
numsections	The total number of sections making up the status bar
Inherited from morpher.misc.status_reporter.StatusReporter (Section 17.2) current, dynamic, estimate, events, maxlen, size, starttime, sym, total	

17 Module morpher.misc.status_reporter

Contains the `StatusReporter` class definition for reporting progress updates

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: November 1, 2011

17.1 Variables

Name	Description
__package__	**Value:** `'morpher.misc'`

17.2 Class StatusReporter

object
└
morpher.misc.status_reporter.StatusReporter

Known Subclasses: morpher.misc.section_reporter.SectionReporter

Used for displaying a status bar and dynamically updating it on the command line

Tracks program progress by keeping an internal counter that can be incremented by the user, and displays an equivalent status bar and estimated completion time on the command line. When an instance is created the user specifies how many "events" must be completed before the program is considered to have finished. The user can then update the number of "events completed" frequently as the program runs, and the status bar will be dynamically updated on the command line along with an estimated completion time, calculated based on the number of events completed so far, the elapsed time, and the number of events left to complete.

StatusReporter objects can also be reused multiple times by using the `start` method, which essentially resets the counter.

Warning: The status bar assumes that no other output is sent to the console in dynamic update mode and will not display correctly otherwise

See Also: `SectionReporter` extends this class with additional capability

17.2.1 Methods

__init__(*self*, *total*=`100`, *size*=`20`, *dynamic*=`True`, *estimate*=`True`)

Initializes a new object with the given settings which can be reused multiple times for printing status bars.

Parameters

`total:` The total number of events that need to be completed, default is 100

(type=integer)

`size:` The number of units in the displayed status bar, default is 20

(type=integer)

`dynamic:` Enables dynamic updating of the same displayed status bar instead of reprinting on a new line, default is *True*

(type=boolean)

`estimate:` Enables displaying the estimated time remaining, default is *True*

(type=boolean)

Overrides: object.__init__

start(*self*, *msg*=`' Status:'`)

Resets the internal counters, prints a message, and prints the empty status bar.

Parameters

`msg:` The message to print just above the status bar, default is "Status:"

(type=string)

Note: The elapsed time is calculated from the last time this method was called for this object

pulse(*self*, *events*=`1`)

Increments the number of events completed by the given amount, or 1 by default, then reprints the status bar.

Parameters

`events:` The amount to increment the completed events by, default is 1

(type=integer)

correct(*self*, *events*)

Sets the number of completed events to the given number

Parameters

`events`: The number to set the completed event counter to

(type=integer)

done(*self*)

Triggers immediate completion for this status bar, just as if all the events had completed normally

printBar(*self*)

Formats and prints the status bar to stdout.

When displayed the status bar should appear similar to:

```
Status:
\ [====          ]  25% Estimated  1 min 30 sec remaining.....
```

If dynamic updating is set, the last line is erased and rewritten each time the bar is updated; otherwise it is reprinted on the next line.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

17.2.2 Properties

Name	Description
Inherited from object	
__class__	

17.2.3 Instance Variables

Name	Description
current	The number of units to display in the status bar
dynamic	Boolean determining if the status bar should be erased and reprinted on the console instead of printed on a new line for each update

continued on next page

Name	Description
estimate	Boolean determining if the estimated time remaining should be displayed along with the status bar
events	The total number of events that have occurred
maxlen	The maximum length that the status bar can print on a line
size	The number of units in the displayed status bar
starttime	The time that the `start` method was called
sym	The currently displayed "spinner" symbol
total	The total number of events the statusbar is tracking

18 Module morpher.morpher

Contains the `Morpher` class for intelligently fuzzing Application Programming Interface (API) calls to third-party DLLs.

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 21, 2011

18.1 Variables

Name	Description
__package__	**Value:** `'morpher'`

18.2 Class Morpher

object
└ **morpher.morpher.Morpher**

The top-level object for the Morpher tool.

Morpher is executed in three consecutive, mostly seperate phases: parsing, collecting, and fuzzing. The functionality of these three phases is performed by seperate classes (the `Parser`, `Collector`, and `Fuzzer` classes respectively). The `Morpher` class doesn't actually perform a lot of functionality - it is mainly responsible for coordinating these three phases and providing a top-level object for the whole process.

To Do: Add an option to disable printing messages/status bar

18.2.1 Methods

__init__(*self*, ***params*)

Sets up a `Config` object for this morpher and initializes the `logging` system. Almost all other objects that make up Morpher will share a reference to this same `Config` object.

Parameters

`params`: dictionary

(type=Parameters to override the configuration with)

Overrides: object.__init__

run(*self*)

Runs the Morpher program. The Morpher tool is implemented by instantiating a `Parser` object, a `Collector` object, and a `Fuzzer` object and running their respective main functions in order. This function is mainly just responsible for triggering the three main phases of the tool in order and displaying appropriate output to the console.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

18.2.2 Properties

Name	Description
Inherited from object	
__class__	

18.2.3 Instance Variables

Name	Description
cfg	The `Config` object
log	The `logging` object

19 Package morpher.parser

Package documentation

(GRAPH)

19.1 Modules

- **dllexp**: Created on Oct 25, 2011
  (Section 20, p. 69)
- **parser**: Created on Oct 21, 2011
  (Section 21, p. 70)

20 Module morpher.parser.dllexp

Created on Oct 25, 2011

Author: Rob

20.1 Variables

Name	Description
__package__	**Value:** `'morpher.parser'`

20.2 Class DllExp

object
└
morpher.parser.dllexp.DllExp

Acts as a front-end to call the DLL Explorer (dllexp.exe) tool

20.2.1 Methods

__init__(*self*, *cfg*)

init documentation

Overrides: object.__init__

getFunctions(*self*)

Runs the dllexp.exe tool to pull the export table from the target DLL. Returns a list of (funcname, ordinal, rel_addr) tuples - (string, int, int)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

20.2.2 Properties

Name	Description
Inherited from object	
__class__	

21 Module morpher.parser.parser

Created on Oct 21, 2011

Author: Rob

21.1 Variables

Name	Description
CPPPATH	**Value:** `'../../tools/tcc/tcc.exe'`
__package__	**Value:** `'morpher.parser'`

21.2 Class Parser

object
└ **morpher.parser.parser.Parser**

Parser documentation

21.2.1 Methods

__init__(*self*, *cfg*)

init documentation

Overrides: object.__init__

parse_file(*self*)

```
Parse a C file using pycparser.

filename:
    Name of the file you want to parse.

use_cpp:
    Set to True if you want to execute the C pre-processor
    on the file prior to parsing it.

cpp_path:
    If use_cpp is True, this is the path to 'cpp' on your
    system. If no path is provided, it attempts to just
    execute 'cpp', so it must be in your PATH.

cpp_args:
    If use_cpp is True, set this to the command line
    arguments strings to cpp. Be careful with quotes -
    it's best to pass a raw string (r'') here.
    For example:
    r'-I../utils/fake_libc_include'
    If several arguments are required, pass a list of
    strings.

When successful, an AST is returned. ParseError can be
thrown if the file doesn't parse successfully.

Errors from cpp will be printed out.
```

parseXML(*self*, *ast*, *element*, *name*, *state*, *printflag*)

Fill in data!!

parse(*self*)

Analyzes the target DLL and header file to retrieve function prototypes. Outputs a XML file containing a model of the exported prototypes

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

21.2.2 Properties

Name	Description
Inherited from object __class__	

22 Package morpher.ply

22.1 Modules

23 Module morpher.ply.cpp

23.1 Functions

t_CPP_WS(t)

\s+

CPP_INTEGER(t)

((((((0x)|(0X))[0-9a-fA-F]+)|(\d+))([uU]|[lL]|[uU][lL]|[lL][uU])?)

t_CPP_INTEGER(t)

((((((0x)|(0X))[0-9a-fA-F]+)|(\d+))([uU]|[lL]|[uU][lL]|[lL][uU])?)

t_CPP_STRING(t)

\"([^\\\n]|(\\(.|\n)))*?\"

t_CPP_CHAR(t)

(L)?\'([^\\\n]|(\\(.|\n)))*?\'

t_CPP_COMMENT(t)

(/*(.|\n)*?*/)|(//.*?\n)

t_error(t)

trigraph(*input*)

23.2 Variables

Name	Description
tokens	**Value:** `('CPP_ID', 'CPP_INTEGER', 'CPP_FLOAT', 'CPP_STRING', 'CPP...`
literals	**Value:** `'+-*/%|&~^<>=!?()[]{}.,;:\\\'"'`
t_CPP_POUND	**Value:** `'\\#'`
t_CPP_DPOUND	**Value:** `'\\#\\#'`
t_CPP_ID	**Value:** `'[A-Za-z_][\\w_]*'`
t_CPP_FLOAT	**Value:** `'((\\d+)(\\.\\d+)(e(\\+|-)?(\\d+))? | (\\d+)e(\\+|-)?(\\d...`
__package__	**Value:** `'morpher.ply'`

23.3 Class Macro

object
└ **morpher.ply.cpp.Macro**

23.3.1 Methods

__init__(*self*, *name*, *value*, *arglist*=`None`, *variadic*=`False`)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

23.3.2 Properties

Name	Description
Inherited from object	
__class__	

23.4 Class Preprocessor

object
└
morpher.ply.cpp.Preprocessor

23.4.1 Methods

__init__(*self*, *lexer*=None)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

tokenize(*self*, *text*)

error(*self*, *file*, *line*, *msg*)

lexprobe(*self*)

add_path(*self*, *path*)

group_lines(*self*, *input*)

tokenstrip(*self*, *tokens*)

collect_args(*self*, *tokenlist*)

macro_prescan(*self*, *macro*)

macro_expand_args(*self*, *macro*, *args*)

expand_macros(*self*, *tokens*, *expanded*=None)

evalexpr(*self*, *tokens*)

parsegen(*self*, *input*, *source*=None)

include(*self*, *tokens*)

define(*self*, *tokens*)

undef(*self*, *tokens*)

parse(*self*, *input*, *source*=`None`, *ignore*={})

token(*self*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

23.4.2 Properties

Name	Description
Inherited from object	
__class__	

24 Module morpher.ply.ctokens

24.1 Functions

t_COMMENT(t)

/*(.|\n)*?*/

t_CPPCOMMENT(t)

//.*\n

24.2 Variables

Name	Description
tokens	**Value:** ['ID', 'TYPEID', 'ICONST', 'FCONST', 'SCONST', 'CCONST', ...
t_PLUS	**Value:** '\\+'
t_MINUS	**Value:** '-'
t_TIMES	**Value:** '*'
t_DIVIDE	**Value:** '/'
t_MODULO	**Value:** '%'
t_OR	**Value:** '\\|'
t_AND	**Value:** '&'
t_NOT	**Value:** '~'
t_XOR	**Value:** '\\^'
t_LSHIFT	**Value:** '<<'
t_RSHIFT	**Value:** '>>'
t_LOR	**Value:** '\\|\\|'
t_LAND	**Value:** '&&'
t_LNOT	**Value:** '!'
t_LT	**Value:** '<'
t_GT	**Value:** '>'
t_LE	**Value:** '<='
t_GE	**Value:** '>='
t_EQ	**Value:** '=='
t_NE	**Value:** '!='
t_EQUALS	**Value:** '='
t_TIMESEQUAL	**Value:** '*='
t_DIVEQUAL	**Value:** '/='
t_MODEQUAL	**Value:** '%='
t_PLUSEQUAL	**Value:** '\\+='
t_MINUSEQUAL	**Value:** '-='

continued on next page

Name	Description
t_LSHIFTEQUAL	**Value:** '<<='
t_RSHIFTEQUAL	**Value:** '>>='
t_ANDEQUAL	**Value:** '&='
t_OREQUAL	**Value:** '\\|='
t_XOREQUAL	**Value:** '^='
t_INCREMENT	**Value:** '\\+\\+'
t_DECREMENT	**Value:** '--'
t_ARROW	**Value:** '->'
t_TERNARY	**Value:** '\\?'
t_LPAREN	**Value:** '\\('
t_RPAREN	**Value:** '\\)'
t_LBRACKET	**Value:** '\\['
t_RBRACKET	**Value:** '\\]'
t_LBRACE	**Value:** '\\{'
t_RBRACE	**Value:** '\\}'
t_COMMA	**Value:** ','
t_PERIOD	**Value:** '\\.'
t_SEMI	**Value:** ';'
t_COLON	**Value:** ':'
t_ELLIPSIS	**Value:** '\\.\\.\\.'
t_ID	**Value:** '[A-Za-z_][A-Za-z0-9_]*'
t_INTEGER	**Value:** '\\d+([uU]|[lL]|[uU][lL]|[lL][uU])?'
t_FLOAT	**Value:** '((\\d+)(\\.\\d+)(e(\\+|-)?(\\d+))? | (\\d+)e(\\+|-)?(\\d...
t_STRING	**Value:** '\\"([^\\\\\\n]|(\\\\.))*?\\"'
t_CHARACTER	**Value:** '(L)?\\\'([^\\\\\\n]|(\\\\.))*?\\\''
__package__	**Value:** None

25 Module morpher.ply.lex

Version: 3.4

25.1 Functions

func_code(*f*)

get_caller_module_dict(*levels*)

lex(*module*=None, *object*=None, *debug*=0, *optimize*=0, *lextab*='lextab', *reflags*=0, *nowarn*=0, *outputdir*='', *debuglog*=None, *errorlog*=None)

runmain(*lexer*=None, *data*=None)

TOKEN(*r*)

Token(*r*)

25.2 Variables

Name	Description
__tabversion__	**Value:** '3.2'
StringTypes	**Value:** (<type 'str'>, <type 'unicode'>)
__package__	**Value:** 'morpher.ply'

25.3 Class LexError

object

exceptions.BaseException

exceptions.Exception

morpher.ply.lex.LexError

25.3.1 Methods

__init__(*self*, *message*, *s*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

25.3.2 Properties

Name	Description
Inherited from exceptions.BaseException	
args, message	
Inherited from object	
__class__	

25.4 Class LexToken

object
└
morpher.ply.lex.LexToken

25.4.1 Methods

__str__(*self*)

str(x)

Overrides: object.__str__ extit(inherited documentation)

__repr__(*self*)

repr(x)

Overrides: object.__repr__ extit(inherited documentation)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

25.4.2 Properties

Name	Description
Inherited from object	
__class__	

25.5 Class PlyLogger

object
└
morpher.ply.lex.PlyLogger

25.5.1 Methods

__init__(*self*, *f*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

critical(*self*, *msg*, **args*, ***kwargs*)

warning(*self*, *msg*, **args*, ***kwargs*)

error(*self*, *msg*, **args*, ***kwargs*)

info(*self*, *msg*, **args*, ***kwargs*)

debug(*self*, *msg*, **args*, ***kwargs*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

25.5.2 Properties

Name	Description
Inherited from object	
__class__	

25.6 Class NullLogger

object
└
morpher.ply.lex.NullLogger

25.6.1 Methods

__getattribute__(*self*, *name*)

x.__getattribute__('name') <==> x.name

Overrides: object.__getattribute__ extit(inherited documentation)

__call__(*self*, **args*, ***kwargs*)

Inherited from object

__delattr__(), __format__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

25.6.2 Properties

Name	Description
Inherited from object	
__class__	

25.7 Class Lexer

25.7.1 Methods

__init__(*self*)

clone(*self*, *object*=None)

writetab(*self*, *tabfile*, *outputdir*='')

readtab(*self*, *tabfile*, *fdict*)

input(*self*, *s*)

begin(*self*, *state*)

push_state(*self*, *state*)

pop_state(*self*)

current_state(*self*)

skip(*self*, *n*)

token(*self*)

__iter__(*self*)

next(*self*)

__next__(*self*)

25.8 Class LexerReflect

object
└ **morpher.ply.lex.LexerReflect**

25.8.1 Methods

__init__(*self*, *ldict*, *log*=`None`, *reflags*=`0`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

get_all(*self*)

validate_all(*self*)

get_tokens(*self*)

validate_tokens(*self*)

get_literals(*self*)

validate_literals(*self*)

get_states(*self*)

get_rules(*self*)

validate_rules(*self*)

validate_file(*self*, *filename*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

25.8.2 Properties

Name	Description
Inherited from object	
__class__	

26 Module morpher.ply.yacc

Version: 3.4

26.1 Functions

func_code(*f*)

load_ply_lex()

format_result(*r*)

format_stack_entry(*r*)

rightmost_terminal(*symbols*, *terminals*)

digraph(*X*, *R*, *FP*)

traverse(*x*, *N*, *stack*, *F*, *X*, *R*, *FP*)

get_caller_module_dict(*levels*)

parse_grammar(*doc*, *file*, *line*)

yacc(*method*='LALR', *debug*=1, *module*=None, *tabmodule*='parsetab', *start*=None, *check_recursion*=1, *optimize*=0, *write_tables*=1, *debugfile*='parser.out', *outputdir*='', *debuglog*=None, *errorlog*=None, *picklefile*=None)

26.2 Variables

Name	Description
__tabversion__	**Value:** '3.2'
yaccdebug	**Value:** 1
debug_file	**Value:** 'parser.out'
tab_module	**Value:** 'parsetab'
default_lr	**Value:** 'LALR'
error_count	**Value:** 3
yaccdevel	**Value:** 0

continued on next page

Name	Description
resultlimit	**Value:** `40`
pickle_protocol	**Value:** `0`
MAXINT	**Value:** `2147483647`
__package__	**Value:** `'morpher.ply'`

26.3 Class PlyLogger

object
└
morpher.ply.yacc.PlyLogger

26.3.1 Methods

__init__(*self*, *f*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

debug(*self*, *msg*, **args*, ***kwargs*)

info(*self*, *msg*, **args*, ***kwargs*)

warning(*self*, *msg*, **args*, ***kwargs*)

error(*self*, *msg*, **args*, ***kwargs*)

critical(*self*, *msg*, **args*, ***kwargs*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

26.3.2 Properties

Name	Description
Inherited from object	
__class__	

26.4 Class NullLogger

object
└ **morpher.ply.yacc.NullLogger**

26.4.1 Methods

__getattribute__(*self*, *name*)

x.__getattribute__('name') <==> x.name

Overrides: object.__getattribute__ extit(inherited documentation)

__call__(*self*, **args*, ***kwargs*)

Inherited from object

__delattr__(), __format__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

26.4.2 Properties

Name	Description
Inherited from object	
__class__	

26.5 Class YaccError

object
└ exceptions.BaseException
└ exceptions.Exception
└ **morpher.ply.yacc.YaccError**

Known Subclasses: morpher.ply.yacc.GrammarError, morpher.ply.yacc.LALRError, morpher.ply.yacc.VersionError

26.5.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

26.5.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

26.6 Class YaccSymbol

26.6.1 Methods

__str__(*self*)

__repr__(*self*)

26.7 Class YaccProduction

26.7.1 Methods

__init__(*self*, *s*, *stack*=`None`)

__getitem__(*self*, *n*)

__setitem__(*self*, *n*, *v*)

__getslice__(*self*, *i*, *j*)

__len__(*self*)

lineno(*self*, *n*)

set_lineno(*self*, *n*, *lineno*)

linespan(*self*, *n*)

lexpos(*self*, *n*)

lexspan(*self*, *n*)

error(*self*)

26.8 Class LRParser

26.8.1 Methods

__init__(*self*, *lrtab*, *errorf*)

errok(*self*)

restart(*self*)

parse(*self*, *input*=`None`, *lexer*=`None`, *debug*=`0`, *tracking*=`0`, *tokenfunc*=`None`)

parsedebug(*self*, *input*=`None`, *lexer*=`None`, *debug*=`None`, *tracking*=`0`, *tokenfunc*=`None`)

parseopt(*self*, *input*=`None`, *lexer*=`None`, *debug*=`0`, *tracking*=`0`, *tokenfunc*=`None`)

parseopt_notrack(*self*, *input*=`None`, *lexer*=`None`, *debug*=`0`, *tracking*=`0`, *tokenfunc*=`None`)

26.9 Class Production

object
└ **morpher.ply.yacc.Production**

26.9.1 Methods

__init__(*self*, *number*, *name*, *prod*, *precedence*=`('right', 0)`, *func*=`None`, *file*=`''`, *line*=`0`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

str(x)

Overrides: object.__str__ extit(inherited documentation)

__repr__(*self*)

repr(x)

Overrides: object.__repr__ extit(inherited documentation)

__len__(*self*)

__nonzero__(*self*)

__getitem__(*self*, *index*)

lr_item(*self*, *n*)

bind(*self*, *pdict*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

26.9.2 Properties

Name	Description
Inherited from object	
__class__	

26.9.3 Class Variables

Name	Description
reduced	**Value:** `0`

26.10 Class MiniProduction

object
└
morpher.ply.yacc.MiniProduction

26.10.1 Methods

__init__(*self*, *str*, *name*, *len*, *func*, *file*, *line*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

str(x)

Overrides: object.__str__ extit(inherited documentation)

__repr__(*self*)

repr(x)

Overrides: object.__repr__ extit(inherited documentation)

bind(*self*, *pdict*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

26.10.2 Properties

Name	Description
Inherited from object	
__class__	

26.11 Class LRItem

object
└
morpher.ply.yacc.LRItem

26.11.1 Methods

__init__(*self*, *p*, *n*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

str(x)

Overrides: object.__str__ extit(inherited documentation)

__repr__(*self*)

repr(x)

Overrides: object.__repr__ extit(inherited documentation)

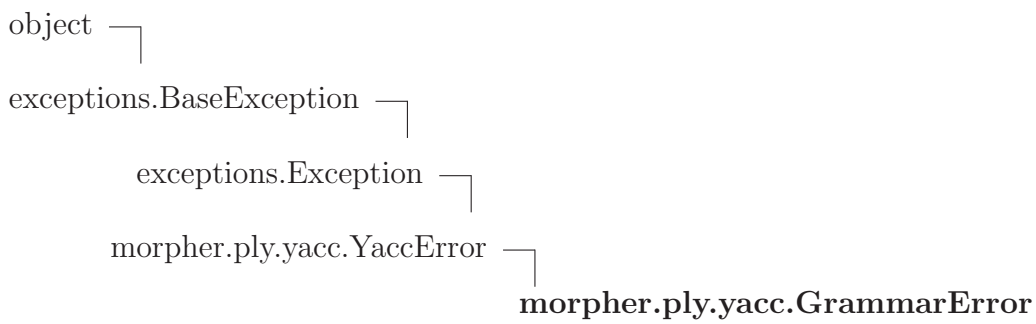
Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

26.11.2 Properties

Name	Description
Inherited from object	
__class__	

26.12 Class GrammarError



26.12.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

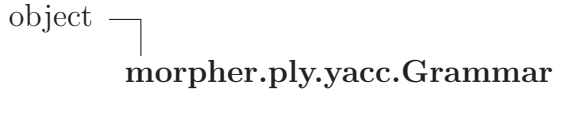
Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

26.12.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

26.13 Class Grammar



26.13.1 Methods

__init__(*self*, *terminals*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

__len__(*self*)

__getitem__(*self*, *index*)

set_precedence(*self*, *term*, *assoc*, *level*)

add_production(*self*, *prodname*, *syms*, *func*=`None`, *file*=`''`, *line*=`0`)

set_start(*self*, *start*=`None`)

find_unreachable(*self*)

infinite_cycles(*self*)

undefined_symbols(*self*)

unused_terminals(*self*)

unused_rules(*self*)

unused_precedence(*self*)

compute_first(*self*)

compute_follow(*self*, *start*=`None`)

build_lritems(*self*)

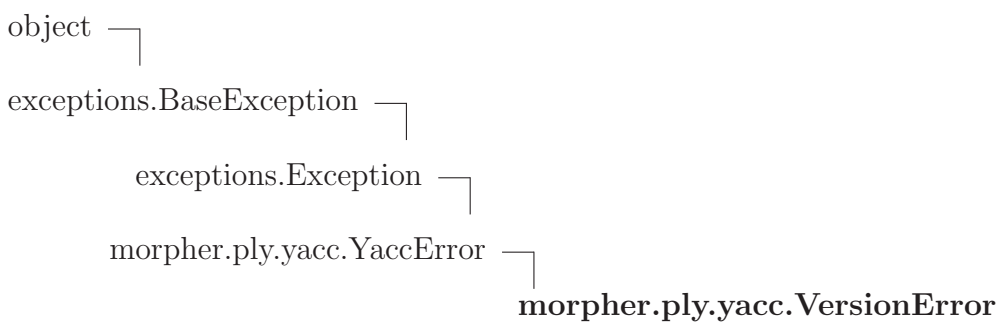
Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

26.13.2 Properties

Name	Description
Inherited from object	
__class__	

26.14 Class VersionError



26.14.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

26.14.2 Properties

Name	Description
Inherited from exceptions.BaseException	
args, message	
Inherited from object	
__class__	

26.15 Class LRTable

object
└ **morpher.ply.yacc.LRTable**

Known Subclasses: morpher.ply.yacc.LRGeneratedTable

26.15.1 Methods

__init__(*self*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

read_table(*self*, *module*)

read_pickle(*self*, *filename*)

bind_callables(*self*, *pdict*)

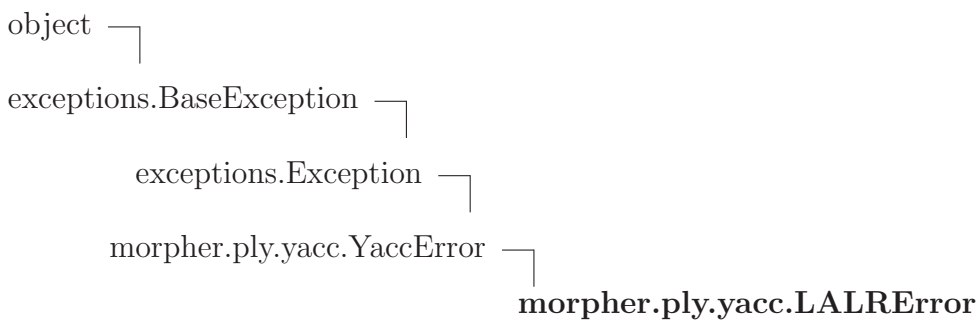
Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

26.15.2 Properties

Name	Description
Inherited from object	
__class__	

26.16 Class LALRError



26.16.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

26.16.2 Properties

Name	Description
Inherited from exceptions.BaseException	
args, message	
Inherited from object	
__class__	

26.17 Class LRGeneratedTable

26.17.1 Methods

__init__(*self*, *grammar*, *method*='LALR', *log*=None)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

lr0_closure(*self*, *I*)

lr0_goto(*self*, *I*, *x*)

lr0_items(*self*)

compute_nullable_nonterminals(*self*)

find_nonterminal_transitions(*self*, *C*)

dr_relation(*self*, *C*, *trans*, *nullable*)

reads_relation(*self*, *C*, *trans*, *empty*)

compute_lookback_includes(*self*, *C*, *trans*, *nullable*)

compute_read_sets(*self*, *C*, *ntrans*, *nullable*)

compute_follow_sets(*self*, *ntrans*, *readsets*, *inclsets*)

add_lookaheads(*self*, *lookbacks*, *followset*)

add_lalr_lookaheads(*self*, *C*)

lr_parse_table(*self*)

write_table(*self*, *modulename*, *outputdir*='', *signature*='')

pickle_table(*self*, *filename*, *signature*='')

Inherited from morpher.ply.yacc.LRTable(Section 26.15)

bind_callables(), read_pickle(), read_table()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

26.17.2 Properties

Name	Description
Inherited from object	
__class__	

26.18 Class ParserReflect

object
└
morpher.ply.yacc.ParserReflect

26.18.1 Methods

__init__(*self*, *pdict*, *log*=None)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

get_all(*self*)

validate_all(*self*)

signature(*self*)

validate_files(*self*)

get_start(*self*)

validate_start(*self*)

get_error_func(*self*)

validate_error_func(*self*)

get_tokens(*self*)

validate_tokens(*self*)

get_precedence(*self*)

validate_precedence(*self*)

get_pfunctions(*self*)

validate_pfunctions(*self*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

26.18.2 Properties

Name	Description
Inherited from object	
__class__	

27 Package morpher.pycparser

Version: 2.05

27.1 Modules

28 Module morpher.pycparser._ast_gen

28.1 Variables

Name	Description
__package__	**Value:** `'morpher.pycparser'`

28.2 Class ASTCodeGenerator

object
└
morpher.pycparser._ast_gen.ASTCodeGenerator

28.2.1 Methods

__init__(*self*, *cfg_filename*=`'_c_ast.cfg'`)

Initialize the code generator from a configuration file.

Overrides: object.__init__

generate(*self*, *file*=`None`)

Generates the code into file, an open file buffer.

parse_cfgfile(*self*, *filename*)

Parse the configuration file and yield pairs of (name, contents) for each node.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

28.2.2 Properties

Name	Description
Inherited from object	
__class__	

28.3 Class NodeCfg

object
└
morpher.pycparser._ast_gen.NodeCfg

Node configuration.

name: node name contents: a list of contents - attributes and child nodes See comment at the top of the configuration file for details.

28.3.1 Methods

__init__(*self*, *name*, *contents*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

generate_source(*self*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

28.3.2 Properties

Name	Description
Inherited from object	
__class__	

29 Module morpher.pycparser._build_tables

29.1 Variables

Name	Description
ast_gen	**Value:** `ASTCodeGenerator('_c_ast.cfg')`

30 Module morpher.pycparser.c_ast

30.1 Variables

Name	Description
__package__	**Value:** `'morpher.pycparser'`

30.2 Class Node

object
└
morpher.pycparser.c_ast.Node

Known Subclasses: morpher.pycparser.c_ast.ArrayDecl, morpher.pycparser.c_ast.ArrayRef, morpher.pycparser.c_ast.Assignment, morpher.pycparser.c_ast.BinaryOp, morpher.pycparser.c_ast.Break, morpher.pycparser.c_ast.Case, morpher.pycparser.c_ast.Cast, morpher.pycparser.c_ast.Compound, morpher.pycparser.c_ast.CompoundLiteral, morpher.pycparser.c_ast.Constant, morpher.pycparser.c_ast.Co morpher.pycparser.c_ast.Decl, morpher.pycparser.c_ast.DeclList, morpher.pycparser.c_ast.Default, morpher.pycparser.c_ast.DoWhile, morpher.pycparser.c_ast.EllipsisParam, morpher.pycparser.c_ast.Empty morpher.pycparser.c_ast.Enum, morpher.pycparser.c_ast.Enumerator, morpher.pycparser.c_ast.Enumerato morpher.pycparser.c_ast.ExprList, morpher.pycparser.c_ast.FileAST, morpher.pycparser.c_ast.For, morpher.pycparser.c_ast.FuncCall, morpher.pycparser.c_ast.FuncDecl, morpher.pycparser.c_ast.FuncDef, morpher.pycparser.c_ast.Goto, morpher.pycparser.c_ast.ID, morpher.pycparser.c_ast.IdentifierType, morpher.pycparser.c_ast.If, morpher.pycparser.c_ast.Label, morpher.pycparser.c_ast.NamedInitializer, morpher.pycparser.c_ast.ParamList, morpher.pycparser.c_ast.PtrDecl, morpher.pycparser.c_ast.Return, morpher.pycparser.c_ast.Struct, morpher.pycparser.c_ast.StructRef, morpher.pycparser.c_ast.Switch, morpher.pycparser.c_ast.TernaryOp, morpher.pycparser.c_ast.TypeDecl, morpher.pycparser.c_ast.Typedef, morpher.pycparser.c_ast.Typename, morpher.pycparser.c_ast.UnaryOp, morpher.pycparser.c_ast.Union, morpher.pycparser.c_ast.While

Abstract base class for AST nodes.

30.2.1 Methods

children(*self*)

A sequence of all children that are Nodes

show(*self*, *buf*=sys.stdout, *offset*=0, *attrnames*=False, *showcoord*=False)

```
Pretty print the Node and all its attributes and
children (recursively) to a buffer.

file:
    Open IO buffer into which the Node is printed.

offset:
    Initial offset (amount of leading spaces)

attrnames:
    True if you want to see the attribute names in
    name=value pairs. False to only see the values.

showcoord:
    Do you want the coordinates of each Node to be
    displayed.
```

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.2.2 Properties

Name	Description
Inherited from object	
__class__	

30.3 Class NodeVisitor

object
└ **morpher.pycparser.c_ast.NodeVisitor**

```
A base NodeVisitor class for visiting c_ast nodes.
Subclass it and define your own visit_XXX methods, where
XXX is the class name you want to visit with these
methods.

For example:
```

```
class ConstantVisitor(NodeVisitor):
    def __init__(self):
        self.values = []

    def visit_Constant(self, node):
        self.values.append(node.value)

Creates a list of values of all the constant nodes
encountered below the given node. To use it:

cv = ConstantVisitor()
cv.visit(node)

Notes:

*   generic_visit() will be called for AST nodes for which
    no visit_XXX method was defined.
*   The children of nodes for which a visit_XXX was
    defined will not be visited - if you need this, call
    generic_visit() on the node.
    You can use:
        NodeVisitor.generic_visit(self, node)
*   Modeled after Python's own AST visiting facilities
    (the ast module of Python 3.0)
```

30.3.1 Methods

visit(*self*, *node*)

Visit a node.

generic_visit(*self*, *node*)

Called if no explicit visitor function exists for a node. Implements preorder visiting of the node.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.3.2 Properties

Name	Description
Inherited from object	
__class__	

30.4 Class ArrayDecl

object
morpher.pycparser.c_ast.Node
morpher.pycparser.c_ast.ArrayDecl

30.4.1 Methods

__init__(*self*, *type*, *dim*, *coord*=None)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.4.2 Properties

Name	Description
Inherited from object	
__class__	

30.4.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.5 Class ArrayRef

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.ArrayRef

30.5.1 Methods

__init__(*self*, *name*, *subscript*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.5.2 Properties

Name	Description
Inherited from object	
__class__	

30.5.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.6 Class Assignment

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Assignment

30.6.1 Methods

__init__(*self*, *op*, *lvalue*, *rvalue*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.6.2 Properties

Name	Description
Inherited from object	
__class__	

30.6.3 Class Variables

Name	Description
attr_names	**Value:** `('op')`

30.7 Class BinaryOp

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.BinaryOp

30.7.1 Methods

__init__(*self*, *op*, *left*, *right*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.7.2 Properties

Name	Description
Inherited from object	
__class__	

30.7.3 Class Variables

Name	Description
attr_names	**Value:** `('op')`

30.8 Class Break

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Break

30.8.1 Methods

__init__(*self*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

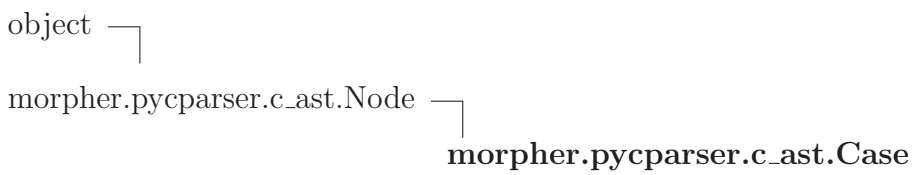
30.8.2 Properties

Name	Description
Inherited from object	
__class__	

30.8.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.9 Class Case



30.9.1 Methods

__init__(*self*, *expr*, *stmt*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

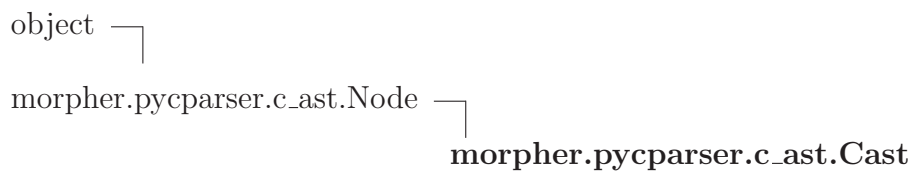
30.9.2 Properties

Name	Description
Inherited from object	
__class__	

30.9.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.10 Class Cast



30.10.1 Methods

__init__(*self*, *to_type*, *expr*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.10.2 Properties

Name	Description
Inherited from object	
__class__	

30.10.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.11 Class Compound

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Compound

30.11.1 Methods

__init__(*self*, *block_items*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.11.2 Properties

Name	Description
Inherited from object	
__class__	

30.11.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.12 Class CompoundLiteral

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.CompoundLiteral

30.12.1 Methods

__init__(*self*, *type*, *init*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

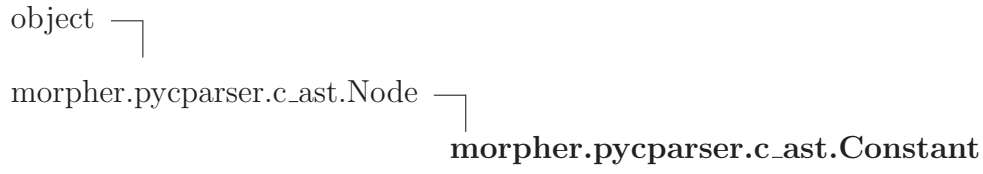
30.12.2 Properties

Name	Description
Inherited from object	
__class__	

30.12.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.13 Class Constant



30.13.1 Methods

__init__(*self*, *type*, *value*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.13.2 Properties

Name	Description
Inherited from object	
__class__	

30.13.3 Class Variables

Name	Description
attr_names	**Value:** `('type', 'value')`

30.14 Class Continue

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Continue

30.14.1 Methods

__init__(*self*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.14.2 Properties

Name	Description
Inherited from object	
__class__	

30.14.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.15 Class Decl

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Decl

30.15.1 Methods

__init__(*self*, *name*, *quals*, *storage*, *funcspec*, *type*, *init*, *bitsize*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.15.2 Properties

Name	Description
Inherited from object	
__class__	

30.15.3 Class Variables

Name	Description
attr_names	**Value:** `('name', 'quals', 'storage', 'funcspec')`

30.16 Class DeclList

object
morpher.pycparser.c_ast.Node
morpher.pycparser.c_ast.DeclList

30.16.1 Methods

__init__(*self*, *decls*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.16.2 Properties

Name	Description
Inherited from object	
__class__	

30.16.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.17 Class Default

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Default

30.17.1 Methods

__init__(*self*, *stmt*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.17.2 Properties

Name	Description
Inherited from object	
__class__	

30.17.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.18 Class DoWhile

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.DoWhile

30.18.1 Methods

__init__(*self*, *cond*, *stmt*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.18.2 Properties

Name	Description
Inherited from object	
__class__	

30.18.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.19 Class EllipsisParam

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.EllipsisParam

30.19.1 Methods

__init__(*self*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.19.2 Properties

Name	Description
Inherited from object	
__class__	

30.19.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.20 Class EmptyStatement

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.EmptyStatement

30.20.1 Methods

__init__(*self*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.20.2 Properties

Name	Description
Inherited from object	
__class__	

30.20.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.21 Class Enum

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Enum

30.21.1 Methods

__init__(*self*, *name*, *values*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.21.2 Properties

Name	Description
Inherited from object	
__class__	

30.21.3 Class Variables

Name	Description
attr_names	**Value:** `('name')`

30.22 Class Enumerator

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Enumerator

30.22.1 Methods

__init__(*self*, *name*, *value*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.22.2 Properties

Name	Description
Inherited from object	
__class__	

30.22.3 Class Variables

Name	Description
attr_names	**Value:** `('name')`

30.23 Class EnumeratorList

object
morpher.pycparser.c_ast.Node
morpher.pycparser.c_ast.EnumeratorList

30.23.1 Methods

__init__(*self*, *enumerators*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.23.2 Properties

Name	Description
Inherited from object	
__class__	

30.23.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.24 Class ExprList

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.ExprList

30.24.1 Methods

__init__(*self*, *exprs*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.24.2 Properties

Name	Description
Inherited from object	
__class__	

30.24.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.25 Class FileAST

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.FileAST

30.25.1 Methods

__init__(*self*, *ext*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.25.2 Properties

Name	Description
Inherited from object	
__class__	

30.25.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.26 Class For

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.For

30.26.1 Methods

__init__(*self*, *init*, *cond*, *next*, *stmt*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.26.2 Properties

Name	Description
Inherited from object	
__class__	

30.26.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.27 Class FuncCall

object
morpher.pycparser.c_ast.Node
morpher.pycparser.c_ast.FuncCall

30.27.1 Methods

__init__(*self*, *name*, *args*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.27.2 Properties

Name	Description
Inherited from object	
__class__	

30.27.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.28 Class FuncDecl

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.FuncDecl

30.28.1 Methods

__init__(*self*, *args*, *type*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.28.2 Properties

Name	Description
Inherited from object	
__class__	

30.28.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.29 Class FuncDef

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.FuncDef

30.29.1 Methods

__init__(*self*, *decl*, *param_decls*, *body*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.29.2 Properties

Name	Description
Inherited from object	
__class__	

30.29.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.30 Class Goto

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Goto

30.30.1 Methods

__init__(*self*, *name*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.30.2 Properties

Name	Description
Inherited from object	
__class__	

30.30.3 Class Variables

Name	Description
attr_names	**Value:** `('name')`

30.31 Class ID

object

morpher.pycparser.c_ast.Node

morpher.pycparser.c_ast.ID

30.31.1 Methods

__init__(*self*, *name*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.31.2 Properties

Name	Description
Inherited from object	
__class__	

30.31.3 Class Variables

Name	Description
attr_names	**Value:** `('name')`

30.32 Class IdentifierType

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.IdentifierType

30.32.1 Methods

__init__(*self*, *names*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.32.2 Properties

Name	Description
Inherited from object	
__class__	

30.32.3 Class Variables

Name	Description
attr_names	**Value:** `('names')`

30.33 Class If

object
morpher.pycparser.c_ast.Node
morpher.pycparser.c_ast.If

30.33.1 Methods

__init__(*self*, *cond*, *iftrue*, *iffalse*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.33.2 Properties

Name	Description
Inherited from object	
__class__	

30.33.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.34 Class Label

object

morpher.pycparser.c_ast.Node

morpher.pycparser.c_ast.Label

30.34.1 Methods

__init__(*self*, *name*, *stmt*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.34.2 Properties

Name	Description
Inherited from object	
__class__	

30.34.3 Class Variables

Name	Description
attr_names	**Value:** `('name')`

30.35 Class NamedInitializer

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.NamedInitializer

30.35.1 Methods

__init__(*self*, *name*, *expr*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.35.2 Properties

Name	Description
Inherited from object	
__class__	

30.35.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.36 Class ParamList

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.ParamList

30.36.1 Methods

__init__(*self*, *params*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.36.2 Properties

Name	Description
Inherited from object	
__class__	

30.36.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.37 Class PtrDecl

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.PtrDecl

30.37.1 Methods

__init__(*self*, *quals*, *type*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.37.2 Properties

Name	Description
Inherited from object	
__class__	

30.37.3 Class Variables

Name	Description
attr_names	**Value:** `('quals')`

30.38 Class Return

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Return

30.38.1 Methods

__init__(*self*, *expr*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.38.2 Properties

Name	Description
Inherited from object	
__class__	

30.38.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.39 Class Struct

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Struct

30.39.1 Methods

__init__(*self*, *name*, *decls*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.39.2 Properties

Name	Description
Inherited from object	
__class__	

30.39.3 Class Variables

Name	Description
attr_names	**Value:** `('name')`

30.40 Class StructRef

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.StructRef

30.40.1 Methods

__init__(*self*, *name*, *type*, *field*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.40.2 Properties

Name	Description
Inherited from object	
__class__	

30.40.3 Class Variables

Name	Description
attr_names	**Value:** `('type')`

30.41 Class Switch

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Switch

30.41.1 Methods

__init__(*self*, *cond*, *stmt*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.41.2 Properties

Name	Description
Inherited from object	
__class__	

30.41.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.42 Class TernaryOp

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.TernaryOp

30.42.1 Methods

__init__(*self*, *cond*, *iftrue*, *iffalse*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.42.2 Properties

Name	Description
Inherited from object	
__class__	

30.42.3 Class Variables

Name	Description
attr_names	**Value:** `()`

30.43 Class TypeDecl

object
morpher.pycparser.c_ast.Node
morpher.pycparser.c_ast.TypeDecl

30.43.1 Methods

__init__(*self*, *declname*, *quals*, *type*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.43.2 Properties

Name	Description
Inherited from object	
__class__	

30.43.3 Class Variables

Name	Description
attr_names	**Value:** `('declname', 'quals')`

30.44 Class Typedef

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Typedef

30.44.1 Methods

__init__(*self*, *name*, *quals*, *storage*, *type*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.44.2 Properties

Name	Description
Inherited from object	
__class__	

30.44.3 Class Variables

Name	Description
attr_names	**Value:** `('name', 'quals', 'storage')`

30.45 Class Typename

object
morpher.pycparser.c_ast.Node
morpher.pycparser.c_ast.Typename

30.45.1 Methods

__init__(*self*, *quals*, *type*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.45.2 Properties

Name	Description
Inherited from object	
__class__	

30.45.3 Class Variables

Name	Description
attr_names	**Value:** `('quals')`

30.46 Class UnaryOp

object
morpher.pycparser.c_ast.Node
morpher.pycparser.c_ast.UnaryOp

30.46.1 Methods

__init__(*self*, *op*, *expr*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.46.2 Properties

Name	Description
Inherited from object	
__class__	

30.46.3 Class Variables

Name	Description
attr_names	**Value:** `('op')`

30.47 Class Union

object
└
morpher.pycparser.c_ast.Node
└
morpher.pycparser.c_ast.Union

30.47.1 Methods

__init__(*self*, *name*, *decls*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.47.2 Properties

Name	Description
Inherited from object	
__class__	

30.47.3 Class Variables

Name	Description
attr_names	**Value:** `('name')`

30.48 Class While

object
morpher.pycparser.c_ast.Node
morpher.pycparser.c_ast.While

30.48.1 Methods

__init__(*self*, *cond*, *stmt*, *coord*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

children(*self*)

A sequence of all children that are Nodes

Overrides: morpher.pycparser.c_ast.Node.children extit(inherited documentation)

Inherited from morpher.pycparser.c_ast.Node(Section 30.2)

show()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

30.48.2 Properties

Name	Description
Inherited from object	
__class__	

30.48.3 Class Variables

Name	Description
attr_names	**Value:** `()`

31 Module morpher.pycparser.c_lexer

31.1 Variables

Name	Description
__package__	**Value:** `'morpher.pycparser'`

31.2 Class CLexer

object
└
morpher.pycparser.c_lexer.CLexer

A lexer for the C language. After building it, set the input text with input(), and call token() to get new tokens.

The public attribute filename can be set to an initial filaneme, but the lexer will update it upon #line directives.

31.2.1 Methods

__**init**__(*self*, *error_func*, *type_lookup_func*)

```
Create a new Lexer.

error_func:
    An error function. Will be called with an error
    message, line and column as arguments, in case of
    an error during lexing.

type_lookup_func:
    A type lookup function. Given a string, it must
    return True IFF this string is a name of a type
    that was defined with a typedef earlier.
```

Overrides: object.__init__

build(*self*, ***kwargs*)

Builds the lexer from the specification. Must be called after the lexer object is created.

This method exists separately, because the PLY manual warns against calling lex.lex inside __init__

reset_lineno(*self*)

Resets the internal line number counter of the lexer.

input(*self*, *text*)

token(*self*)

t_PPHASH(*self*, *t*)

[\t]*\#

t_ppline_FILENAME(*self*, *t*)

"([^"\\\n]|(\\(([a-zA-Z._~!=&\^\-\\?'"])|([0-7]{1,3})|(x[0-9a-fA-F]+))))*"

t_ppline_LINE_NUMBER(*self*, *t*)

(0(u?ll|U?LL|([uU][lL])|([lL][uU])|[uU]|[lL])?)|([1-9][0-9]*(u?ll|U?LL|([uU][lL])|([lL][uU])|[uU]|[lL])?)

t_ppline_NEWLINE(*self*, *t*)

\n

t_ppline_PPLINE(*self*, *t*)

line

t_ppline_error(*self*, *t*)

t_NEWLINE(*self*, *t*)

\n+

t_FLOAT_CONST(*self*, *t*)

(((((([0-9]*\.[0-9]+)|([0-9]+\.))([eE][-+]?[0-9]+)?)|([0-9]+([eE][-+]?[0-9]+)))[FfLl]?)

t_INT_CONST_HEX(*self*, *t*)

0[xX][0-9a-fA-F]+(u?ll|U?LL|([uU][lL])|([lL][uU])|[uU]|[lL])?

t_BAD_CONST_OCT(*self*, *t*)

0[0-7]*[89]

t_INT_CONST_OCT(*self*, *t*)

0[0-7]*(u?ll|U?LL|([uU][lL])|([lL][uU])|[uU]|[lL])?

t_INT_CONST_DEC(*self*, *t*)

(0(u?ll|U?LL|([uU][lL])|([lL][uU])|[uU]|[lL])?)|([1-9][0-9]*(u?ll|U?LL|([uU][lL])|([lL][uU])|[uU]|[lL])?)

t_CHAR_CONST(*self*, *t*)

'([^'\\\n]|(\\(([a-zA-Z._~!=&\^\-\\?'"])|([0-7]{1,3})|(x[0-9a-fA-F]+))))'

t_WCHAR_CONST(*self*, *t*)

L'([^'\\\n]|(\\(([a-zA-Z._~!=&\^\-\\?'"])|([0-7]{1,3})|(x[0-9a-fA-F]+))))'

t_UNMATCHED_QUOTE(*self*, *t*)

('([^'\\\n]|(\\(([a-zA-Z._~!=&\^\-\\?'"])|([0-7]{1,3})|(x[0-9a-fA-F]+))))*\n)|('([^'\\\n]|(\\(([a-zA-Z._~!=&\^\-\\?'"])|([0-7]{1,3})|(x[0-9a-fA-F]+))))*$)

t_BAD_CHAR_CONST(*self*, *t*)

('([^'\\\n]|(\\(([a-zA-Z._~!=&\^\-\\?'"])|([0-7]{1,3})|(x[0-9a-fA-F]+))))[^']+')|('')|('([\\][^a-zA-Z._~^!=&\^\-\\?'"x0-7])[^'\n]*')

t_WSTRING_LITERAL(*self*, *t*)

L"([^"\\\n]|(\\(([a-zA-Z._~!=&\^\-\\?'"])|([0-7]{1,3})|(x[0-9a-fA-F]+))))*"

t_BAD_STRING_LITERAL(*self*, *t*)

"([^"\\\n]|(\\(([a-zA-Z._~!=&\^\-\\?'"])|([0-7]{1,3})|(x[0-9a-fA-F]+))))*([\\][^a-zA-Z._~^!=&\^\-\\?'"x0-7])([^"\\\n]|(\\(([a-zA-Z._~!=&\^\-\\?'"])|([0-7]{1,3})|(x[0-9a-fA-F]+))))*"

t_ID(*self*, *t*)

[a-zA-Z_][0-9a-zA-Z_]*

t_error(*self*, *t*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

31.2.2 Properties

Name	Description
Inherited from object	
__class__	

31.2.3 Class Variables

Name	Description
keywords	**Value:** ('AUTO', '_BOOL', 'BREAK', 'CASE', 'CHAR', 'CONST', 'CONT...
keyword_map	**Value:** {'_Bool': '_BOOL', 'auto': 'AUTO', 'break': 'BREAK', 'cas...
tokens	**Value:** ('AUTO', '_BOOL', 'BREAK', 'CASE', 'CHAR', 'CONST', 'CONT...
identifier	**Value:** '[a-zA-Z_][0-9a-zA-Z_]*'
integer_suffix_opt	**Value:** '(u?ll|U?LL|([uU][lL])|([lL][uU])|[uU]|[lL])?'
decimal_constant	**Value:** '(0(u?ll|U?LL|([uU][lL])|([lL][uU])|[uU]|[lL])?)|([1-9][0...
octal_constant	**Value:** '0[0-7]*(u?ll|U?LL|([uU][lL])|([lL][uU])|[uU]|[lL])?'
hex_constant	**Value:** '0[xX][0-9a-fA-F]+(u?ll|U?LL|([uU][lL])|([lL][uU])|[uU]|[...
bad_octal_constant	**Value:** '0[0-7]*[89]'
simple_escape	**Value:** '([a-zA-Z._~!=&\\^\\-\\\\?\'"])'
octal_escape	**Value:** '([0-7]{1,3})'
hex_escape	**Value:** '(x[0-9a-fA-F]+)'
bad_escape	**Value:** '([\\\\][^a-zA-Z._~^!=&\\^\\-\\\\?\'"x0-7])'
escape_sequence	**Value:** '(\\\\(([a-zA-Z._~!=&\\^\\-\\\\?\'"])|([0-7]{1,3})|(x[0-9...
cconst_char	**Value:** '([^\'\\\n]|(\\\\(([a-zA-Z._~!=&\\^\\-\\\\?\'"])|([0-7...
char_const	**Value:** '\'([^\'\\\n]|(\\\\(([a-zA-Z._~!=&\\^\\-\\\\?\'"])|([0...
wchar_const	**Value:** 'L\'([^\'\\\n]|(\\\\(([a-zA-Z._~!=&\\^\\-\\\\?\'"])|([...
unmatched_quote	**Value:** '(\'([^\'\\\n]|(\\\\(([a-zA-Z._~!=&\\^\\-\\\\?\'"])|([...
bad_char_const	**Value:** '(\'([^\'\\\n]|(\\\\(([a-zA-Z._~!=&\\^\\-\\\\?\'"])|([...
string_char	**Value:** '([^"\\\n]|(\\\\(([a-zA-Z._~!=&\\^\\-\\\\?\'"])|([0-7]...
string_literal	**Value:** '"([^"\\\n]|(\\\\(([a-zA-Z._~!=&\\^\\-\\\\?\'"])|([0-7...
wstring_literal	**Value:** 'L"([^"\\\n]|(\\\\(([a-zA-Z._~!=&\\^\\-\\\\?\'"])|([0-...
bad_string_literal	**Value:** '"([^"\\\n]|(\\\\(([a-zA-Z._~!=&\\^\\-\\\\?\'"])|([0-7...

continued on next page

Name	Description
exponent_part	**Value:** '([eE][-+]?[0-9]+)'
fractional_constant	**Value:** '([0-9]*\\.[0-9]+)|([0-9]+\\.)'
floating_constant	**Value:** '(((((([0-9]*\\.[0-9]+)|([0-9]+\\.))([eE][-+]?[0-9]+)?)|([..
states	**Value:** (('ppline', 'exclusive'))
t_ppline_ignore	**Value:** ' \t'
t_ignore	**Value:** ' \t'
t_PLUS	**Value:** '\\+'
t_MINUS	**Value:** '-'
t_TIMES	**Value:** '*'
t_DIVIDE	**Value:** '/'
t_MOD	**Value:** '%'
t_OR	**Value:** '\\\|'
t_AND	**Value:** '&'
t_NOT	**Value:** '~'
t_XOR	**Value:** '\\^'
t_LSHIFT	**Value:** '<<'
t_RSHIFT	**Value:** '>>'
t_LOR	**Value:** '\\\|\\\|'
t_LAND	**Value:** '&&'
t_LNOT	**Value:** '!'
t_LT	**Value:** '<'
t_GT	**Value:** '>'
t_LE	**Value:** '<='
t_GE	**Value:** '>='
t_EQ	**Value:** '=='
t_NE	**Value:** '!='
t_EQUALS	**Value:** '='
t_TIMESEQUAL	**Value:** '*='
t_DIVEQUAL	**Value:** '/='
t_MODEQUAL	**Value:** '%='
t_PLUSEQUAL	**Value:** '\\+='
t_MINUSEQUAL	**Value:** '-='
t_LSHIFTEQUAL	**Value:** '<<='
t_RSHIFTEQUAL	**Value:** '>>='
t_ANDEQUAL	**Value:** '&='
t_OREQUAL	**Value:** '\\\|='
t_XOREQUAL	**Value:** '\\^='
t_PLUSPLUS	**Value:** '\\+\\+'
t_MINUSMINUS	**Value:** '--'
t_ARROW	**Value:** '->'
t_CONDOP	**Value:** '\\?'

continued on next page

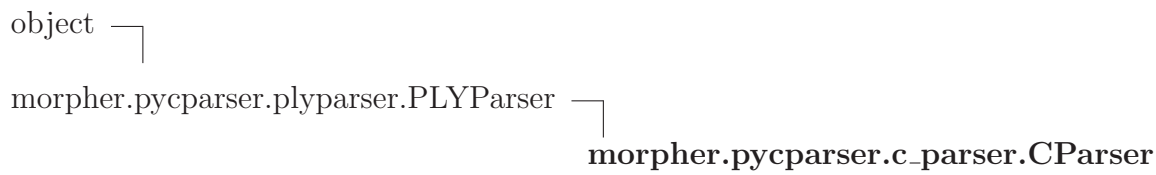
Name	Description
t_LPAREN	**Value:** '\\('
t_RPAREN	**Value:** '\\)'
t_LBRACKET	**Value:** '\\['
t_RBRACKET	**Value:** '\\]'
t_LBRACE	**Value:** '\\{'
t_RBRACE	**Value:** '\\}'
t_COMMA	**Value:** ','
t_PERIOD	**Value:** '\\.'
t_SEMI	**Value:** ';'
t_COLON	**Value:** ':'
t_ELLIPSIS	**Value:** '\\.\\.\\.'
t_STRING_LITERAL	**Value:** '"([^"\\\\\\n]|(\\\\(([a-zA-Z._~!=&\\^\\-\\\\?\'"])|([0-7...
keyword	**Value:** 'WHILE'

32 Module morpher.pycparser.c_parser

32.1 Variables

Name	Description
__package__	**Value:** `'morpher.pycparser'`

32.2 Class CParser



32.2.1 Methods

__init__(*self*, *lex_optimize*=True, *lextab*='pycparser.lextab', *yacc_optimize*=True, *yacctab*='pycparser.yacctab', *yacc_debug*=False)

```
Create a new CParser.

Some arguments for controlling the debug/optimization
level of the parser are provided. The defaults are
tuned for release/performance mode.
The simple rules for using them are:
*) When tweaking CParser/CLexer, set these to False
*) When releasing a stable parser, set to True

lex_optimize:
    Set to False when you're modifying the lexer.
    Otherwise, changes in the lexer won't be used, if
    some lextab.py file exists.
    When releasing with a stable lexer, set to True
    to save the re-generation of the lexer table on
    each run.

lextab:
    Points to the lex table that's used for optimized
    mode. Only if you're modifying the lexer and want
    some tests to avoid re-generating the table, make
    this point to a local lex table file (that's been
    earlier generated with lex_optimize=True)

yacc_optimize:
    Set to False when you're modifying the parser.
    Otherwise, changes in the parser won't be used, if
    some parsetab.py file exists.
    When releasing with a stable parser, set to True
    to save the re-generation of the parser table on
    each run.

yacctab:
    Points to the yacc table that's used for optimized
    mode. Only if you're modifying the parser, make
    this point to a local yacc table file

yacc_debug:
    Generate a parser.out file that explains how yacc
    built the parsing table from the grammar.
```

Overrides: object.__init__

parse(*self*, *text*, *filename*='', *debuglevel*=0)

```
Parses C code and returns an AST.

text:
    A string containing the C source code

filename:
    Name of the file being parsed (for meaningful
    error messages)

debuglevel:
    Debug level to yacc
```

p_translation_unit_1(*self*, *p*)

translation_unit : external_declaration

p_translation_unit_2(*self*, *p*)

translation_unit : translation_unit external_declaration

p_external_declaration_1(*self*, *p*)

external_declaration : function_definition

p_external_declaration_2(*self*, *p*)

external_declaration : declaration

p_external_declaration_3(*self*, *p*)

external_declaration : pp_directive

p_external_declaration_4(*self*, *p*)

external_declaration : SEMI

p_pp_directive(*self*, *p*)

pp_directive : PPHASH

p_function_definition_1(*self*, *p*)

function_definition : declarator declaration_list_opt compound_statement

p_function_definition_2(*self*, *p*)

function_definition : declaration_specifiers declarator declaration_list_opt compound_statement

p_statement(*self*, *p*)

statement : labeled_statement | expression_statement | compound_statement | selection_statement | iteration_statement | jump_statement

p_decl_body(*self*, *p*)

decl_body : declaration_specifiers init_declarator_list_opt

p_declaration(*self*, *p*)

declaration : decl_body SEMI

p_declaration_list(*self*, *p*)

declaration_list : declaration | declaration_list declaration

p_declaration_specifiers_1(*self*, *p*)

declaration_specifiers : type_qualifier declaration_specifiers_opt

p_declaration_specifiers_2(*self*, *p*)

declaration_specifiers : type_specifier declaration_specifiers_opt

p_declaration_specifiers_3(*self*, *p*)

declaration_specifiers : storage_class_specifier declaration_specifiers_opt

p_declaration_specifiers_4(*self*, *p*)

declaration_specifiers : function_specifier declaration_specifiers_opt

p_storage_class_specifier(*self*, *p*)

storage_class_specifier : AUTO | REGISTER | STATIC | EXTERN | TYPEDEF

p_function_specifier(*self*, *p*)

function_specifier : INLINE

p_type_specifier_1(*self*, *p*)

type_specifier : VOID | _BOOL | CHAR | SHORT | INT | LONG | FLOAT | DOUBLE | SIGNED | UNSIGNED | typedef_name | enum_specifier | struct_or_union_specifier

p_type_qualifier(*self*, *p*)

type_qualifier : CONST | RESTRICT | VOLATILE

p_init_declarator_list(*self*, *p*)

init_declarator_list : init_declarator | init_declarator_list COMMA init_declarator

p_init_declarator(*self*, *p*)

init_declarator : declarator | declarator EQUALS initializer

p_specifier_qualifier_list_1(*self*, *p*)

specifier_qualifier_list : type_qualifier specifier_qualifier_list_opt

p_specifier_qualifier_list_2(*self*, *p*)

specifier_qualifier_list : type_specifier specifier_qualifier_list_opt

p_struct_or_union_specifier_1(*self*, *p*)

struct_or_union_specifier : struct_or_union ID | struct_or_union TYPEID

p_struct_or_union_specifier_2(*self*, *p*)

struct_or_union_specifier : struct_or_union brace_open struct_declaration_list brace_close

p_struct_or_union_specifier_3(*self*, *p*)

struct_or_union_specifier : struct_or_union ID brace_open struct_declaration_list brace_close | struct_or_union TYPEID brace_open struct_declaration_list brace_close

p_struct_or_union(*self*, *p*)

struct_or_union : STRUCT | UNION

p_struct_declaration_list(*self*, *p*)

struct_declaration_list : struct_declaration | struct_declaration_list struct_declaration

p_struct_declaration_1(*self*, *p*)

struct_declaration : specifier_qualifier_list struct_declarator_list_opt SEMI

p_struct_declarator_list(*self*, *p*)

struct_declarator_list : struct_declarator | struct_declarator_list COMMA struct_declarator

p_struct_declarator_1(*self*, *p*)

struct_declarator : declarator

p_struct_declarator_2(*self*, *p*)

struct_declarator : declarator COLON constant_expression | COLON constant_expression

p_enum_specifier_1(*self*, *p*)

enum_specifier : ENUM ID | ENUM TYPEID

p_enum_specifier_2(*self*, *p*)

enum_specifier : ENUM brace_open enumerator_list brace_close

p_enum_specifier_3(*self*, *p*)

enum_specifier : ENUM ID brace_open enumerator_list brace_close | ENUM TYPEID brace_open enumerator_list brace_close

p_enumerator_list(*self*, *p*)

enumerator_list : enumerator | enumerator_list COMMA | enumerator_list COMMA enumerator

p_enumerator(*self*, *p*)

enumerator : ID | ID EQUALS constant_expression

p_declarator_1(*self*, *p*)

declarator : direct_declarator

p_declarator_2(*self*, *p*)

declarator : pointer direct_declarator

p_direct_declarator_1(*self*, *p*)

direct_declarator : ID

p_direct_declarator_2(*self*, *p*)

direct_declarator : LPAREN declarator RPAREN

p_direct_declarator_3(*self*, *p*)

direct_declarator : direct_declarator LBRACKET assignment_expression_opt RBRACKET

p_direct_declarator_4(*self*, *p*)

direct_declarator : direct_declarator LBRACKET TIMES RBRACKET

p_direct_declarator_5(*self*, *p*)

direct_declarator : direct_declarator LPAREN parameter_type_list RPAREN | direct_declarator LPAREN identifier_list_opt RPAREN

p_pointer(*self*, *p*)

pointer : TIMES type_qualifier_list_opt | TIMES type_qualifier_list_opt pointer

p_type_qualifier_list(*self*, *p*)

type_qualifier_list : type_qualifier | type_qualifier_list type_qualifier

p_parameter_type_list(*self*, *p*)

parameter_type_list : parameter_list | parameter_list COMMA ELLIPSIS

p_parameter_list(*self*, *p*)

parameter_list : parameter_declaration | parameter_list COMMA parameter_declaration

p_parameter_declaration_1(*self*, *p*)

parameter_declaration : declaration_specifiers declarator

p_parameter_declaration_2(*self*, *p*)

parameter_declaration : declaration_specifiers abstract_declarator_opt

p_identifier_list(*self*, *p*)

identifier_list : identifier | identifier_list COMMA identifier

p_initializer_1(*self*, *p*)

initializer : assignment_expression

p_initializer_2(*self*, *p*)

initializer : brace_open initializer_list brace_close | brace_open initializer_list COMMA brace_close

p_initializer_list(*self*, *p*)

initializer_list : designation_opt initializer | initializer_list COMMA designation_opt initializer

p_designation(*self*, *p*)

designation : designator_list EQUALS

p_designator_list(*self*, *p*)

designator_list : designator | designator_list designator

p_designator(*self*, *p*)

designator : LBRACKET constant_expression RBRACKET | PERIOD identifier

p_type_name(*self*, *p*)

type_name : specifier_qualifier_list abstract_declarator_opt

p_abstract_declarator_1(*self*, *p*)

abstract_declarator : pointer

p_abstract_declarator_2(*self*, *p*)

abstract_declarator : pointer direct_abstract_declarator

p_abstract_declarator_3(*self*, *p*)

abstract_declarator : direct_abstract_declarator

p_direct_abstract_declarator_1(*self*, *p*)

direct_abstract_declarator : LPAREN abstract_declarator RPAREN

p_direct_abstract_declarator_2(*self*, *p*)

direct_abstract_declarator : direct_abstract_declarator LBRACKET assignment_expression_opt RBRACKET

p_direct_abstract_declarator_3(*self*, *p*)

direct_abstract_declarator : LBRACKET assignment_expression_opt RBRACKET

p_direct_abstract_declarator_4(*self*, *p*)

direct_abstract_declarator : direct_abstract_declarator LBRACKET TIMES RBRACKET

p_direct_abstract_declarator_5(*self*, *p*)

direct_abstract_declarator : LBRACKET TIMES RBRACKET

p_direct_abstract_declarator_6(*self*, *p*)

direct_abstract_declarator : direct_abstract_declarator LPAREN parameter_type_list_opt RPAREN

p_direct_abstract_declarator_7(*self*, *p*)

direct_abstract_declarator : LPAREN parameter_type_list_opt RPAREN

p_block_item(*self*, *p*)

block_item : declaration | statement

p_block_item_list(*self*, *p*)

block_item_list : block_item | block_item_list block_item

p_compound_statement_1(*self*, *p*)

compound_statement : brace_open block_item_list_opt brace_close

p_labeled_statement_1(*self*, *p*)

labeled_statement : ID COLON statement

p_labeled_statement_2(*self*, *p*)

labeled_statement : CASE constant_expression COLON statement

p_labeled_statement_3(*self*, *p*)

labeled_statement : DEFAULT COLON statement

p_selection_statement_1(*self*, *p*)

selection_statement : IF LPAREN expression RPAREN statement

p_selection_statement_2(*self*, *p*)

selection_statement : IF LPAREN expression RPAREN statement ELSE statement

p_selection_statement_3(*self*, *p*)

selection_statement : SWITCH LPAREN expression RPAREN statement

p_iteration_statement_1(*self*, *p*)

iteration_statement : WHILE LPAREN expression RPAREN statement

p_iteration_statement_2(*self*, *p*)

iteration_statement : DO statement WHILE LPAREN expression RPAREN SEMI

p_iteration_statement_3(*self*, *p*)

iteration_statement : FOR LPAREN expression_opt SEMI expression_opt SEMI expression_opt RPAREN statement

p_iteration_statement_4(*self*, *p*)

iteration_statement : FOR LPAREN declaration expression_opt SEMI expression_opt RPAREN statement

p_jump_statement_1(*self*, *p*)

jump_statement : GOTO ID SEMI

p_jump_statement_2(*self*, *p*)

jump_statement : BREAK SEMI

p_jump_statement_3(*self*, *p*)

jump_statement : CONTINUE SEMI

p_jump_statement_4(*self*, *p*)

jump_statement : RETURN expression SEMI | RETURN SEMI

p_expression_statement(*self*, *p*)

expression_statement : expression_opt SEMI

p_expression(*self*, *p*)

expression : assignment_expression | expression COMMA assignment_expression

p_typedef_name(*self*, *p*)

typedef_name : TYPEID

p_assignment_expression(*self*, *p*)

assignment_expression : conditional_expression | unary_expression assignment_operator assignment_expression

p_assignment_operator(*self*, *p*)

assignment_operator : EQUALS | XOREQUAL | TIMESEQUAL | DIVEQUAL | MODEQUAL | PLUSEQUAL | MINUSEQUAL | LSHIFTEQUAL | RSHIFTEQUAL | ANDEQUAL | OREQUAL

p_constant_expression(*self*, *p*)

constant_expression : conditional_expression

p_conditional_expression(*self*, *p*)

conditional_expression : binary_expression | binary_expression CONDOP expression COLON conditional_expression

p_binary_expression(*self*, *p*)

binary_expression : cast_expression | binary_expression TIMES binary_expression | binary_expression DIVIDE binary_expression | binary_expression MOD binary_expression | binary_expression PLUS binary_expression | binary_expression MINUS binary_expression | binary_expression RSHIFT binary_expression | binary_expression LSHIFT binary_expression | binary_expression LT binary_expression | binary_expression LE binary_expression | binary_expression GE binary_expression | binary_expression GT binary_expression | binary_expression EQ binary_expression | binary_expression NE binary_expression | binary_expression AND binary_expression | binary_expression OR binary_expression | binary_expression XOR binary_expression | binary_expression LAND binary_expression | binary_expression LOR binary_expression

p_cast_expression_1(*self*, *p*)

cast_expression : unary_expression

p_cast_expression_2(*self*, *p*)

cast_expression : LPAREN type_name RPAREN cast_expression

p_unary_expression_1(*self*, *p*)

unary_expression : postfix_expression

p_unary_expression_2(*self*, *p*)

unary_expression : PLUSPLUS unary_expression | MINUSMINUS unary_expression | unary_operator cast_expression

p_unary_expression_3(*self*, *p*)

unary_expression : SIZEOF unary_expression | SIZEOF LPAREN type_name RPAREN

p_unary_operator(*self*, *p*)

unary_operator : AND | TIMES | PLUS | MINUS | NOT | LNOT

p_postfix_expression_1(*self*, *p*)

postfix_expression : primary_expression

p_postfix_expression_2(*self*, *p*)

postfix_expression : postfix_expression LBRACKET expression RBRACKET

p_postfix_expression_3(*self*, *p*)

postfix_expression : postfix_expression LPAREN argument_expression_list RPAREN | postfix_expression LPAREN RPAREN

p_postfix_expression_4(*self*, *p*)

postfix_expression : postfix_expression PERIOD identifier | postfix_expression ARROW identifier

p_postfix_expression_5(*self*, *p*)

postfix_expression : postfix_expression PLUSPLUS | postfix_expression MINUSMINUS

p_postfix_expression_6(*self*, *p*)

postfix_expression : LPAREN type_name RPAREN brace_open initializer_list brace_close | LPAREN type_name RPAREN brace_open initializer_list COMMA brace_close

p_primary_expression_1(*self*, *p*)

primary_expression : identifier

p_primary_expression_2(*self*, *p*)

primary_expression : constant

p_primary_expression_3(*self*, *p*)

primary_expression : unified_string_literal | unified_wstring_literal

p_primary_expression_4(*self*, *p*)

primary_expression : LPAREN expression RPAREN

p_argument_expression_list(*self*, *p*)

argument_expression_list : assignment_expression | argument_expression_list COMMA assignment_expression

p_identifier(*self*, *p*)

identifier : ID

p_constant_1(*self*, *p*)

constant : INT_CONST_DEC | INT_CONST_OCT | INT_CONST_HEX

p_constant_2(*self*, *p*)

constant : FLOAT_CONST

p_constant_3(*self*, *p*)

constant : CHAR_CONST | WCHAR_CONST

p_unified_string_literal(*self*, *p*)

unified_string_literal : STRING_LITERAL | unified_string_literal STRING_LITERAL

p_unified_wstring_literal(*self*, *p*)

unified_wstring_literal : WSTRING_LITERAL | unified_wstring_literal WSTRING_LITERAL

p_brace_open(*self*, *p*)

brace_open : LBRACE

p_brace_close(*self*, *p*)

brace_close : RBRACE

p_empty(*self*, *p*)

empty :

p_error(*self*, *p*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

32.2.2 Properties

Name	Description
Inherited from object	
__class__	

32.2.3 Class Variables

Name	Description
precedence	**Value:** `(('left', 'LOR'), ('left', 'LAND'), ('left', 'OR'), ('lef...`

33 Module morpher.pycparser.lextab

33.1 Variables

Name	Description
__package__	**Value:** `None`

34 Module morpher.pycparser.plyparser

34.1 Variables

Name	Description
__package__	**Value:** `None`

34.2 Class Coord

object
└
morpher.pycparser.plyparser.Coord

Coordinates of a syntactic element. Consists of:

- File name
- Line number
- (optional) column number, for the Lexer

34.2.1 Methods

__init__(*self*, *file*, *line*, *column*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

str(x)

Overrides: object.__str__ extit(inherited documentation)

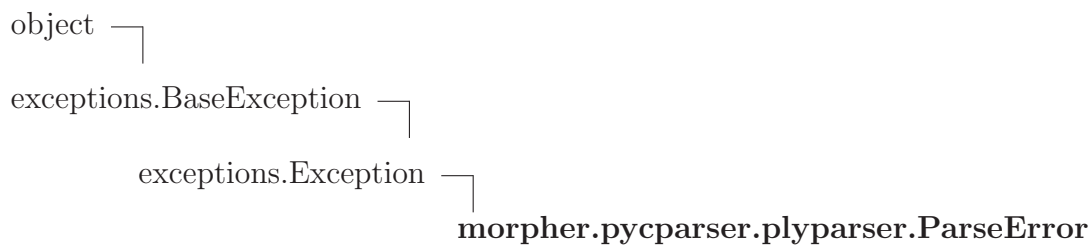
Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

34.2.2 Properties

Name	Description
Inherited from object	
__class__	

34.3 Class ParseError



34.3.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

34.3.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

34.4 Class PLYParser

Known Subclasses: morpher.pycparser.c_parser.CParser

34.4.1 Methods

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

34.4.2 Properties

Name	Description
Inherited from object	
__class__	

35 Module morpher.pycparser.yacctab

35.1 Variables

Name	Description
__package__	**Value:** `None`

36 Package morpher.pydbg

Author: Pedram Amini

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

36.1 Modules

37 Module morpher.pydbg.breakpoint

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

37.1 Variables

Name	Description
__package__	**Value:** `None`

37.2 Class breakpoint

Soft breakpoint object.

37.2.1 Methods

__init__(*self*, *address*=`None`, *original_byte*=`None`, *description*=`''`, *restore*=`True`, *handler*=`None`)

Parameters

`address`: Address of breakpoint
(type=DWORD)

`original_byte`: Original byte stored at breakpoint address
(type=Byte)

`description`: (Optional) Description of breakpoint
(type=String)

`restore`: (Optional, def=True) Flag controlling whether or not to restore the breakpoint
(type=Boolean)

`handler`: (Optional, def=None) Optional handler to call for this bp instead of the default handler
(type=Function Pointer)

37.2.2 Class Variables

Name	Description
address	**Value:** `None`
original_byte	**Value:** `None`
description	**Value:** `None`
restore	**Value:** `None`
handler	**Value:** `None`

38 Module morpher.pydbg.defines

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

38.1 Variables

Name	Description
TH32CS_SNAPHEAPLIST	**Value:** 1
TH32CS_SNAPPROCESS	**Value:** 2
TH32CS_SNAPTHREAD	**Value:** 4
TH32CS_SNAPMODULE	**Value:** 8
TH32CS_INHERIT	**Value:** 2147483648
TH32CS_SNAPALL	**Value:** 15
EXCEPTION_DEBUG_EVENT	**Value:** 1
CREATE_THREAD_DEBUG_EVENT	**Value:** 2
CREATE_PROCESS_DEBUG_EVENT	**Value:** 3
EXIT_THREAD_DEBUG_EVENT	**Value:** 4
EXIT_PROCESS_DEBUG_EVENT	**Value:** 5
LOAD_DLL_DEBUG_EVENT	**Value:** 6
UNLOAD_DLL_DEBUG_EVENT	**Value:** 7
OUTPUT_DEBUG_STRING_EVENT	**Value:** 8
RIP_EVENT	**Value:** 9
USER_CALLBACK_DEBUG_EVENT	**Value:** 3735928559
EXCEPTION_ACCESS_VIOLATION	**Value:** 3221225477
EXCEPTION_BREAKPOINT	**Value:** 2147483651

continued on next page

Name	Description
EXCEPTION_GUARD_PAGE	**Value:** 2147483649
EXCEPTION_SINGLE_STEP	**Value:** 2147483652
HW_ACCESS	**Value:** 3
HW_EXECUTE	**Value:** 0
HW_WRITE	**Value:** 1
CONTEXT_CONTROL	**Value:** 65537
CONTEXT_FULL	**Value:** 65543
CONTEXT_DEBUG_REGISTERS	**Value:** 65552
CREATE_NEW_CONSOLE	**Value:** 16
DBG_CONTINUE	**Value:** 65538
DBG_EXCEPTION_NOT_HANDLED	**Value:** 2147549185
DBG_EXCEPTION_HANDLED	**Value:** 65537
DEBUG_PROCESS	**Value:** 1
DEBUG_ONLY_THIS_PROCESS	**Value:** 2
EFLAGS_RF	**Value:** 65536
EFLAGS_TRAP	**Value:** 256
ERROR_NO_MORE_FILES	**Value:** 18
FILE_MAP_READ	**Value:** 4
FORMAT_MESSAGE_ALLOCATE_BUFFER	**Value:** 256
FORMAT_MESSAGE_FROM_SYSTEM	**Value:** 4096
INVALID_HANDLE_VALUE	**Value:** 4294967295
MEM_COMMIT	**Value:** 4096
MEM_DECOMMIT	**Value:** 16384
MEM_IMAGE	**Value:** 16777216
MEM_RELEASE	**Value:** 32768
PAGE_NOACCESS	**Value:** 1
PAGE_READONLY	**Value:** 2
PAGE_READWRITE	**Value:** 4
PAGE_WRITECOPY	**Value:** 8
PAGE_EXECUTE	**Value:** 16
PAGE_EXECUTE_READ	**Value:** 32

continued on next page

Name	Description
PAGE_EXECUTE_READ-WRITE	**Value:** 64
PAGE_EXECUTE_WRIT-ECOPY	**Value:** 128
PAGE_GUARD	**Value:** 256
PAGE_NOCACHE	**Value:** 512
PAGE_WRITECOMBIN-E	**Value:** 1024
PROCESS_ALL_ACCESS	**Value:** 2035711
SE_PRIVILEGE_ENABL-ED	**Value:** 2
SW_SHOW	**Value:** 5
THREAD_ALL_ACCESS	**Value:** 2032639
TOKEN_ADJUST_PRIVI-LEGES	**Value:** 32
UDP_TABLE_OWNER_P-ID	**Value:** 1
VIRTUAL_MEM	**Value:** 12288
SysDbgReadMsr	**Value:** 16
SysDbgWriteMsr	**Value:** 17
AF_INET	**Value:** 2
AF_INET6	**Value:** 23
MIB_TCP_STATE_LISTE-N	**Value:** 2
TCP_TABLE_OWNER_P-ID_ALL	**Value:** 5
DEFAULT_MODE	**Value:** 0
GetLastError	**Value:** <_FuncPtr object at 0x0253AA08>
RTLD_GLOBAL	**Value:** 0
RTLD_LOCAL	**Value:** 0
__package__	**Value:** 'morpher.pydbg'
c_types	**Value:** (<type '_ctypes.Structure'>, <class 'ctypes.c_char'>, <cl...
cdll	**Value:** <ctypes.LibraryLoader object at 0x0249FB90>
memmove	**Value:** <CFunctionType object at 0x0253AA80>
memset	**Value:** <CFunctionType object at 0x0253AAF8>
oledll	**Value:** <ctypes.LibraryLoader object at 0x0249FC30>
pydll	**Value:** <ctypes.LibraryLoader object at 0x0249FBB0>

continued on next page

Name	Description
pythonapi	**Value:** <PyDLL 'python dll', handle 1e000000 at 249fbd0>
windll	**Value:** <ctypes.LibraryLoader object at 0x0249FBF0>

38.2 Class THREADENTRY32

object
??._CData
_ctypes.Structure
morpher.pydbg.defines.THREADENTRY32

38.2.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

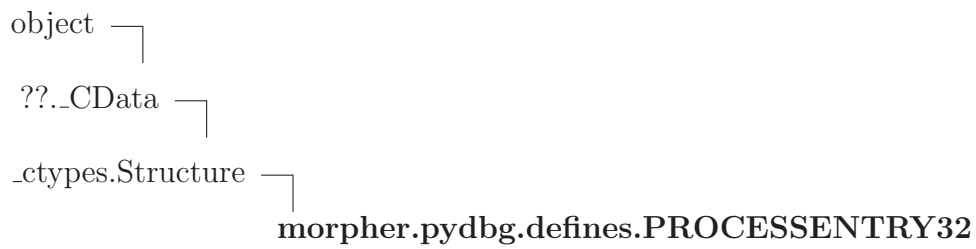
38.2.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

38.2.3 Class Variables

Name	Description
fields	**Value:** `[('dwSize', <class 'ctypes.c_ulong'>), ('cntUsage', <clas...`
cntUsage	**Value:** `<Field type=c_ulong, ofs=4, size=4>`
dwFlags	**Value:** `<Field type=c_ulong, ofs=24, size=4>`
dwSize	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
th32OwnerProcessID	**Value:** `<Field type=c_ulong, ofs=12, size=4>`
th32ThreadID	**Value:** `<Field type=c_ulong, ofs=8, size=4>`
tpBasePri	**Value:** `<Field type=c_ulong, ofs=16, size=4>`
tpDeltaPri	**Value:** `<Field type=c_ulong, ofs=20, size=4>`

38.3 Class PROCESSENTRY32



38.3.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

38.3.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

38.3.3 Class Variables

Name	Description
fields	**Value:** `[('dwSize', <class 'ctypes.c_ulong'>), ('cntUsage', <clas...`
cntThreads	**Value:** `<Field type=c_ulong, ofs=20, size=4>`
cntUsage	**Value:** `<Field type=c_ulong, ofs=4, size=4>`
dwFlags	**Value:** `<Field type=c_ulong, ofs=32, size=4>`
dwSize	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
pcPriClassBase	**Value:** `<Field type=c_ulong, ofs=28, size=4>`
szExeFile	**Value:** `<Field type=c_char_Array_260, ofs=36, size=260>`
th32DefaultHeapID	**Value:** `<Field type=c_ulong, ofs=12, size=4>`
th32ModuleID	**Value:** `<Field type=c_ulong, ofs=16, size=4>`
th32ParentProcessID	**Value:** `<Field type=c_ulong, ofs=24, size=4>`
th32ProcessID	**Value:** `<Field type=c_ulong, ofs=8, size=4>`

38.4 Class MODULEENTRY32

object
??._CData
_ctypes.Structure
morpher.pydbg.defines.MODULEENTRY32

38.4.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

38.4.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

38.4.3 Class Variables

Name	Description
fields	**Value:** `[('dwSize', <class 'ctypes.c_ulong'>), ('th32ModuleID', <...`
GlblcntUsage	**Value:** `<Field type=c_ulong, ofs=12, size=4>`
ProccntUsage	**Value:** `<Field type=c_ulong, ofs=16, size=4>`

continued on next page

Name	Description
dwSize	**Value:** <Field type=c_ulong, ofs=0, size=4>
hModule	**Value:** <Field type=c_ulong, ofs=28, size=4>
modBaseAddr	**Value:** <Field type=c_ulong, ofs=20, size=4>
modBaseSize	**Value:** <Field type=c_ulong, ofs=24, size=4>
szExePath	**Value:** <Field type=c_char_Array_260, ofs=288, size=260>
szModule	**Value:** <Field type=c_char_Array_256, ofs=32, size=256>
th32ModuleID	**Value:** <Field type=c_ulong, ofs=4, size=4>
th32ProcessID	**Value:** <Field type=c_ulong, ofs=8, size=4>

38.5 Class MIB_TCPTABLE_OWNER_PID

object
??._CData
_ctypes.Structure
morpher.pydbg.defines.MIB_TCPTABLE_OWNER_PID

38.5.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

38.5.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

38.5.3 Class Variables

Name	Description
fields	**Value:** `[('dwNumEntries', <class 'ctypes.c_ulong'>), ('table', <c...`
dwNumEntries	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
table	**Value:** `<Field type=_MIB_TCPROW_OWNER_PID_Array_512, ofs=4, size=...`

38.6 Class MIB_UDPTABLE_OWNER_PID

object
 ??._CData
  _ctypes.Structure
   morpher.pydbg.defines.MIB_UDPTABLE_OWNER_PID

38.6.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

38.6.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

38.6.3 Class Variables

Name	Description
fields	**Value:** `[('dwNumEntries', <class 'ctypes.c_ulong'>), ('table', <c...`
dwNumEntries	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
table	**Value:** `<Field type=_MIB_UDPROW_OWNER_PID_Array_512, ofs=4, size=...`

38.7 Class SYSDBG_MSR

object
??._CData
_ctypes.Structure
morpher.pydbg.defines.SYSDBG_MSR

38.7.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

38.7.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

38.7.3 Class Variables

Name	Description
fields	**Value:** `[('Address', <class 'ctypes.c_ulong'>), ('Data', <class '...`
Address	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
Data	**Value:** `<Field type=c_ulonglong, ofs=8, size=8>`

39 Module morpher.pydbg.hardware_breakpoint

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

39.1 Variables

Name	Description
__package__	**Value:** `None`

39.2 Class hardware_breakpoint

Hardware breakpoint object.

39.2.1 Methods

__init__(*self*, *address*=None, *length*=0, *condition*='', *description*='', *restore*=True, *slot*=None, *handler*=None)

Parameters

`address:` Address to set hardware breakpoint at

(type=DWORD)

`length:` Size of hardware breakpoint (byte, word or dword)

(type=Integer (1, 2 or 4))

`condition:` Condition to set the hardware breakpoint to activate on

(type=Integer (HW_ACCESS, HW_WRITE, HW_EXECUTE))

`description:` (Optional) Description of breakpoint

(type=String)

`restore:` (Optional, def=True) Flag controlling whether or not to restore the breakpoint

(type=Boolean)

`slot:` (Optional, Def=None) Debug register slot this hardware breakpoint sits in.

(type=Integer (0-3))

`handler:` (Optional, def=None) Optional handler to call for this bp instead of the default handler

(type=Function Pointer)

39.2.2 Class Variables

Name	Description
address	**Value:** `None`
length	**Value:** `None`
condition	**Value:** `None`
description	**Value:** `None`
restore	**Value:** `None`
slot	**Value:** `None`
handler	**Value:** `None`

40 Module morpher.pydbg.memory_breakpoint

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

40.1 Variables

Name	Description
__package__	**Value:** `'morpher.pydbg'`

40.2 Class memory_breakpoint

Memory breakpoint object.

40.2.1 Methods

__init__(*self*, *address*=`None`, *size*=`None`, *mbi*=`None`, *description*=`''`, *handler*=`None`)

Parameters

`address`: Address of breakpoint
(type=DWORD)

`size`: Size of buffer we want to break on
(type=Integer)

`mbi`: MEMORY_BASIC_INFORMATION of page containing buffer we want to break on
(type=MEMORY_BASIC_INFORMATION)

`description`: (Optional) Description of breakpoint
(type=String)

`handler`: (Optional, def=None) Optional handler to call for this bp instead of the default handler
(type=Function Pointer)

40.2.2 Class Variables

Name	Description
address	**Value:** `None`
size	**Value:** `None`
mbi	**Value:** `None`
description	**Value:** `None`
handler	**Value:** `None`
read_count	**Value:** `0`
split_count	**Value:** `0`
copy_depth	**Value:** `0`
id	**Value:** `0`
on_stack	**Value:** `False`

41 Module morpher.pydbg.memory_snapshot_block

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

41.1 Variables

Name	Description
__package__	**Value:** `None`

41.2 Class memory_snapshot_block

Memory block object, used in memory snapshots.

41.2.1 Methods

__init__(*self*, *mbi*=`None`, *data*=`None`)

Parameters

`mbi`: MEMORY_BASIC_INFORMATION of memory block
(type=MEMORY_BASIC_INFORMATION)

`data`: Raw bytes stored in memory block at time of snapshot
(type=Raw Bytes)

41.2.2 Class Variables

Name	Description
mbi	**Value:** `None`
data	**Value:** `None`

42 Module morpher.pydbg.memory_snapshot_context

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

42.1 Variables

Name	Description
__package__	**Value:** `None`

42.2 Class memory_snapshot_context

Thread context object, used in memory snapshots.

42.2.1 Methods

__init__(*self*, *thread_id*=`None`, *context*=`None`)

Parameters

`thread_id`: Thread ID
(type=Integer)

`context`: Context of thread specified by ID at time of snapshot
(type=CONTEXT)

42.2.2 Class Variables

Name	Description
thread_id	**Value:** `None`
context	**Value:** `None`

43 Module morpher.pydbg.my_ctypes

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

43.1 Variables

Name	Description
c_types	**Value:** (<type '_ctypes.Structure'>, <class 'ctypes.c_char'>, <cl...
__package__	**Value:** 'morpher.pydbg'

44 Module morpher.pydbg.pdx

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

44.1 Variables

Name	Description
kernel32	**Value:** <WinDLL 'kernel32', handle 76a70000 at 249fc70>
AF_INET	**Value:** 2
AF_INET6	**Value:** 23
CONTEXT_CONTROL	**Value:** 65537
CONTEXT_DEBUG_REGISTERS	**Value:** 65552
CONTEXT_FULL	**Value:** 65543
CREATE_NEW_CONSOLE	**Value:** 16
CREATE_PROCESS_DEBUG_EVENT	**Value:** 3
CREATE_THREAD_DEBUG_EVENT	**Value:** 2
DBG_CONTINUE	**Value:** 65538
DBG_EXCEPTION_HANDLED	**Value:** 65537
DBG_EXCEPTION_NOT_HANDLED	**Value:** 2147549185
DEBUG_ONLY_THIS_PROCESS	**Value:** 2
DEBUG_PROCESS	**Value:** 1
DEFAULT_MODE	**Value:** 0
EFLAGS_RF	**Value:** 65536
EFLAGS_TRAP	**Value:** 256
ERROR_NO_MORE_FILES	**Value:** 18
EXCEPTION_ACCESS_VIOLATION	**Value:** 3221225477
EXCEPTION_BREAKPOINT	**Value:** 2147483651

continued on next page

Name	Description
EXCEPTION_DEBUG_EVENT	**Value:** 1
EXCEPTION_GUARD_PAGE	**Value:** 2147483649
EXCEPTION_SINGLE_STEP	**Value:** 2147483652
EXIT_PROCESS_DEBUG_EVENT	**Value:** 5
EXIT_THREAD_DEBUG_EVENT	**Value:** 4
FILE_MAP_READ	**Value:** 4
FORMAT_MESSAGE_ALLOCATE_BUFFER	**Value:** 256
FORMAT_MESSAGE_FROM_SYSTEM	**Value:** 4096
GetLastError	**Value:** <_FuncPtr object at 0x0253AA08>
HW_ACCESS	**Value:** 3
HW_EXECUTE	**Value:** 0
HW_WRITE	**Value:** 1
INVALID_HANDLE_VALUE	**Value:** 4294967295
LOAD_DLL_DEBUG_EVENT	**Value:** 6
MEM_COMMIT	**Value:** 4096
MEM_DECOMMIT	**Value:** 16384
MEM_IMAGE	**Value:** 16777216
MEM_RELEASE	**Value:** 32768
MIB_TCP_STATE_LISTEN	**Value:** 2
OUTPUT_DEBUG_STRING_EVENT	**Value:** 8
PAGE_EXECUTE	**Value:** 16
PAGE_EXECUTE_READ	**Value:** 32
PAGE_EXECUTE_READWRITE	**Value:** 64
PAGE_EXECUTE_WRITECOPY	**Value:** 128
PAGE_GUARD	**Value:** 256
PAGE_NOACCESS	**Value:** 1
PAGE_NOCACHE	**Value:** 512
PAGE_READONLY	**Value:** 2
PAGE_READWRITE	**Value:** 4

continued on next page

Name	Description
PAGE_WRITECOMBINE	**Value:** 1024
PAGE_WRITECOPY	**Value:** 8
PROCESS_ALL_ACCESS	**Value:** 2035711
RIP_EVENT	**Value:** 9
RTLD_GLOBAL	**Value:** 0
RTLD_LOCAL	**Value:** 0
SE_PRIVILEGE_ENABLED	**Value:** 2
SW_SHOW	**Value:** 5
SysDbgReadMsr	**Value:** 16
SysDbgWriteMsr	**Value:** 17
TCP_TABLE_OWNER_PID_ALL	**Value:** 5
TH32CS_INHERIT	**Value:** 2147483648
TH32CS_SNAPALL	**Value:** 15
TH32CS_SNAPHEAPLIST	**Value:** 1
TH32CS_SNAPMODULE	**Value:** 8
TH32CS_SNAPPROCESS	**Value:** 2
TH32CS_SNAPTHREAD	**Value:** 4
THREAD_ALL_ACCESS	**Value:** 2032639
TOKEN_ADJUST_PRIVILEGES	**Value:** 32
UDP_TABLE_OWNER_PID	**Value:** 1
UNLOAD_DLL_DEBUG_EVENT	**Value:** 7
USER_CALLBACK_DEBUG_EVENT	**Value:** 3735928559
VIRTUAL_MEM	**Value:** 12288
__package__	**Value:** 'morpher.pydbg'
c_types	**Value:** (<type '_ctypes.Structure'>, <class 'ctypes.c_char'>, <cl...
cdll	**Value:** <ctypes.LibraryLoader object at 0x0249FB90>
memmove	**Value:** <CFunctionType object at 0x0253AA80>
memset	**Value:** <CFunctionType object at 0x0253AAF8>
oledll	**Value:** <ctypes.LibraryLoader object at 0x0249FC30>

continued on next page

Name	Description
pydll	**Value:** <ctypes.LibraryLoader object at 0x0249FBB0>
pythonapi	**Value:** <PyDLL 'python dll', handle 1e000000 at 249fbd0>
windll	**Value:** <ctypes.LibraryLoader object at 0x0249FBF0>

44.2 Class pdx

object
exceptions.BaseException
exceptions.Exception
morpher.pydbg.pdx.pdx

This class is used internally for raising custom exceptions and includes support for automated Windows error message resolution and formatting. For example, to raise a generic error you can use:

```
raise pdx("Badness occured.")
```

To raise a Windows API error you can use:

```
raise pdx("SomeWindowsApi()", True)
```

44.2.1 Methods

__init__(*self*, *message*, *win32_exception*=False)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__

__str__(*self*)

str(x)
Overrides: object.__str__ extit(inherited documentation)

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

44.2.2 Properties

Name	Description
Inherited from exceptions.BaseException	
args	
Inherited from object	
__class__	

44.2.3 Class Variables

Name	Description
message	**Value:** `None`
error_code	**Value:** `None`

45 Module morpher.pydbg.pydasm

45.1 Functions

get_instruction(...)

Decode an instruction from the given buffer.

Takes in a string containing the data to disassemble and the mode, either MODE_16 or MODE_32. Returns an Instruction object or None if the instruction can't be disassembled.

get_instruction_string(...)

Transform an instruction object into its string representation.

The function takes an Instruction object; its format, either FORMAT_INTEL or FORMAT_ATT and finally an offset (refer to libdasm for meaning). Returns a string representation of the disassembled instruction.

get_mnemonic_string(...)

Transform an instruction object's mnemonic into its string representation.

The function takes an Instruction object and its format, either FORMAT_INTEL or FORMAT_ATT. Returns a string representation of the mnemonic.

get_operand_string(...)

```
Transform an instruction object's operand into its string representation.

The function takes an Instruction object; the operand index (0,1,2);
 its format, either FORMAT_INTEL or FORMAT_ATT and finally an offset
(refer to libdasm for meaning). Returns a string representation of
the disassembled operand.
```

get_register_type(...)

Get the type of the register used by the operand.

The function takes an Operand object and returns a Long representing the type of the register.

45.2 Variables

Name	Description
FORMAT_ATT	**Value:** `0`
FORMAT_INTEL	**Value:** `1`
INSTRUCTION_TYPE_ADC	**Value:** `6`
INSTRUCTION_TYPE_ADD	**Value:** `4`
INSTRUCTION_TYPE_AND	**Value:** `33`
INSTRUCTION_TYPE_ASC	**Value:** `0`
INSTRUCTION_TYPE_BSF	**Value:** `50`
INSTRUCTION_TYPE_BSR	**Value:** `51`
INSTRUCTION_TYPE_BSWAP	**Value:** `52`
INSTRUCTION_TYPE_BT	**Value:** `46`
INSTRUCTION_TYPE_BTC	**Value:** `49`
INSTRUCTION_TYPE_BTR	**Value:** `48`
INSTRUCTION_TYPE_BTS	**Value:** `47`
INSTRUCTION_TYPE_CALL	**Value:** `42`
INSTRUCTION_TYPE_CLD	**Value:** `57`
INSTRUCTION_TYPE_CMP	**Value:** `30`
INSTRUCTION_TYPE_CMPS	**Value:** `21`
INSTRUCTION_TYPE_DCL	**Value:** `1`
INSTRUCTION_TYPE_DEC	**Value:** `10`
INSTRUCTION_TYPE_DIV	**Value:** `11`
INSTRUCTION_TYPE_EIMUL	**Value:** `26`
INSTRUCTION_TYPE_ENTER	**Value:** `44`

continued on next page

Name	Description
INSTRUCTION_TYPE_FADD	**Value:** 61
INSTRUCTION_TYPE_FADDP	**Value:** 62
INSTRUCTION_TYPE_FCMOVC	**Value:** 60
INSTRUCTION_TYPE_FCOM	**Value:** 79
INSTRUCTION_TYPE_FCOMI	**Value:** 82
INSTRUCTION_TYPE_FCOMIP	**Value:** 83
INSTRUCTION_TYPE_FCOMP	**Value:** 80
INSTRUCTION_TYPE_FCOMPP	**Value:** 81
INSTRUCTION_TYPE_FDIV	**Value:** 73
INSTRUCTION_TYPE_FDIVP	**Value:** 74
INSTRUCTION_TYPE_FDIVR	**Value:** 75
INSTRUCTION_TYPE_FDIVRP	**Value:** 76
INSTRUCTION_TYPE_FFREE	**Value:** 98
INSTRUCTION_TYPE_FFREEP	**Value:** 99
INSTRUCTION_TYPE_FIADD	**Value:** 63
INSTRUCTION_TYPE_FICOM	**Value:** 96
INSTRUCTION_TYPE_FICOMP	**Value:** 97
INSTRUCTION_TYPE_FIDIV	**Value:** 77
INSTRUCTION_TYPE_FIDIVR	**Value:** 78
INSTRUCTION_TYPE_FILD	**Value:** 95
INSTRUCTION_TYPE_FIMUL	**Value:** 72

continued on next page

Name	Description
INSTRUCTION_TYPE_FIST	**Value:** 91
INSTRUCTION_TYPE_FISTP	**Value:** 92
INSTRUCTION_TYPE_FISTTP	**Value:** 93
INSTRUCTION_TYPE_FISUB	**Value:** 66
INSTRUCTION_TYPE_FISUBR	**Value:** 69
INSTRUCTION_TYPE_FLD	**Value:** 94
INSTRUCTION_TYPE_FMUL	**Value:** 70
INSTRUCTION_TYPE_FMULP	**Value:** 71
INSTRUCTION_TYPE_FPU	**Value:** 103
INSTRUCTION_TYPE_FPU_CTRL	**Value:** 102
INSTRUCTION_TYPE_FST	**Value:** 89
INSTRUCTION_TYPE_FSTP	**Value:** 90
INSTRUCTION_TYPE_FSUB	**Value:** 64
INSTRUCTION_TYPE_FSUBP	**Value:** 65
INSTRUCTION_TYPE_FSUBR	**Value:** 67
INSTRUCTION_TYPE_FSUBRP	**Value:** 68
INSTRUCTION_TYPE_FUCOM	**Value:** 84
INSTRUCTION_TYPE_FUCOMI	**Value:** 87
INSTRUCTION_TYPE_FUCOMIP	**Value:** 88
INSTRUCTION_TYPE_FUCOMP	**Value:** 85
INSTRUCTION_TYPE_FUCOMPP	**Value:** 86

continued on next page

Name	Description
INSTRUCTION_TYPE_FXCH	**Value:** 100
INSTRUCTION_TYPE_IDIV	**Value:** 12
INSTRUCTION_TYPE_IMUL	**Value:** 25
INSTRUCTION_TYPE_INC	**Value:** 9
INSTRUCTION_TYPE_INT	**Value:** 45
INSTRUCTION_TYPE_JECXZ	**Value:** 38
INSTRUCTION_TYPE_JMP	**Value:** 36
INSTRUCTION_TYPE_JMPC	**Value:** 37
INSTRUCTION_TYPE_LEA	**Value:** 28
INSTRUCTION_TYPE_LFP	**Value:** 56
INSTRUCTION_TYPE_LODS	**Value:** 16
INSTRUCTION_TYPE_LOOP	**Value:** 41
INSTRUCTION_TYPE_MMX	**Value:** 104
INSTRUCTION_TYPE_MOV	**Value:** 2
INSTRUCTION_TYPE_MOVC	**Value:** 40
INSTRUCTION_TYPE_MOVS	**Value:** 18
INSTRUCTION_TYPE_MOVSR	**Value:** 3
INSTRUCTION_TYPE_MOVSX	**Value:** 19
INSTRUCTION_TYPE_MOVZX	**Value:** 20
INSTRUCTION_TYPE_MUL	**Value:** 24
INSTRUCTION_TYPE_NEG	**Value:** 14

continued on next page

Name	Description
INSTRUCTION_TYPE_NOT	**Value:** 13
INSTRUCTION_TYPE_OR	**Value:** 34
INSTRUCTION_TYPE_OTHER	**Value:** 106
INSTRUCTION_TYPE_POP	**Value:** 35
INSTRUCTION_TYPE_PRIV	**Value:** 107
INSTRUCTION_TYPE_PUSH	**Value:** 32
INSTRUCTION_TYPE_RET	**Value:** 43
INSTRUCTION_TYPE_ROX	**Value:** 23
INSTRUCTION_TYPE_SBB	**Value:** 8
INSTRUCTION_TYPE_SCAS	**Value:** 17
INSTRUCTION_TYPE_SETC	**Value:** 39
INSTRUCTION_TYPE_SGDT	**Value:** 53
INSTRUCTION_TYPE_SHX	**Value:** 22
INSTRUCTION_TYPE_SIDT	**Value:** 54
INSTRUCTION_TYPE_SLDT	**Value:** 55
INSTRUCTION_TYPE_SSE	**Value:** 105
INSTRUCTION_TYPE_STD	**Value:** 58
INSTRUCTION_TYPE_STOS	**Value:** 15
INSTRUCTION_TYPE_SUB	**Value:** 7
INSTRUCTION_TYPE_SYSENTER	**Value:** 101
INSTRUCTION_TYPE_TEST	**Value:** 31

continued on next page

Name	Description
INSTRUCTION_TYPE_XADD	**Value:** 5
INSTRUCTION_TYPE_XCHG	**Value:** 29
INSTRUCTION_TYPE_XLAT	**Value:** 59
INSTRUCTION_TYPE_XOR	**Value:** 27
MODE_16	**Value:** 1
MODE_32	**Value:** 0
OPERAND_TYPE_IMMEDIATE	**Value:** 3
OPERAND_TYPE_MEMORY	**Value:** 1
OPERAND_TYPE_NONE	**Value:** 0
OPERAND_TYPE_REGISTER	**Value:** 2
REGISTER_EAX	**Value:** 0
REGISTER_EBP	**Value:** 5
REGISTER_EBX	**Value:** 3
REGISTER_ECX	**Value:** 1
REGISTER_EDI	**Value:** 7
REGISTER_EDX	**Value:** 2
REGISTER_ESI	**Value:** 6
REGISTER_ESP	**Value:** 4
REGISTER_NOP	**Value:** 8
REGISTER_TYPE_CONTROL	**Value:** 4
REGISTER_TYPE_DEBUG	**Value:** 3
REGISTER_TYPE_FPU	**Value:** 8
REGISTER_TYPE_GEN	**Value:** 1
REGISTER_TYPE_MMX	**Value:** 7
REGISTER_TYPE_SEGMENT	**Value:** 2
REGISTER_TYPE_TEST	**Value:** 5
REGISTER_TYPE_XMM	**Value:** 6
__package__	**Value:** None

46 Module morpher.pydbg.pydbg

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

46.1 Variables

Name	Description
ntdll	**Value:** <WinDLL 'ntdll', handle 77a40000 at 23d6e50>
iphlpapi	**Value:** <WinDLL 'iphlpapi', handle 73820000 at 23dbbb0>
kernel32	**Value:** <WinDLL 'kernel32', handle 76a70000 at 249fc70>
advapi32	**Value:** <WinDLL 'advapi32', handle 77460000 at 23d6570>
AF_INET	**Value:** 2
AF_INET6	**Value:** 23
CONTEXT_CONTROL	**Value:** 65537
CONTEXT_DEBUG_REGISTERS	**Value:** 65552
CONTEXT_FULL	**Value:** 65543
CREATE_NEW_CONSOLE	**Value:** 16
CREATE_PROCESS_DEBUG_EVENT	**Value:** 3
CREATE_THREAD_DEBUG_EVENT	**Value:** 2
DBG_CONTINUE	**Value:** 65538
DBG_EXCEPTION_HANDLED	**Value:** 65537
DBG_EXCEPTION_NOT_HANDLED	**Value:** 2147549185
DEBUG_ONLY_THIS_PROCESS	**Value:** 2
DEBUG_PROCESS	**Value:** 1
DEFAULT_MODE	**Value:** 0
EFLAGS_RF	**Value:** 65536
EFLAGS_TRAP	**Value:** 256

continued on next page

Name	Description
ERROR_NO_MORE_FILES	**Value:** 18
EXCEPTION_ACCESS_VIOLATION	**Value:** 3221225477
EXCEPTION_BREAKPOINT	**Value:** 2147483651
EXCEPTION_DEBUG_EVENT	**Value:** 1
EXCEPTION_GUARD_PAGE	**Value:** 2147483649
EXCEPTION_SINGLE_STEP	**Value:** 2147483652
EXIT_PROCESS_DEBUG_EVENT	**Value:** 5
EXIT_THREAD_DEBUG_EVENT	**Value:** 4
FILE_MAP_READ	**Value:** 4
FORMAT_MESSAGE_ALLOCATE_BUFFER	**Value:** 256
FORMAT_MESSAGE_FROM_SYSTEM	**Value:** 4096
GetLastError	**Value:** <_FuncPtr object at 0x0253AA08>
HW_ACCESS	**Value:** 3
HW_EXECUTE	**Value:** 0
HW_WRITE	**Value:** 1
INVALID_HANDLE_VALUE	**Value:** 4294967295
LOAD_DLL_DEBUG_EVENT	**Value:** 6
MEM_COMMIT	**Value:** 4096
MEM_DECOMMIT	**Value:** 16384
MEM_IMAGE	**Value:** 16777216
MEM_RELEASE	**Value:** 32768
MIB_TCP_STATE_LISTEN	**Value:** 2
OUTPUT_DEBUG_STRING_EVENT	**Value:** 8
PAGE_EXECUTE	**Value:** 16
PAGE_EXECUTE_READ	**Value:** 32
PAGE_EXECUTE_READWRITE	**Value:** 64

continued on next page

Name	Description
PAGE_EXECUTE_WRITECOPY	**Value:** 128
PAGE_GUARD	**Value:** 256
PAGE_NOACCESS	**Value:** 1
PAGE_NOCACHE	**Value:** 512
PAGE_READONLY	**Value:** 2
PAGE_READWRITE	**Value:** 4
PAGE_WRITECOMBINE	**Value:** 1024
PAGE_WRITECOPY	**Value:** 8
PROCESS_ALL_ACCESS	**Value:** 2035711
RIP_EVENT	**Value:** 9
RTLD_GLOBAL	**Value:** 0
RTLD_LOCAL	**Value:** 0
SE_PRIVILEGE_ENABLED	**Value:** 2
SW_SHOW	**Value:** 5
SysDbgReadMsr	**Value:** 16
SysDbgWriteMsr	**Value:** 17
TCP_TABLE_OWNER_PID_ALL	**Value:** 5
TH32CS_INHERIT	**Value:** 2147483648
TH32CS_SNAPALL	**Value:** 15
TH32CS_SNAPHEAPLIST	**Value:** 1
TH32CS_SNAPMODULE	**Value:** 8
TH32CS_SNAPPROCESS	**Value:** 2
TH32CS_SNAPTHREAD	**Value:** 4
THREAD_ALL_ACCESS	**Value:** 2032639
TOKEN_ADJUST_PRIVILEGES	**Value:** 32
UDP_TABLE_OWNER_PID	**Value:** 1
UNLOAD_DLL_DEBUG_EVENT	**Value:** 7
USER_CALLBACK_DEBUG_EVENT	**Value:** 3735928559
VIRTUAL_MEM	**Value:** 12288
__package__	**Value:** 'morpher.pydbg'
c_types	**Value:** (<type '_ctypes.Structure'>, <class 'ctypes.c_char'>, <cl...
cdll	**Value:** <ctypes.LibraryLoader object at 0x0249FB90>

continued on next page

Name	Description
memmove	**Value:** <CFunctionType object at 0x0253AA80>
memset	**Value:** <CFunctionType object at 0x0253AAF8>
oledll	**Value:** <ctypes.LibraryLoader object at 0x0249FC30>
psapi	**Value:** <WinDLL 'psapi', handle 76cb0000 at 23ed450>
pydll	**Value:** <ctypes.LibraryLoader object at 0x0249FBB0>
pythonapi	**Value:** <PyDLL 'python dll', handle 1e000000 at 249fbd0>
windll	**Value:** <ctypes.LibraryLoader object at 0x0249FBF0>

46.2 Class pydbg

This class implements standard low leven functionality including:

- The load() / attach() routines.
- The main debug event loop.
- Convenience wrappers for commonly used Windows API.
- Single step toggling routine.
- Win32 error handler wrapped around PDX.
- Base exception / event handler routines which are meant to be overridden.

Higher level functionality is also implemented including:

- Register manipulation.
- Soft (INT 3) breakpoints.
- Memory breakpoints (page permissions).
- Hardware breakpoints.
- Exception / event handling call backs.
- Pydasm (libdasm) disassembly wrapper.
- Process memory snapshotting and restoring.
- Endian manipulation routines.
- Debugger hiding.
- Function resolution.
- "Intelligent" memory derefencing.

- Stack/SEH unwinding.
- Etc...

46.2.1 Methods

__init__(*self*, *ff*=`True`, *cs*=`False`)

Set the default attributes. See the source if you want to modify the default creation values.

Parameters

`ff`: (Optional, Def=True) Flag controlling whether or not pydbg attaches to forked processes

(type=Boolean)

`cs`: (Optional, Def=False) Flag controlling whether or not pydbg is in client/server (socket) mode

(type=Boolean)

addr_to_dll(*self*, *address*)

Return the system DLL that contains the address specified.

Parameters

`address`: Address to search system DLL ranges for

(type=DWORD)

Return Value

System DLL that contains the address specified or None if not found.

(type=system_dll)

addr_to_module(*self*, *address*)

Return the MODULEENTRY32 structure for the module that contains the address specified.

Parameters

`address`: Address to search loaded module ranges for

(type=DWORD)

Return Value

MODULEENTRY32 strucutre that contains the address specified or None if not found.

(type=MODULEENTRY32)

attach(*self*, *pid*)

Attach to the specified process by PID. Saves a process handle in self.h_process and prevents debuggee from exiting on debugger quit.

Parameters

`pid`: Process ID to attach to

(type=Integer)

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

bp_del(*self*, *address*)

Removes the breakpoint from target address.

Parameters

`address`: Address or list of addresses to remove breakpoint from

(type=DWORD or List)

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

See Also: bp_set(), bp_del_all(), bp_is_ours()

bp_del_all(*self*)

Removes all breakpoints from the debuggee.

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

See Also: bp_set(), bp_del(), bp_is_ours()

bp_del_hw(*self*, *address*=`None`, *slot*=`None`)

Removes the hardware breakpoint from the specified address or slot. Either an address or a slot must be specified, but not both.

Parameters

`address`: (Optional) Address to remove hardware breakpoint from.

(type=DWORD)

`slot`: (Optional)

(type=Integer (0 through 3))

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

See Also: bp_set_hw(), bp_del_hw_all()

bp_del_hw_all(*self*)

Removes all hardware breakpoints from the debuggee.

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

See Also: bp_set_hw(), bp_del_hw()

bp_del_mem(*self*, *address*)

Removes the memory breakpoint from target address.

Parameters

`address`: Address or list of addresses to remove memory breakpoint from

(type=DWORD)

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

See Also: bp_del_mem_all(), bp_set_mem(), bp_is_ours_mem()

bp_del_mem_all(*self*)

Removes all memory breakpoints from the debuggee.

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

See Also: bp_del_mem(), bp_set_mem(), bp_is_ours_mem()

bp_is_ours(*self*, *address_to_check*)

Determine if a breakpoint address belongs to us.

Parameters

`address_to_check`: Address to check if we have set a breakpoint at

(type=DWORD)

Return Value

True if breakpoint in question is ours, False otherwise

(type=Bool)

See Also: bp_set(), bp_del(), bp_del_all()

bp_is_ours_mem(*self*, *address_to_check*)

Determines if the specified address falls within the range of one of our memory breakpoints. When handling potential memory breakpoint exceptions it is mandatory to check the offending address with this routine as memory breakpoints are implemented by changing page permissions and the referenced address may very well exist within the same page as a memory breakpoint but not within the actual range of the buffer we wish to break on.

Parameters

`address_to_check:` Address to check if we have set a breakpoint on

(type=DWORD)

Return Value

The starting address of the buffer our breakpoint triggered on or False if address falls outside range.

(type=Mixed)

See Also: bp_set_mem(), bp_del_mem(), bp_del_mem_all()

bp_set(*self*, *address*, *description*=’’, *restore*=`True`, *handler*=`None`)

Sets a breakpoint at the designated address. Register an EXCEPTION_BREAKPOINT callback handler to catch breakpoint events. If a list of addresses is submitted to this routine then the entire list of new breakpoints get the same description and restore. The optional ”handler” parameter can be used to identify a function to specifically handle the specified bp, as opposed to the generic bp callback handler. The prototype of the callback routines is:

```
func (pydbg)
    return DBG_CONTINUE     # or other continue status
```

Parameters

`address:` Address or list of addresses to set breakpoint at

(type=DWORD or List)

`description:` (Optional) Description to associate with this breakpoint

(type=String)

`restore:` (Optional, def=True) Flag controlling whether or not to restore the breakpoint

(type=Bool)

`handler:` (Optional, def=None) Optional handler to call for this bp instead of the default handler

(type=Function Pointer)

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

See Also: bp_is_ours(), bp_del(), bp_del_all()

bp_set_hw(*self*, *address*, *length*, *condition*, *description*='', *restore*=True, *handler*=None)

Sets a hardware breakpoint at the designated address. Register an EXCEPTION_SINGLE_STEP callback handler to catch hardware breakpoint events. Setting hardware breakpoints requires the internal h_thread handle be set. This means that you can not set one outside the context of an debug event handler. If you want to set a hardware breakpoint as soon as you attach to or load a process, do so in the first chance breakpoint handler.

For more information regarding the Intel x86 debug registers and hardware breakpoints see:

```
http://pdos.csail.mit.edu/6.828/2005/readings/ia32/IA32-3.pdf
Section 15.2
```

Alternatively, you can register a custom handler to handle hits on the specific hw breakpoint slot.

*Warning: Setting hardware breakpoints during the first system breakpoint will be removed upon process continue. A better approach is to set a software breakpoint that when hit will set your hardware breakpoints.

Parameters

`address:` Address to set hardware breakpoint at

(type=DWORD)

`length:` Size of hardware breakpoint in bytes (byte, word or dword)

(type=Integer (1, 2 or 4))

`condition:` Condition to set the hardware breakpoint to activate on

(type=Integer (HW_ACCESS, HW_WRITE, HW_EXECUTE))

`description:` (Optional) Description of breakpoint

(type=String)

`restore:` (Optional, def=True) Flag controlling whether or not to restore the breakpoint

(type=Boolean)

`handler:` (Optional, def=None) Optional handler to call for this bp instead of the default handler

(type=Function Pointer)

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

Note: Hardware breakpoints are handled globally throughout the entire

bp_set_mem(*self*, *address*, *size*, *description*='', *handler*=None)

Sets a memory breakpoint at the target address. This is implemented by changing the permissions of the page containing the address to PAGE_GUARD. To catch memory breakpoints you have to register the EXCEPTION_GUARD_PAGE callback. Within the callback handler check the internal pydbg variable self.memory_breakpoint_hit to determine if the violation was a result of a direct memory breakpoint hit or some unrelated event. Alternatively, you can register a custom handler to handle the memory breakpoint. Memory breakpoints are automatically restored via the internal single step handler. To remove a memory breakpoint, you must explicitly call bp_del_mem().

Parameters

`address:` Starting address of the buffer to break on

(type=DWORD)

`size:` Size of the buffer to break on

(type=Integer)

`description:` (Optional) Description to associate with this breakpoint

(type=String)

`handler:` (Optional, def=None) Optional handler to call for this bp instead of the default handler

(type=Function Pointer)

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

See Also: bp_is_ours_mem(), bp_del_mem(), bp_del_mem_all()

close_handle(*self*, *handle*)

Convenience wraper around kernel32.CloseHandle()

Parameters

`handle:` Handle to close

(type=Handle)

Return Value

Return value from CloseHandle().

(type=Bool)

dbg_print_all_debug_registers(*self*)

*** DEBUG ROUTINE ***

This is a debugging routine that was used when debugging hardware breakpoints. It was too useful to be removed from the release code.

dbg_print_all_guarded_pages(*self*)

*** DEBUG ROUTINE ***

This is a debugging routine that was used when debugging memory breakpoints. It was too useful to be removed from the release code.

debug_active_process(*self*, *pid*)

Convenience wrapper around GetLastError() and FormatMessage(). Returns the error code and formatted message associated with the last error. You probably do not want to call this directly, rather look at attach().

Parameters

`pid`: Process ID to attach to

(type=Integer)

Raises

`pdx` An exception is raised on failure.

debug_event_iteration(*self*)

Check for and process a debug event.

debug_event_loop(*self*)

Enter the infinite debug event handling loop. This is the main loop of the debugger and is responsible for catching debug events and exceptions and dispatching them appropriately. This routine will check for and call the USER_CALLBACK_DEBUG_EVENT callback on each loop iteration. run() is an alias for this routine.

Raises

`pdx` An exception is raised on any exceptional conditions, such as debugger being interrupted or debuggee quiting.

See Also: run()

debug_set_process_kill_on_exit(*self*, *kill_on_exit*)

Convenience wrapper around DebugSetProcessKillOnExit().

Parameters

`kill_on_exit:` True to kill the process on debugger exit, False to let debuggee continue running.

(type=Bool)

Raises

`pdx` An exception is raised on failure.

detach(*self*)

Detach from debuggee.

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

disasm(*self*, *address*)

Pydasm disassemble utility function wrapper. Stores the pydasm decoded instruction in self.instruction.

Parameters

`address:` Address to disassemble at

(type=DWORD)

Return Value

Disassembled string.

(type=String)

disasm_around(*self*, *address*, *num_inst*=5)

Given a specified address this routine will return the list of 5 instructions before and after the instruction at address (including the instruction at address, so 11 instructions in total). This is accomplished by grabbing a larger chunk of data around the address than what is predicted as necessary and then disassembling forward. If during the forward disassembly the requested address lines up with the start of an instruction, then the assumption is made that the forward disassembly self corrected itself and the instruction set is returned. If we are unable to align with the original address, then we modify our data slice and try again until we do.

Parameters

`address:` Address to disassemble around

(type=DWORD)

`num_inst:` (Optional, Def=5) Number of instructions to disassemble up/down from address

(type=Integer)

Return Value

List of tuples (address, disassembly) of instructions around the specified address.

(type=List)

dump_context(*self*, *context*=`None`, *stack_depth*=`5`, *print_dots*=`True`)

Return an informational block of text describing the CPU context of the current thread. Information includes:

- Disassembly at current EIP
- Register values in hex, decimal and "smart" dereferenced
- ESP, ESP+4, ESP+8 ... values in hex, decimal and "smart" dereferenced

Parameters

`context`: (Optional) Current thread context to examine

(type=Context)

`stack_depth`: (Optional, def:5) Number of dwords to dereference off of the stack (not including ESP)

(type=Integer)

`print_dots`: (Optional, def:True) Controls suppression of dot in place of non-printable

(type=Bool)

Return Value

Information about current thread context.

(type=String)

See Also: dump_context_list()

dump_context_list(*self*, *context*=`None`, *stack_depth*=`5`, *print_dots*=`True`, *hex_dump*=`False`)

Return an informational list of items describing the CPU context of the current thread. Information includes:

- Disassembly at current EIP
- Register values in hex, decimal and "smart" dereferenced
- ESP, ESP+4, ESP+8 ... values in hex, decimal and "smart" dereferenced

Parameters

`context:` (Optional) Current thread context to examine

(type=Context)

`stack_depth:` (Optional, def:5) Number of dwords to dereference off of the stack (not including ESP)

(type=Integer)

`print_dots:` (Optional, def:True) Controls suppression of dot in place of non-printable

(type=Bool)

`hex_dump:` (Optional, def=False) Return a hex dump in the absense of string detection

(type=Bool)

Return Value

Dictionary of information about current thread context.

(type=Dictionary)

See Also: dump_context()

enumerate_modules(*self*)

Using the CreateToolhelp32Snapshot() API enumerate and return the list of module name / base address tuples that belong to the debuggee

Return Value

List of module name / base address tuples.

(type=List)

See Also: iterate_modules()

enumerate_processes(*self*)

Using the CreateToolhelp32Snapshot() API enumerate all system processes returning a list of pid / process name tuples.

Return Value

List of pid / process name tuples.

Example:

```
for (pid, name) in pydbg.enumerate_processes():
    if name == "test.exe":
        break

pydbg.attach(pid)
```

(type=List)

See Also: iterate_processes()

enumerate_threads(*self*)

Using the CreateToolhelp32Snapshot() API enumerate all system threads returning a list of thread IDs that belong to the debuggee.

Return Value

List of thread IDs belonging to the debuggee.

Example:

```
for thread_id in self.enumerate_threads():
    context = self.get_thread_context(None, thread_id)
```

(type=List)

See Also: iterate_threads()

event_handler_create_process(*self*)

This is the default CREATE_PROCESS_DEBUG_EVENT handler.

Return Value

Debug event continue status.

(type=DWORD)

event_handler_create_thread(*self*)

This is the default CREATE_THREAD_DEBUG_EVENT handler.

Return Value

Debug event continue status.

(type=DWORD)

event_handler_exit_process(*self*)

This is the default EXIT_PROCESS_DEBUG_EVENT handler.

Raises

`pdx` An exception is raised to denote process exit.

event_handler_exit_thread(*self*)

This is the default EXIT_THREAD_DEBUG_EVENT handler.

Return Value

Debug event continue status.

(type=DWORD)

event_handler_load_dll(*self*)

This is the default LOAD_DLL_DEBUG_EVENT handler. You can access the last loaded dll in your callback handler with the following example code:

```
last_dll = pydbg.get_system_dll(-1)
print "loading:%s from %s into:%08x size:%d" % (last_dll.name, last_dll.path, las
```

The get_system_dll() routine is preferred over directly accessing the internal data structure for proper and transparent client/server support.

Return Value

Debug event continue status.

(type=DWORD)

event_handler_unload_dll(*self*)

This is the default UNLOAD_DLL_DEBUG_EVENT handler.

Return Value

Debug event continue status.

(type=DWORD)

exception_handler_access_violation(*self*)

This is the default EXCEPTION_ACCESS_VIOLATION handler. Responsible for handling the access violation and passing control to the registered user callback handler.

Return Value

Debug event continue status.

(type=DWORD)

Attention: If you catch an access violaton and wish to terminate the process, you *must* still return DBG_CONTINUE to avoid a deadlock.

exception_handler_breakpoint(*self*)

This is the default EXCEPTION_BREAKPOINT handler, responsible for transparently restoring soft breakpoints and passing control to the registered user callback handler.

Return Value

Debug event continue status.

(type=DWORD)

exception_handler_guard_page(*self*)

This is the default EXCEPTION_GUARD_PAGE handler, responsible for transparently restoring memory breakpoints passing control to the registered user callback handler.

Return Value

Debug event continue status.

(type=DWORD)

exception_handler_single_step(*self*)

This is the default EXCEPTION_SINGLE_STEP handler, responsible for transparently restoring breakpoints and passing control to the registered user callback handler.

Return Value

Debug event continue status.

(type=DWORD)

func_resolve(*self*, *dll*, *function*)

Utility function that resolves the address of a given module / function name pair under the context of the debugger.

Parameters

`dll:` Name of the DLL (case-insensitive)

(type=String)

`function:` Name of the function to resolve (case-sensitive)

(type=String)

Return Value

Address

(type=DWORD)

See Also: func_resolve_debuggee()

func_resolve_debuggee(*self*, *dll_name*, *func_name*)

Utility function that resolves the address of a given module / function name pair under the context of the debuggee. Note: Be weary of calling this function from within a LOAD_DLL handler as the module is not yet fully loaded and therefore the snapshot will not include it.

Parameters

`dll_name:` Name of the DLL (case-insensitive, ex:ws2_32.dll)

(type=String)

`func_name:` Name of the function to resolve (case-sensitive)

(type=String)

Return Value

Address of the symbol in the target process address space if it can be resolved, None otherwise

(type=DWORD)

Author: Otto Ebeling

See Also: func_resolve()

To Do: Add support for followed imports.

get_ascii_string(*self*, *data*)

Retrieve the ASCII string, if any, from data. Ensure that the string is valid by checking against the minimum length requirement defined in self.STRING_EXPLORATION_MIN_LENGTH.

Parameters

`data`: Data to explore for printable ascii string

(type=Raw)

Return Value

False on failure, ascii string on discovered string.

(type=String)

get_arg(*self*, *index*, *context*=`None`)

Given a thread context, this convenience routine will retrieve the function argument at the specified index. The return address of the function can be retrieved by specifying an index of 0. This routine should be called from breakpoint handlers at the top of a function.

Parameters

`index`: Data to explore for printable ascii string

(type=Integer)

`context`: (Optional) Current thread context to examine

(type=Context)

Return Value

Value of specified argument.

(type=DWORD)

get_attr(*self*, *attribute*)

Return the value for the specified class attribute. This routine should be used over directly accessing class member variables for transparent support across local vs. client/server debugger clients.

Parameters

`attribute`: Name of attribute to return.

(type=String)

Return Value

Requested attribute or None if not found.

(type=Mixed)

See Also: set_attr()

get_debug_privileges(*self*)

Obtain necessary privileges for debugging.

Raises

`pdx` An exception is raised on failure.

get_instruction(*self*, *address*)

Pydasm disassemble utility function wrapper. Returns the pydasm decoded instruction in self.instruction.

Parameters

`address`: Address to disassemble at

(type=DWORD)

Return Value

pydasm instruction

(type=pydasm instruction)

get_printable_string(*self*, *data*, *print_dots*=`True`)

description

Parameters

`data`: Data to explore for printable ascii string

(type=Raw)

`print_dots`: (Optional, def:True) Controls suppression of dot in place of non-printable

(type=Bool)

Return Value

False on failure, discovered printable chars in string otherwise.

(type=String)

get_register(*self*, *register*)

Get the value of a register in the debuggee within the context of the self.h_thread.

Parameters

`register:` One of EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP, EIP

(type=Register)

Return Value

Value of specified register.

(type=DWORD)

Raises

`pdx` An exception is raised on failure.

get_system_dll(*self*, *idx*)

Return the system DLL at the specified index. If the debugger is in client / server mode, remove the PE structure (we do not want to send that mammoth over the wire).

Parameters

`idx:` Index into self.system_dlls[] to retrieve DLL from.

(type=Integer)

Return Value

Requested attribute or None if not found.

(type=Mixed)

get_thread_context(*self*, *thread_handle*=`None`, *thread_id*=`0`)

Convenience wrapper around GetThreadContext(). Can obtain a thread context via a handle or thread id.

Parameters

`thread_handle:` (Optional) Handle of thread to get context of
(type=HANDLE)

`thread_id:` (Optional) ID of thread to get context of
(type=Integer)

Return Value

Thread CONTEXT on success.
(type=CONTEXT)

Raises

`pdx` An exception is raised on failure.

get_unicode_string(*self*, *data*)

description

Parameters

`data:` Data to explore for printable unicode string
(type=Raw)

Return Value

False on failure, ascii-converted unicode string on discovered string.
(type=String)

hex_dump(*self*, *data*, *addr*=`0`, *prefix*=`''`)

Utility function that converts data into hex dump format.

Parameters

`data:` Raw bytes to view in hex dump
(type=Raw Bytes)

`addr:` (Optional, def=0) Address to start hex offset display from
(type=DWORD)

`prefix:` String to prefix each line of hex dump with.
(type=String (Optional, def=""))

Return Value

Hex dump of data.
(type=String)

hide_debugger(*self*)

Hide the presence of the debugger. This routine requires an active context and therefore can not be called immediately after a load() for example. Call it from the first chance breakpoint handler. This routine hides the debugger in the following ways:

- Modifies the PEB flag that IsDebuggerPresent() checks for.

Raises

`pdx` An exception is raised if we are unable to hide the debugger for various reasons.

is_address_on_stack(*self*, *address*, *context*=`None`)

Utility function to determine if the specified address exists on the current thread stack or not.

Parameters

`address`: Address to check

(type=DWORD)

`context`: (Optional) Current thread context to examine

(type=Context)

Return Value

True if address lies in current threads stack range, False otherwise.

(type=Bool)

iterate_modules(*self*)

A simple iterator function that can be used to iterate through all modules the target process has mapped in its address space. Yielded objects are of type MODULEENTRY32.

Return Value

Iterated module entries.

(type=MODULEENTRY32)

Author: Otto Ebeling

See Also: enumerate_modules()

Warning: break-ing out of loops over this routine will cause a handle leak.

iterate_processes(*self*)

A simple iterator function that can be used to iterate through all running processes. Yielded objects are of type PROCESSENTRY32.

Return Value

Iterated process entries.

(type=PROCESSENTRY32)

See Also: enumerate_processes()

Warning: break-ing out of loops over this routine will cause a handle leak.

iterate_threads(*self*)

A simple iterator function that can be used to iterate through all running processes. Yielded objects are of type THREADENTRY32.

Return Value

Iterated process entries.

(type=THREADENTRY32)

See Also: enumerate_threads()

Warning: break-ing out of loops over this routine will cause a handle leak.

flip_endian(*self*, *dword*)

Utility function to flip the endianess a given DWORD into raw bytes.

Parameters

`dword`: DWORD whose endianess to flip

(type=DWORD)

Return Value

Converted DWORD in raw bytes.

(type=Raw Bytes)

flip_endian_dword(*self*, *bytes*)

Utility function to flip the endianess of a given set of raw bytes into a DWORD.

Parameters

`bytes`: Raw bytes whose endianess to flip

(type=Raw Bytes)

Return Value

Converted DWORD.

(type=DWORD)

load(*self*, *path_to_file*, *command_line*=`None`, *create_new_console*=`False`, *show_window*=`True`)

Load the specified executable and optional command line arguments into the debugger.

Parameters

`path_to_file`:	Full path to executable to load in debugger *(type=String)*
`command_line`:	(Optional, def=None) Command line arguments to pass to debuggee *(type=String)*
`create_new_console`:	(Optional, def=False) Create a new console for the debuggee. *(type=Boolean)*
`show_window`:	(Optional, def=True) Show / hide the debuggee window. *(type=Boolean)*

Raises

`pdx` An exception is raised if we are unable to load the specified executable in the debugger.

To Do: This routines needs to be further tested ... I nomally just attach.

open_process(*self*, *pid*)

Convenience wrapper around OpenProcess().

Parameters

`pid`: Process ID to attach to

(type=Integer)

Raises

`pdx` An exception is raised on failure.

open_thread(*self*, *thread_id*)

Convenience wrapper around OpenThread().

Parameters

`thread_id`: ID of thread to obtain handle to

(type=Integer)

Raises

`pdx` An exception is raised on failure.

page_guard_clear(*self*)

Clear all debugger-set PAGE_GUARDs from memory. This is useful for suspending memory breakpoints to single step past a REP instruction.

Return Value

Self

(type=pydbg)

See Also: page_guard_restore()

page_guard_restore(*self*)

Restore all previously cleared debugger-set PAGE_GUARDs from memory. This is useful for suspending memory breakpoints to single step past a REP instruction.

Return Value

Self

(type=pydbg)

See Also: page_guard_clear()

pid_to_port(*self*, *pid*)

A helper function that enumerates the IPv4 endpoints for a given process ID.

Parameters

`pid`: Process ID to find port information on.

(type=Integer)

Return Value

A list of the protocol, bound address and listening port

(type=A list of tuples)

Raises

`pdx` An exception is raised on failure

Author: Justin Seitz

process_restore(*self*)

Restore memory / context snapshot of the debuggee. All threads must be suspended before calling this routine.

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

process_snapshot(*self*, *mem_only*=`False`)

Take memory / context snapshot of the debuggee. All threads must be suspended before calling this routine.

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

read(*self*, *address*, *length*)

Alias to read_process_memory().

See Also: read_process_memory

read_msr(*self*, *address*)

Read data from the specified MSR address.

Parameters

`address`: MSR address to read from.

(type=DWORD)

Return Value

(read status, msr structure)

(type=tuple)

See Also: write_msr

read_process_memory(*self*, *address*, *length*)

Read from the debuggee process space.

Parameters

`address`: Address to read from.

(type=DWORD)

`length`: Length, in bytes, of data to read.

(type=Integer)

Return Value

Read data.

(type=Raw)

Raises

`pdx` An exception is raised on failure.

resume_all_threads(*self*)

Resume all process threads.

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

See Also: suspend_all_threads()

resume_thread(*self*, *thread_id*)

Resume the specified thread.

Parameters

`thread_id`: ID of thread to resume.

(type=DWORD)

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

ret_self(*self*)

This convenience routine exists for internal functions to call and transparently return the correct version of self. Specifically, an object in normal mode and a moniker when in client/server mode.

Return Value

Client / server safe version of self

run(*self*)

Alias for debug_event_loop().

See Also: debug_event_loop()

seh_unwind(*self*, *context*=`None`)

Unwind the the Structured Exception Handler (SEH) chain of the current or specified thread to the best of our abilities. The SEH chain is a simple singly linked list, the head of which is pointed to by fs:0. In cases where the SEH chain is corrupted and the handler address points to invalid memory, it will be returned as 0xFFFFFFFF.

Parameters

`context`: (Optional) Current thread context to examine

(type=Context)

Return Value

Naturally ordered list of SEH addresses and handlers.

(type=List of Tuples)

set_attr(*self*, *attribute*, *value*)

Return the value for the specified class attribute. This routine should be used over directly accessing class member variables for transparent support across local vs. client/server debugger clients.

Parameters

`attribute:` Name of attribute to return.

(type=String)

`value:` Value to set attribute to.

(type=Mixed)

See Also: set_attr()

set_callback(*self*, *exception_code*, *callback_func*)

Set a callback for the specified exception (or debug event) code. The prototype of the callback routines is:

```
func (pydbg):
    return DBG_CONTINUE     # or other continue status
```

You can register callbacks for any exception code or debug event. Look in the source for all event_handler_??? and exception_handler_??? routines to see which ones have internal processing (internal handlers will still pass control to your callback). You can also register a user specified callback that is called on each loop iteration from within debug_event_loop(). The callback code is USER_CALLBACK_DEBUG_EVENT and the function prototype is:

```
func (pydbg)
    return DBG_CONTINUE     # or other continue status
```

User callbacks do not / should not access debugger or contextual information.

Parameters

`exception_code:` Exception code to establish a callback for

(type=Long)

`callback_func:` Function to call when specified exception code is caught.

(type=Function)

set_debugger_active(*self*, *enable*)

Enable or disable the control flag for the main debug event loop. This is a convenience shortcut over set_attr.

Parameters

`enable`: Flag controlling the main debug event loop.

(type=Boolean)

set_register(*self*, *register*, *value*)

Set the value of a register in the debuggee within the context of the self.h_thread.

Parameters

`register`: One of EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP, EIP

(type=Register)

`value`: Value to set register to

(type=DWORD)

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

set_thread_context(*self*, *context*, *thread_handle*=`None`, *thread_id*=`0`)

Convenience wrapper around SetThreadContext(). Can set a thread context via a handle or thread id.

Parameters

`thread_handle:` (Optional) Handle of thread to get context of
(type=HANDLE)

`context:` Context to apply to specified thread
(type=CONTEXT)

`thread_id:` (Optional, Def=0) ID of thread to get context of
(type=Integer)

Return Value

Self
(type=pydbg)

Raises

`pdx` An exception is raised on failure.

sigint_handler(*self*, *signal_number*, *stack_frame*)

Interrupt signal handler. We override the default handler to disable the run flag and exit the main debug event loop.

Parameters

`signal_number:` *(type=)*

`stack_frame:` *(type=)*

single_step(*self*, *enable*, *thread_handle*=`None`)

Enable or disable single stepping in the specified thread or self.h_thread if a thread handle is not specified.

Parameters

`enable:` True to enable single stepping, False to disable

(type=Bool)

`thread_handle:` (Optional, Def=None) Handle of thread to put into single step mode

(type=Handle)

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

smart_dereference(*self*, *address*, *print_dots*=`True`, *hex_dump*=`False`)

"Intelligently" discover data behind an address. The address is dereferenced and explored in search of an ASCII or Unicode string. In the absense of a string the printable characters are returned with non-printables represented as dots (.). The location of the discovered data is returned as well as either "heap", "stack" or the name of the module it lies in (global data).

Parameters

`address:` Address to smart dereference

(type=DWORD)

`print_dots:` (Optional, def:True) Controls suppression of dot in place of non-printable

(type=Bool)

`hex_dump:` (Optional, def=False) Return a hex dump in the absense of string detection

(type=Bool)

Return Value

String of data discovered behind dereference.

(type=String)

stack_range(*self*, *context*=`None`)

Determine the stack range (top and bottom) of the current or specified thread. The desired information is located at offsets 4 and 8 from the Thread Environment Block (TEB), which in turn is pointed to by fs:0.

Parameters

`context`: (Optional) Current thread context to examine

(type=Context)

Return Value

List containing (stack_top, stack_bottom) on success, False otherwise.

(type=Mixed)

stack_unwind(*self*, *context*=`None`)

Unwind the stack to the best of our ability. This function is really only useful if called when EBP is actually used as a frame pointer. If it is otherwise being used as a general purpose register then stack unwinding will fail immediately.

Parameters

`context`: (Optional) Current thread context to examine

(type=Context)

Return Value

The current call stack ordered from most recent call backwards.

(type=List)

suspend_all_threads(*self*)

Suspend all process threads.

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

See Also: resume_all_threads()

suspend_thread(*self*, *thread_id*)

Suspend the specified thread.

Parameters

`thread_id:` ID of thread to suspend

(type=DWORD)

Return Value

Self

(type=pydbg)

Raises

`pdx` An exception is raised on failure.

terminate_process(*self*, *exit_code*=`0`, *method*=`'terminateprocess'`)

Terminate the debuggee using the specified method.

"terminateprocess": Terminate the debuggee by calling TerminateProcess(debuggee_handle). "exitprocess": Terminate the debuggee by setting its current EIP to ExitProcess().

Parameters

`exit_code:` (Optional, def=0) Exit code

(type=Integer)

`method:` (Optonal, def="terminateprocess") Termination method. See __doc__ for more info.

(type=String)

Raises

`pdx` An exception is raised on failure.

to_binary(*self*, *number*, *bit_count*=32)

Convert a number into a binary string. This is an ugly one liner that I ripped off of some site.

Parameters

`number`: Number to convert to binary string.

(type=Integer)

`bit_count`: (Optional, Def=32) Number of bits to include in output string.

(type=Integer)

Return Value

Specified integer as a binary string

(type=String)

See Also: to_decimal()

to_decimal(*self*, *binary*)

Convert a binary string into a decimal number.

Parameters

`binary`: Binary string to convert to decimal

(type=String)

Return Value

Specified binary string as an integer

(type=Integer)

See Also: to_binary()

virtual_alloc(*self*, *address*, *size*, *alloc_type*, *protection*)

Convenience wrapper around VirtualAllocEx()

Parameters

`address:` Desired starting address of region to allocate, can be None

(type=DWORD)

`size:` Size of memory region to allocate, in bytes

(type=Integer)

`alloc_type:` The type of memory allocation (most often MEM_COMMIT)

(type=DWORD)

`protection:` Memory protection to apply to the specified region

(type=DWORD)

Return Value

Base address of the allocated region of pages.

(type=DWORD)

Raises

`pdx` An exception is raised on failure.

virtual_free(*self*, *address*, *size*, *free_type*)

Convenience wrapper around VirtualFreeEx()

Parameters

`address:` Pointer to the starting address of the region of memory to be freed

(type=DWORD)

`size:` Size of memory region to free, in bytes

(type=Integer)

`free_type:` The type of free operation

(type=DWORD)

Raises

`pdx` An exception is raised on failure.

virtual_protect(*self*, *base_address*, *size*, *protection*)

Convenience wrapper around VirtualProtectEx()

Parameters

`base_address:` Base address of region of pages whose access protection attributes are to be changed

(type=DWORD)

`size:` Size of the region whose access protection attributes are to be changed

(type=Integer)

`protection:` Memory protection to apply to the specified region

(type=DWORD)

Return Value

Previous access protection.

(type=DWORD)

Raises

`pdx` An exception is raised on failure.

virtual_query(*self*, *address*)

Convenience wrapper around VirtualQueryEx().

Parameters

`address:` Address to query

(type=DWORD)

Return Value

MEMORY_BASIC_INFORMATION

(type=MEMORY_BASIC_INFORMATION)

Raises

`pdx` An exception is raised on failure.

win32_error(*self*, *prefix*=`None`)

Convenience wrapper around GetLastError() and FormatMessage(). Raises an exception with the relevant error code and formatted message.

Parameters

`prefix:` (Optional) String to prefix error message with.

(type=String)

Raises

`pdx` An exception is always raised by this routine.

write(*self*, *address*, *data*, *length*=0)

Alias to write_process_memory().

See Also: write_process_memory

write_msr(*self*, *address*, *data*)

Write data to the specified MSR address.

Parameters

`address`: MSR address to write to.

(type=DWORD)

`data`: Data to write to MSR address.

(type=QWORD)

Return Value

(read status, msr structure)

(type=tuple)

See Also: read_msr

write_process_memory(*self*, *address*, *data*, *length*=0)

Write to the debuggee process space. Convenience wrapper around WriteProcessMemory(). This routine will continuously attempt to write the data requested until it is complete.

Parameters

`address`: Address to write to

(type=DWORD)

`data`: Data to write

(type=Raw Bytes)

`length`: (Optional, Def:len(data)) Length of data, in bytes, to write

(type=DWORD)

Raises

`pdx` An exception is raised on failure.

46.2.2 Class Variables

Name	Description
STRING_EXPLORATON-_BUF_SIZE	**Value:** 256

continued on next page

Name	Description
STRING_EXPLORATIO-N_MIN_LENGTH	**Value:** 2

47 Module morpher.pydbg.pydbg_client

Author: Pedram Amini

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

47.1 Variables

Name	Description
AF_INET	**Value:** 2
AF_INET6	**Value:** 23
CONTEXT_CONTROL	**Value:** 65537
CONTEXT_DEBUG_REGISTERS	**Value:** 65552
CONTEXT_FULL	**Value:** 65543
CREATE_NEW_CONSOLE	**Value:** 16
CREATE_PROCESS_DEBUG_EVENT	**Value:** 3
CREATE_THREAD_DEBUG_EVENT	**Value:** 2
DBG_CONTINUE	**Value:** 65538
DBG_EXCEPTION_HANDLED	**Value:** 65537
DBG_EXCEPTION_NOT_HANDLED	**Value:** 2147549185
DEBUG_ONLY_THIS_PROCESS	**Value:** 2
DEBUG_PROCESS	**Value:** 1
DEFAULT_MODE	**Value:** 0
EFLAGS_RF	**Value:** 65536
EFLAGS_TRAP	**Value:** 256
ERROR_NO_MORE_FILES	**Value:** 18
EXCEPTION_ACCESS_VIOLATION	**Value:** 3221225477
EXCEPTION_BREAKPOINT	**Value:** 2147483651
EXCEPTION_DEBUG_EVENT	**Value:** 1
EXCEPTION_GUARD_PAGE	**Value:** 2147483649

continued on next page

Name	Description
EXCEPTION_SINGLE_STEP	**Value:** 2147483652
EXIT_PROCESS_DEBUG_EVENT	**Value:** 5
EXIT_THREAD_DEBUG_EVENT	**Value:** 4
FILE_MAP_READ	**Value:** 4
FORMAT_MESSAGE_ALLOCATE_BUFFER	**Value:** 256
FORMAT_MESSAGE_FROM_SYSTEM	**Value:** 4096
GetLastError	**Value:** <_FuncPtr object at 0x0253AA08>
HW_ACCESS	**Value:** 3
HW_EXECUTE	**Value:** 0
HW_WRITE	**Value:** 1
INVALID_HANDLE_VALUE	**Value:** 4294967295
LOAD_DLL_DEBUG_EVENT	**Value:** 6
MEM_COMMIT	**Value:** 4096
MEM_DECOMMIT	**Value:** 16384
MEM_IMAGE	**Value:** 16777216
MEM_RELEASE	**Value:** 32768
MIB_TCP_STATE_LISTEN	**Value:** 2
OUTPUT_DEBUG_STRING_EVENT	**Value:** 8
PAGE_EXECUTE	**Value:** 16
PAGE_EXECUTE_READ	**Value:** 32
PAGE_EXECUTE_READWRITE	**Value:** 64
PAGE_EXECUTE_WRITECOPY	**Value:** 128
PAGE_GUARD	**Value:** 256
PAGE_NOACCESS	**Value:** 1
PAGE_NOCACHE	**Value:** 512
PAGE_READONLY	**Value:** 2
PAGE_READWRITE	**Value:** 4
PAGE_WRITECOMBINE	**Value:** 1024
PAGE_WRITECOPY	**Value:** 8
PROCESS_ALL_ACCESS	**Value:** 2035711

continued on next page

Name	Description
RIP_EVENT	**Value:** 9
RTLD_GLOBAL	**Value:** 0
RTLD_LOCAL	**Value:** 0
SE_PRIVILEGE_ENABLED	**Value:** 2
SW_SHOW	**Value:** 5
SysDbgReadMsr	**Value:** 16
SysDbgWriteMsr	**Value:** 17
TCP_TABLE_OWNER_PID_ALL	**Value:** 5
TH32CS_INHERIT	**Value:** 2147483648
TH32CS_SNAPALL	**Value:** 15
TH32CS_SNAPHEAPLIST	**Value:** 1
TH32CS_SNAPMODULE	**Value:** 8
TH32CS_SNAPPROCESS	**Value:** 2
TH32CS_SNAPTHREAD	**Value:** 4
THREAD_ALL_ACCESS	**Value:** 2032639
TOKEN_ADJUST_PRIVILEGES	**Value:** 32
UDP_TABLE_OWNER_PID	**Value:** 1
UNLOAD_DLL_DEBUG_EVENT	**Value:** 7
USER_CALLBACK_DEBUG_EVENT	**Value:** 3735928559
VIRTUAL_MEM	**Value:** 12288
__package__	**Value:** 'morpher.pydbg'
advapi32	**Value:** <WinDLL 'advapi32', handle 77460000 at 23d6570>
c_types	**Value:** (<type '_ctypes.Structure'>, <class 'ctypes.c_char'>, <cl...
cdll	**Value:** <ctypes.LibraryLoader object at 0x0249FB90>
iphlpapi	**Value:** <WinDLL 'iphlpapi', handle 73820000 at 23dbbb0>
kernel32	**Value:** <WinDLL 'kernel32', handle 76a70000 at 249fc70>
memmove	**Value:** <CFunctionType object at 0x0253AA80>
memset	**Value:** <CFunctionType object at 0x0253AAF8>

continued on next page

Name	Description
ntdll	**Value:** <WinDLL 'ntdll', handle 77a40000 at 23d6e50>
oledll	**Value:** <ctypes.LibraryLoader object at 0x0249FC30>
psapi	**Value:** <WinDLL 'psapi', handle 76cb0000 at 23ed450>
pydll	**Value:** <ctypes.LibraryLoader object at 0x0249FBB0>
pythonapi	**Value:** <PyDLL 'python dll', handle 1e000000 at 249fbd0>
windll	**Value:** <ctypes.LibraryLoader object at 0x0249FBF0>

47.2 Class pydbg_client

This class defines the client portion of the decoupled client/server PyDBG debugger. The class was designed to be completely transparent to the end user, requiring only the simple change from:

```
dbg = pydbg()
```

to:

```
dbg = pydbg_client(host, port)
```

Command line options can be used to control which instantiation is used thereby allowing for any PyDBG driver to be used locally or remotely.

47.2.1 Methods

__init__(*self*, *host*, *port*)

Set the default client attributes. The target host and port are required.

Parameters

`host`: Host address of PyDBG server (dotted quad IP address or hostname)

(type=String)

`port`: Port that the PyDBG server is listening on.

(type=Integer)

Raises

`pdx` An exception is raised if a connection to the PyDbg server can not be established.

__getattr__(*self*, *method_name*)

This routine is called by default when a requested attribute (or method) is accessed that has no definition. Unfortunately __getattr__ only passes the requested method name and not the arguments. So we extend the functionality with a little lambda magic to the routine method_missing(). Which is actually how Ruby handles missing methods by default ... with arguments. Now we are just as cool as Ruby.

Parameters

`method_name`: The name of the requested and undefined attribute (or method in our case).

(type=String)

Return Value

Lambda magic passing control (and in turn the arguments we want) to self.method_missing().

(type=Lambda)

debug_event_loop(*self*)

Overriden debug event handling loop. A transparent mirror here with method_missing() would not do. Our debug event loop is reduced here to a data marshaling loop with the server. If the received type from the server is a tuple then we assume a debug or exception event has occured and pass it to any registered callbacks. The callback is free to call arbitrary PyDbg routines. Upon return of the callback, a special token, **DONE**, is used to flag to the PyDbg server that we are done processing the exception and it is free to move on.

method_missing(*self*, *method_name*, **args*, ***kwargs*)

See the notes for __getattr__ for related notes. This method is called, in the Ruby fashion, with the method name and arguments for any requested but undefined class method. We utilize this method to transparently wrap requested PyDBG routines, transmit the method name and arguments to the server, then grab and return the methods return value. This transparency allows us to modify pydbg.py freely without having to add support for newly created methods to pydbg_client.py. Methods that require "special" attention and can not simply be mirrored are individually overridden and handled separately.

Parameters

`method_name:` The name of the requested and undefined attribute (or method in our case).

(type=String)

`args:` Tuple of arguments. @type kwargs Dictionary

(type=Tuple)

`kwargs:` Dictioanry of arguments.

Return Value

Return value of the mirrored method.

(type=Mixed)

pickle_recv(*self*)

This routine is used for marshaling arbitrary data from the PyDbg server. We can send pretty much anything here. For example a tuple containing integers, strings, arbitrary objects and structures. Our "protocol" is a simple length-value protocol where each datagram is prefixed by a 4-byte length of the data to be received.

Return Value

Whatever is received over the socket.

(type=Mixed)

Raises

`pdx` An exception is raised if the connection was severed.

pickle_send(*self*, *data*)

This routine is used for marshaling arbitrary data to the PyDbg server. We can send pretty much anything here. For example a tuple containing integers, strings, arbitrary objects and structures. Our "protocol" is a simple length-value protocol where each datagram is prefixed by a 4-byte length of the data to be received.

Parameters

`data`: Data to marshal and transmit. Data can *pretty much* contain anything you throw at it.

(type=Mixed)

Raises

`pdx` An exception is raised if the connection was severed.

run(*self*)

Alias for debug_event_loop().

See Also: debug_event_loop()

set_callback(*self*, *exception_code*, *callback_func*)

Overriden callback setting routing. A transparent mirror here with method_missing() would not do. We save the requested callback address / exception code pair and then tell the PyDbg server about it. For more information see the documentation of pydbg.set_callback().

Parameters

`exception_code`: Exception code to establish a callback for

(type=Long)

`callback_func`: Function to call when specified exception code is caught.

(type=Function)

47.2.2 Class Variables

Name	Description
host	**Value:** `None`
port	**Value:** `None`
callbacks	**Value:** {}
pydbg	**Value:** `None`

48 Module morpher.pydbg.system_dll

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

48.1 Variables

Name	Description
kernel32	**Value:** <WinDLL 'kernel32', handle 76a70000 at 249fc70>
psapi	**Value:** <WinDLL 'psapi', handle 76cb0000 at 23ed450>
AF_INET	**Value:** 2
AF_INET6	**Value:** 23
CONTEXT_CONTROL	**Value:** 65537
CONTEXT_DEBUG_REGISTERS	**Value:** 65552
CONTEXT_FULL	**Value:** 65543
CREATE_NEW_CONSOLE	**Value:** 16
CREATE_PROCESS_DEBUG_EVENT	**Value:** 3
CREATE_THREAD_DEBUG_EVENT	**Value:** 2
DBG_CONTINUE	**Value:** 65538
DBG_EXCEPTION_HANDLED	**Value:** 65537
DBG_EXCEPTION_NOT_HANDLED	**Value:** 2147549185
DEBUG_ONLY_THIS_PROCESS	**Value:** 2
DEBUG_PROCESS	**Value:** 1
DEFAULT_MODE	**Value:** 0
EFLAGS_RF	**Value:** 65536
EFLAGS_TRAP	**Value:** 256
ERROR_NO_MORE_FILES	**Value:** 18
EXCEPTION_ACCESS_VIOLATION	**Value:** 3221225477

continued on next page

Name	Description
EXCEPTION_BREAKPOINT	**Value:** 2147483651
EXCEPTION_DEBUG_EVENT	**Value:** 1
EXCEPTION_GUARD_PAGE	**Value:** 2147483649
EXCEPTION_SINGLE_STEP	**Value:** 2147483652
EXIT_PROCESS_DEBUG_EVENT	**Value:** 5
EXIT_THREAD_DEBUG_EVENT	**Value:** 4
FILE_MAP_READ	**Value:** 4
FORMAT_MESSAGE_ALLOCATE_BUFFER	**Value:** 256
FORMAT_MESSAGE_FROM_SYSTEM	**Value:** 4096
GetLastError	**Value:** <_FuncPtr object at 0x0253AA08>
HW_ACCESS	**Value:** 3
HW_EXECUTE	**Value:** 0
HW_WRITE	**Value:** 1
INVALID_HANDLE_VALUE	**Value:** 4294967295
LOAD_DLL_DEBUG_EVENT	**Value:** 6
MEM_COMMIT	**Value:** 4096
MEM_DECOMMIT	**Value:** 16384
MEM_IMAGE	**Value:** 16777216
MEM_RELEASE	**Value:** 32768
MIB_TCP_STATE_LISTEN	**Value:** 2
OUTPUT_DEBUG_STRING_EVENT	**Value:** 8
PAGE_EXECUTE	**Value:** 16
PAGE_EXECUTE_READ	**Value:** 32
PAGE_EXECUTE_READWRITE	**Value:** 64
PAGE_EXECUTE_WRITECOPY	**Value:** 128
PAGE_GUARD	**Value:** 256
PAGE_NOACCESS	**Value:** 1
PAGE_NOCACHE	**Value:** 512

continued on next page

Name	Description
PAGE_READONLY	**Value:** 2
PAGE_READWRITE	**Value:** 4
PAGE_WRITECOMBINE	**Value:** 1024
PAGE_WRITECOPY	**Value:** 8
PROCESS_ALL_ACCESS	**Value:** 2035711
RIP_EVENT	**Value:** 9
RTLD_GLOBAL	**Value:** 0
RTLD_LOCAL	**Value:** 0
SE_PRIVILEGE_ENABLED	**Value:** 2
SW_SHOW	**Value:** 5
SysDbgReadMsr	**Value:** 16
SysDbgWriteMsr	**Value:** 17
TCP_TABLE_OWNER_PID_ALL	**Value:** 5
TH32CS_INHERIT	**Value:** 2147483648
TH32CS_SNAPALL	**Value:** 15
TH32CS_SNAPHEAPLIST	**Value:** 1
TH32CS_SNAPMODULE	**Value:** 8
TH32CS_SNAPPROCESS	**Value:** 2
TH32CS_SNAPTHREAD	**Value:** 4
THREAD_ALL_ACCESS	**Value:** 2032639
TOKEN_ADJUST_PRIVILEGES	**Value:** 32
UDP_TABLE_OWNER_PID	**Value:** 1
UNLOAD_DLL_DEBUG_EVENT	**Value:** 7
USER_CALLBACK_DEBUG_EVENT	**Value:** 3735928559
VIRTUAL_MEM	**Value:** 12288
__package__	**Value:** 'morpher.pydbg'
c_types	**Value:** (<type '_ctypes.Structure'>, <class 'ctypes.c_char'>, <cl...
cdll	**Value:** <ctypes.LibraryLoader object at 0x0249FB90>
memmove	**Value:** <CFunctionType object at 0x0253AA80>
memset	**Value:** <CFunctionType object at 0x0253AAF8>

continued on next page

Name	Description
oledll	**Value:** <ctypes.LibraryLoader object at 0x0249FC30>
pydll	**Value:** <ctypes.LibraryLoader object at 0x0249FBB0>
pythonapi	**Value:** <PyDLL 'python dll', handle 1e000000 at 249fbd0>
windll	**Value:** <ctypes.LibraryLoader object at 0x0249FBF0>

48.2 Class system_dll

System DLL descriptor object, used to keep track of loaded system DLLs and locations.

To Do: Add PE parsing support.

48.2.1 Methods

__init__(*self*, *handle*, *base*)

Given a handle and base address of the loaded DLL, determine the DLL name and size to fully initialize the system DLL object.

Parameters

`handle`: Handle to the loaded DLL
(type=HANDLE)

`base`: Loaded address of DLL
(type=DWORD)

Raises

`pdx` An exception is raised on failure.

__del__(*self*)

Close the handle.

48.2.2 Class Variables

Name	Description
handle	**Value:** None
base	**Value:** None
name	**Value:** None

continued on next page

Name	Description
path	**Value:** `None`
pe	**Value:** `None`
size	**Value:** `0`

49 Module morpher.pydbg.windows_h

49.1 Variables

Name	Description
DEFAULT_MODE	**Value:** 0
GetLastError	**Value:** <_FuncPtr object at 0x0253AA08>
RTLD_GLOBAL	**Value:** 0
RTLD_LOCAL	**Value:** 0
__package__	**Value:** 'morpher.pydbg'
c_types	**Value:** (<type '_ctypes.Structure'>, <class 'ctypes.c_char'>, <cl...
cdll	**Value:** <ctypes.LibraryLoader object at 0x0249FB90>
memmove	**Value:** <CFunctionType object at 0x0253AA80>
memset	**Value:** <CFunctionType object at 0x0253AAF8>
oledll	**Value:** <ctypes.LibraryLoader object at 0x0249FC30>
pydll	**Value:** <ctypes.LibraryLoader object at 0x0249FBB0>
pythonapi	**Value:** <PyDLL 'python dll', handle 1e000000 at 249fbd0>
windll	**Value:** <ctypes.LibraryLoader object at 0x0249FBF0>

49.2 Class _TOKEN_PRIVILEGES

object
 ??._CData
 _ctypes.Structure
 morpher.pydbg.windows_h._TOKEN_PRIVILEGES

49.2.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.2.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.2.3 Class Variables

Name	Description
fields	**Value:** `[('PrivilegeCount', <class 'ctypes.c_ulong'>), ('Privileg...`
PrivilegeCount	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
Privileges	**Value:** `<Field type=_LUID_AND_ATTRIBUTES_Array_1, ofs=4, size=12>`

49.3 Class _STARTUPINFOA

object
 ??._CData
  _ctypes.Structure
   morpher.pydbg.windows_h._STARTUPINFOA

49.3.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.3.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.3.3 Class Variables

Name	Description
fields	**Value:** `[('cb', <class 'ctypes.c_ulong'>), ('lpReserved', <class ...`
cb	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
cbReserved2	**Value:** `<Field type=c_ushort, ofs=50, size=2>`
dwFillAttribute	**Value:** `<Field type=c_ulong, ofs=40, size=4>`
dwFlags	**Value:** `<Field type=c_ulong, ofs=44, size=4>`
dwX	**Value:** `<Field type=c_ulong, ofs=16, size=4>`
dwXCountChars	**Value:** `<Field type=c_ulong, ofs=32, size=4>`
dwXSize	**Value:** `<Field type=c_ulong, ofs=24, size=4>`
dwY	**Value:** `<Field type=c_ulong, ofs=20, size=4>`
dwYCountChars	**Value:** `<Field type=c_ulong, ofs=36, size=4>`

continued on next page

Name	Description
dwYSize	**Value:** <Field type=c_ulong, ofs=28, size=4>
hStdError	**Value:** <Field type=c_void_p, ofs=64, size=4>
hStdInput	**Value:** <Field type=c_void_p, ofs=56, size=4>
hStdOutput	**Value:** <Field type=c_void_p, ofs=60, size=4>
lpDesktop	**Value:** <Field type=LP_c_char, ofs=8, size=4>
lpReserved	**Value:** <Field type=LP_c_char, ofs=4, size=4>
lpReserved2	**Value:** <Field type=LP_c_ubyte, ofs=52, size=4>
lpTitle	**Value:** <Field type=LP_c_char, ofs=12, size=4>
wShowWindow	**Value:** <Field type=c_ushort, ofs=48, size=2>

49.4 Class _STARTUPINFOA

object
??._CData
_ctypes.Structure
morpher.pydbg.windows_h._STARTUPINFOA

49.4.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.4.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

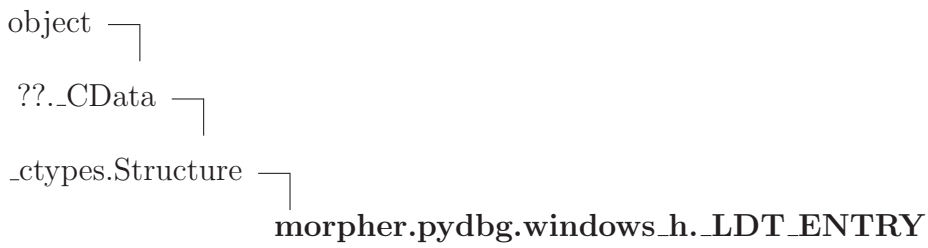
49.4.3 Class Variables

Name	Description
fields	**Value:** `[('cb', <class 'ctypes.c_ulong'>), ('lpReserved', <class ...`
cb	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
cbReserved2	**Value:** `<Field type=c_ushort, ofs=50, size=2>`
dwFillAttribute	**Value:** `<Field type=c_ulong, ofs=40, size=4>`
dwFlags	**Value:** `<Field type=c_ulong, ofs=44, size=4>`
dwX	**Value:** `<Field type=c_ulong, ofs=16, size=4>`
dwXCountChars	**Value:** `<Field type=c_ulong, ofs=32, size=4>`
dwXSize	**Value:** `<Field type=c_ulong, ofs=24, size=4>`
dwY	**Value:** `<Field type=c_ulong, ofs=20, size=4>`
dwYCountChars	**Value:** `<Field type=c_ulong, ofs=36, size=4>`
dwYSize	**Value:** `<Field type=c_ulong, ofs=28, size=4>`
hStdError	**Value:** `<Field type=c_void_p, ofs=64, size=4>`
hStdInput	**Value:** `<Field type=c_void_p, ofs=56, size=4>`
hStdOutput	**Value:** `<Field type=c_void_p, ofs=60, size=4>`
lpDesktop	**Value:** `<Field type=LP_c_char, ofs=8, size=4>`

continued on next page

Name	Description
lpReserved	**Value:** <Field type=LP_c_char, ofs=4, size=4>
lpReserved2	**Value:** <Field type=LP_c_ubyte, ofs=52, size=4>
lpTitle	**Value:** <Field type=LP_c_char, ofs=12, size=4>
wShowWindow	**Value:** <Field type=c_ushort, ofs=48, size=2>

49.5 Class _LDT_ENTRY



49.5.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.5.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.5.3 Class Variables

Name	Description
fields	**Value:** `[('LimitLow', <class 'ctypes.c_ushort'>), ('BaseLow', <cl...`
BaseLow	**Value:** `<Field type=c_ushort, ofs=2, size=2>`
HighWord	**Value:** `<Field type=N10_LDT_ENTRY3DOLLAR_4E, ofs=4, size=4>`
LimitLow	**Value:** `<Field type=c_ushort, ofs=0, size=2>`

49.6 Class _MEMORY_BASIC_INFORMATION

object
??._CData
_ctypes.Structure
morpher.pydbg.windows_h._MEMORY_BASIC_INFORMATION

49.6.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.6.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	

continued on next page

Name	Description
Inherited from object	
__class__	

49.6.3 Class Variables

Name	Description
fields	**Value:** `[('BaseAddress', <class 'ctypes.c_void_p'>), ('Allocation...`
AllocationBase	**Value:** `<Field type=c_void_p, ofs=4, size=4>`
AllocationProtect	**Value:** `<Field type=c_ulong, ofs=8, size=4>`
BaseAddress	**Value:** `<Field type=c_void_p, ofs=0, size=4>`
Protect	**Value:** `<Field type=c_ulong, ofs=20, size=4>`
RegionSize	**Value:** `<Field type=c_ulong, ofs=12, size=4>`
State	**Value:** `<Field type=c_ulong, ofs=16, size=4>`
Type	**Value:** `<Field type=c_ulong, ofs=24, size=4>`

49.7 Class _DEBUG_EVENT

object
??._CData
_ctypes.Structure
morpher.pydbg.windows_h._DEBUG_EVENT

49.7.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

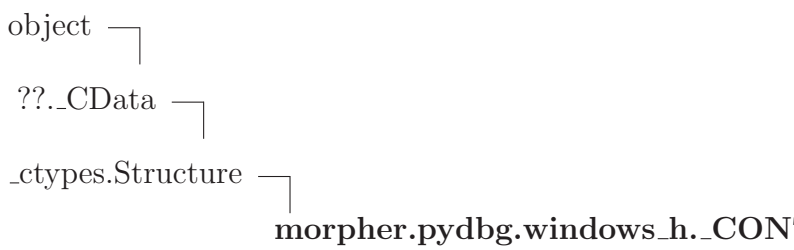
49.7.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.7.3 Class Variables

Name	Description
fields	**Value:** `[('dwDebugEventCode', <class 'ctypes.c_ulong'>), ('dwProc...`
dwDebugEventCode	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
dwProcessId	**Value:** `<Field type=c_ulong, ofs=4, size=4>`
dwThreadId	**Value:** `<Field type=c_ulong, ofs=8, size=4>`
u	**Value:** `<Field type=N12_DEBUG_EVENT4DOLLAR_39E, ofs=12, size=84>`

49.8 Class _CONTEXT



49.8.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.8.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.8.3 Class Variables

Name	Description
fields	**Value:** `[('ContextFlags', <class 'ctypes.c_ulong'>), ('Dr0', <cla...`
ContextFlags	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
Dr0	**Value:** `<Field type=c_ulong, ofs=4, size=4>`
Dr1	**Value:** `<Field type=c_ulong, ofs=8, size=4>`
Dr2	**Value:** `<Field type=c_ulong, ofs=12, size=4>`
Dr3	**Value:** `<Field type=c_ulong, ofs=16, size=4>`
Dr6	**Value:** `<Field type=c_ulong, ofs=20, size=4>`
Dr7	**Value:** `<Field type=c_ulong, ofs=24, size=4>`
EFlags	**Value:** `<Field type=c_ulong, ofs=192, size=4>`

continued on next page

Name	Description
Eax	**Value:** <Field type=c_ulong, ofs=176, size=4>
Ebp	**Value:** <Field type=c_ulong, ofs=180, size=4>
Ebx	**Value:** <Field type=c_ulong, ofs=164, size=4>
Ecx	**Value:** <Field type=c_ulong, ofs=172, size=4>
Edi	**Value:** <Field type=c_ulong, ofs=156, size=4>
Edx	**Value:** <Field type=c_ulong, ofs=168, size=4>
Eip	**Value:** <Field type=c_ulong, ofs=184, size=4>
Esi	**Value:** <Field type=c_ulong, ofs=160, size=4>
Esp	**Value:** <Field type=c_ulong, ofs=196, size=4>
ExtendedRegisters	**Value:** <Field type=c_ubyte_Array_512, ofs=204, size=512>
FloatSave	**Value:** <Field type=_FLOATING_SAVE_AREA, ofs=28, size=112>
SegCs	**Value:** <Field type=c_ulong, ofs=188, size=4>
SegDs	**Value:** <Field type=c_ulong, ofs=152, size=4>
SegEs	**Value:** <Field type=c_ulong, ofs=148, size=4>
SegFs	**Value:** <Field type=c_ulong, ofs=144, size=4>
SegGs	**Value:** <Field type=c_ulong, ofs=140, size=4>
SegSs	**Value:** <Field type=c_ulong, ofs=200, size=4>

49.9 Class _SYSTEM_INFO

object
??._CData
_ctypes.Structure
morpher.pydbg.windows_h._SYSTEM_INFO

49.9.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.9.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.9.3 Class Variables

Name	Description
fields	**Value:** `[('_', <class 'morpher.pydbg.windows_h.N12_SYSTEM_INFO4DO...`
_	**Value:** `<Field type=N12_SYSTEM_INFO4DOLLAR_37E, ofs=0, size=4>`
dwActiveProcessorMask	**Value:** `<Field type=c_ulong, ofs=16, size=4>`

continued on next page

Name	Description
dwAllocationGranularity	**Value:** <Field type=c_ulong, ofs=28, size=4>
dwNumberOfProcessors	**Value:** <Field type=c_ulong, ofs=20, size=4>
dwPageSize	**Value:** <Field type=c_ulong, ofs=4, size=4>
dwProcessorType	**Value:** <Field type=c_ulong, ofs=24, size=4>
lpMaximumApplicationAddress	**Value:** <Field type=c_void_p, ofs=12, size=4>
lpMinimumApplicationAddress	**Value:** <Field type=c_void_p, ofs=8, size=4>
wProcessorLevel	**Value:** <Field type=c_ushort, ofs=32, size=2>
wProcessorRevision	**Value:** <Field type=c_ushort, ofs=34, size=2>

49.10 Class _PROCESS_INFORMATION

object
??._CData
_ctypes.Structure
morpher.pydbg.windows_h._PROCESS_INFORMATION

49.10.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

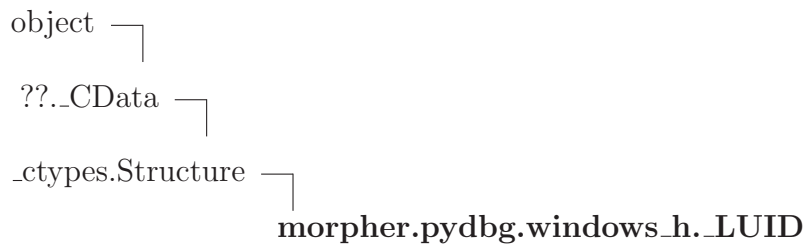
49.10.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.10.3 Class Variables

Name	Description
fields	**Value:** `[('hProcess', <class 'ctypes.c_void_p'>), ('hThread', <cl...`
dwProcessId	**Value:** `<Field type=c_ulong, ofs=8, size=4>`
dwThreadId	**Value:** `<Field type=c_ulong, ofs=12, size=4>`
hProcess	**Value:** `<Field type=c_void_p, ofs=0, size=4>`
hThread	**Value:** `<Field type=c_void_p, ofs=4, size=4>`

49.11 Class _LUID



49.11.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

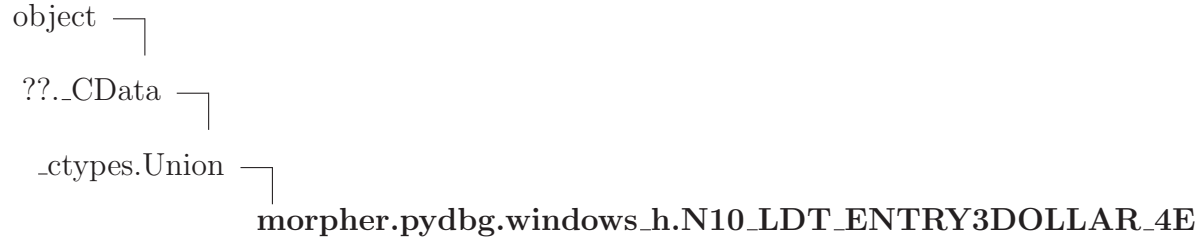
49.11.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.11.3 Class Variables

Name	Description
fields	**Value:** `[('LowPart', <class 'ctypes.c_ulong'>), ('HighPart', <cla...`
HighPart	**Value:** `<Field type=c_long, ofs=4, size=4>`
LowPart	**Value:** `<Field type=c_ulong, ofs=0, size=4>`

49.12 Class N10_LDT_ENTRY3DOLLAR_4E



49.12.1 Methods

Inherited from _ctypes.Union

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.12.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.12.3 Class Variables

Name	Description
fields	**Value:** `[('Bytes', <class 'morpher.pydbg.windows_h.N10_LDT_ENTRY3...`
Bits	**Value:** `<Field type=N10_LDT_ENTRY3DOLLAR_43DOLLAR_6E, ofs=0, size=4>`
Bytes	**Value:** `<Field type=N10_LDT_ENTRY3DOLLAR_43DOLLAR_5E, ofs=0, size=4>`

49.13 Class N10_LDT_ENTRY3DOLLAR_43DOLLAR_5E

object
 ??._CData
  _ctypes.Structure
   morpher.pydbg.windows_h.N10_LDT_ENTRY3DOLLAR_43DOLLAR_5E

49.13.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.13.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.13.3 Class Variables

Name	Description
fields	**Value:** `[('BaseMid', <class 'ctypes.c_ubyte'>), ('Flags1', <class...`
BaseHi	**Value:** `<Field type=c_ubyte, ofs=3, size=1>`
BaseMid	**Value:** `<Field type=c_ubyte, ofs=0, size=1>`
Flags1	**Value:** `<Field type=c_ubyte, ofs=1, size=1>`
Flags2	**Value:** `<Field type=c_ubyte, ofs=2, size=1>`

49.14 Class N10_LDT_ENTRY3DOLLAR_43DOLLAR_6E

object
 ??._CData
  _ctypes.Structure
   morpher.pydbg.windows_h.N10_LDT_ENTRY3DOLLAR_43DOLLAR_6E

49.14.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.14.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

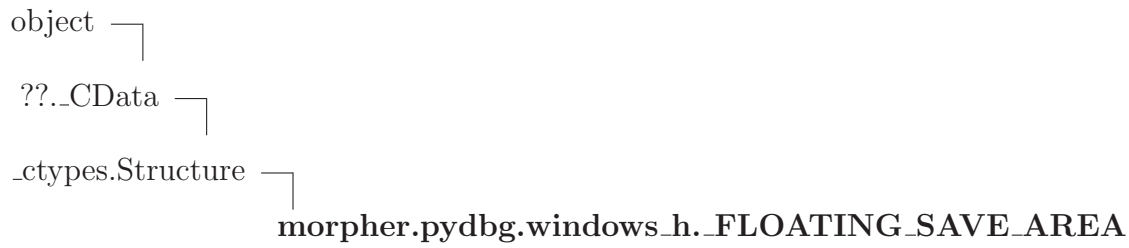
49.14.3 Class Variables

Name	Description
fields	**Value:** `[('BaseMid', <class 'ctypes.c_ulong'>, 8), ('Type', <clas...`
BaseHi	**Value:** `<Field type=c_ulong, ofs=0:24, bits=8>`
BaseMid	**Value:** `<Field type=c_ulong, ofs=0:0, bits=8>`
Default_Big	**Value:** `<Field type=c_ulong, ofs=0:22, bits=1>`
Dpl	**Value:** `<Field type=c_ulong, ofs=0:13, bits=2>`
Granularity	**Value:** `<Field type=c_ulong, ofs=0:23, bits=1>`
LimitHi	**Value:** `<Field type=c_ulong, ofs=0:16, bits=4>`
Pres	**Value:** `<Field type=c_ulong, ofs=0:15, bits=1>`

continued on next page

Name	Description
Reserved_0	**Value:** <Field type=c_ulong, ofs=0:21, bits=1>
Sys	**Value:** <Field type=c_ulong, ofs=0:20, bits=1>
Type	**Value:** <Field type=c_ulong, ofs=0:8, bits=5>

49.15 Class _FLOATING_SAVE_AREA



49.15.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.15.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.15.3 Class Variables

Name	Description
fields	**Value:** [('ControlWord', <class 'ctypes.c_ulong'>), ('StatusWord'...
ControlWord	**Value:** <Field type=c_ulong, ofs=0, size=4>
Cr0NpxState	**Value:** <Field type=c_ulong, ofs=108, size=4>
DataOffset	**Value:** <Field type=c_ulong, ofs=20, size=4>
DataSelector	**Value:** <Field type=c_ulong, ofs=24, size=4>
ErrorOffset	**Value:** <Field type=c_ulong, ofs=12, size=4>
ErrorSelector	**Value:** <Field type=c_ulong, ofs=16, size=4>
RegisterArea	**Value:** <Field type=c_ubyte_Array_80, ofs=28, size=80>
StatusWord	**Value:** <Field type=c_ulong, ofs=4, size=4>
TagWord	**Value:** <Field type=c_ulong, ofs=8, size=4>

49.16 Class N12_SYSTEM_INFO4DOLLAR_37E

object
??._CData
_ctypes.Union
morpher.pydbg.windows_h.N12_SYSTEM_INFO4DOLLAR_37E

49.16.1 Methods

Inherited from _ctypes.Union

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.16.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.16.3 Class Variables

Name	Description
fields	**Value:** `[('dwOemId', <class 'ctypes.c_ulong'>), ('_', <class 'mor...`
_	**Value:** `<Field type=N12_SYSTEM_INFO4DOLLAR_374DOLLAR_38E, ofs=0, ...`
dwOemId	**Value:** `<Field type=c_ulong, ofs=0, size=4>`

49.17 Class N12_SYSTEM_INFO4DOLLAR_374DOLLAR_38E

object
 ??._CData
  _ctypes.Structure
   morpher.pydbg.windows_h.N12_SYSTEM_INFO4DOLLAR_374DOLLAR_

49.17.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

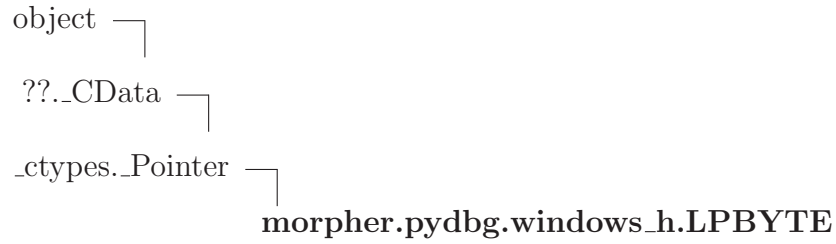
49.17.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.17.3 Class Variables

Name	Description
fields	**Value:** `[('wProcessorArchitecture', <class 'ctypes.c_ushort'>), (...`
wProcessorArchitecture	**Value:** `<Field type=c_ushort, ofs=0, size=2>`
wReserved	**Value:** `<Field type=c_ushort, ofs=2, size=2>`

49.18 Class LPBYTE



49.18.1 Methods

Inherited from _ctypes._Pointer

__delitem__(), __getitem__(), __getslice__(), __init__(), __new__(), __nonzero__(), __setitem__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.18.2 Properties

Name	Description
Inherited from _ctypes._Pointer contents	
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.19 Class N12_DEBUG_EVENT4DOLLAR_39E

object
 ??._CData
  _ctypes.Union
   morpher.pydbg.windows_h.N12_DEBUG_EVENT4DOLLAR_39E

49.19.1 Methods

Inherited from _ctypes.Union

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

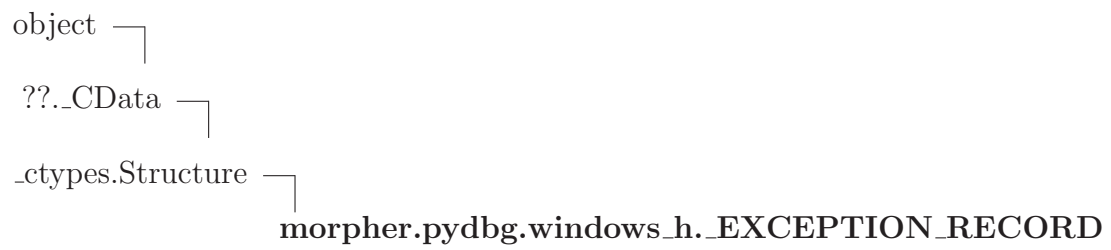
49.19.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.19.3 Class Variables

Name	Description
fields	**Value:** `[('Exception', <class 'morpher.pydbg.windows_h._EXCEPTION...`
CreateProcessInfo	**Value:** `<Field type=_CREATE_PROCESS_DEBUG_INFO, ofs=0, size=40>`
CreateThread	**Value:** `<Field type=_CREATE_THREAD_DEBUG_INFO, ofs=0, size=12>`
DebugString	**Value:** `<Field type=_OUTPUT_DEBUG_STRING_INFO, ofs=0, size=8>`
Exception	**Value:** `<Field type=_EXCEPTION_DEBUG_INFO, ofs=0, size=84>`
ExitProcess	**Value:** `<Field type=_EXIT_PROCESS_DEBUG_INFO, ofs=0, size=4>`
ExitThread	**Value:** `<Field type=_EXIT_THREAD_DEBUG_INFO, ofs=0, size=4>`
LoadDll	**Value:** `<Field type=_LOAD_DLL_DEBUG_INFO, ofs=0, size=24>`
RipInfo	**Value:** `<Field type=_RIP_INFO, ofs=0, size=8>`
UnloadDll	**Value:** `<Field type=_UNLOAD_DLL_DEBUG_INFO, ofs=0, size=4>`

49.20 Class _EXCEPTION_RECORD



49.20.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.20.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

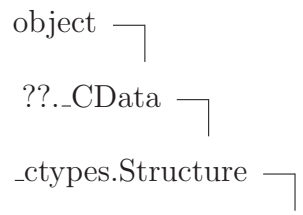
49.20.3 Class Variables

Name	Description
fields	**Value:** `[('ExceptionCode', <class 'ctypes.c_ulong'>), ('Exception...`
ExceptionAddress	**Value:** `<Field type=c_void_p, ofs=12, size=4>`
ExceptionCode	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
ExceptionFlags	**Value:** `<Field type=c_ulong, ofs=4, size=4>`

continued on next page

Name	Description
ExceptionInformation	**Value:** <Field type=c_ulong_Array_15, ofs=20, size=60>
ExceptionRecord	**Value:** <Field type=LP__EXCEPTION_RECORD, ofs=8, size=4>
NumberParameters	**Value:** <Field type=c_ulong, ofs=16, size=4>

49.21 Class _EXCEPTION_DEBUG_INFO



morpher.pydbg.windows_h._EXCEPTION_DEBUG_INFO

49.21.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.21.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.21.3 Class Variables

Name	Description
fields	**Value:** `[('ExceptionRecord', <class 'morpher.pydbg.windows_h._EXC...`
ExceptionRecord	**Value:** `<Field type=_EXCEPTION_RECORD, ofs=0, size=80>`
dwFirstChance	**Value:** `<Field type=c_ulong, ofs=80, size=4>`

49.22 Class _CREATE_THREAD_DEBUG_INFO

object
??._CData
_ctypes.Structure
morpher.pydbg.windows_h._CREATE_THREAD_DEBUG_INFO

49.22.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.22.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.22.3 Class Variables

Name	Description
fields	**Value:** `[('hThread', <class 'ctypes.c_void_p'>), ('lpThreadLocalB...`
hThread	**Value:** `<Field type=c_void_p, ofs=0, size=4>`
lpStartAddress	**Value:** `<Field type=WinFunctionType, ofs=8, size=4>`
lpThreadLocalBase	**Value:** `<Field type=c_void_p, ofs=4, size=4>`

49.23 Class _CREATE_PROCESS_DEBUG_INFO

object
??._CData
_ctypes.Structure
morpher.pydbg.windows_h._CREATE_PROCESS_DEBUG_INFO

49.23.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.23.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.23.3 Class Variables

Name	Description
fields	**Value:** `[('hFile', <class 'ctypes.c_void_p'>), ('hProcess', <clas...`
dwDebugInfoFileOffset	**Value:** `<Field type=c_ulong, ofs=16, size=4>`
fUnicode	**Value:** `<Field type=c_ushort, ofs=36, size=2>`
hFile	**Value:** `<Field type=c_void_p, ofs=0, size=4>`
hProcess	**Value:** `<Field type=c_void_p, ofs=4, size=4>`
hThread	**Value:** `<Field type=c_void_p, ofs=8, size=4>`
lpBaseOfImage	**Value:** `<Field type=c_void_p, ofs=12, size=4>`
lpImageName	**Value:** `<Field type=c_void_p, ofs=32, size=4>`
lpStartAddress	**Value:** `<Field type=WinFunctionType, ofs=28, size=4>`
lpThreadLocalBase	**Value:** `<Field type=c_void_p, ofs=24, size=4>`
nDebugInfoSize	**Value:** `<Field type=c_ulong, ofs=20, size=4>`

49.24 Class _EXIT_THREAD_DEBUG_INFO

object
??._CData
_ctypes.Structure
morpher.pydbg.windows_h._EXIT_THREAD_DEBUG_INFO

49.24.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

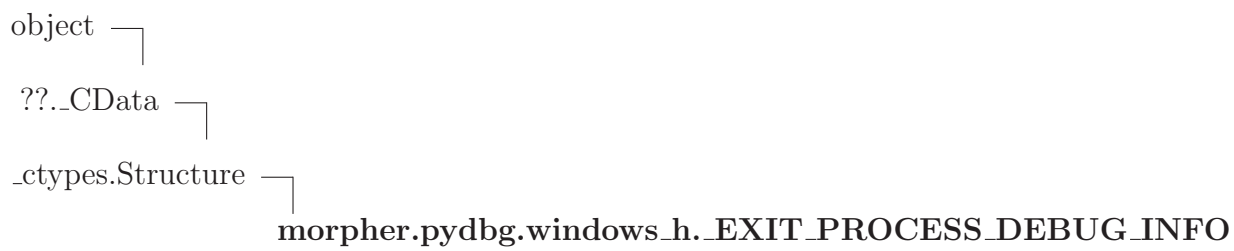
49.24.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.24.3 Class Variables

Name	Description
fields	**Value:** `[('dwExitCode', <class 'ctypes.c_ulong'>)]`
dwExitCode	**Value:** `<Field type=c_ulong, ofs=0, size=4>`

49.25 Class _EXIT_PROCESS_DEBUG_INFO



49.25.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.25.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.25.3 Class Variables

Name	Description
fields	**Value:** `[('dwExitCode', <class 'ctypes.c_ulong'>)]`
dwExitCode	**Value:** `<Field type=c_ulong, ofs=0, size=4>`

49.26 Class _LOAD_DLL_DEBUG_INFO

object
??._CData
_ctypes.Structure
morpher.pydbg.windows_h._LOAD_DLL_DEBUG_INFO

49.26.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.26.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.26.3 Class Variables

Name	Description
fields	**Value:** `[('hFile', <class 'ctypes.c_void_p'>), ('lpBaseOfDll', <c...`
dwDebugInfoFileOffset	**Value:** `<Field type=c_ulong, ofs=8, size=4>`
fUnicode	**Value:** `<Field type=c_ushort, ofs=20, size=2>`
hFile	**Value:** `<Field type=c_void_p, ofs=0, size=4>`
lpBaseOfDll	**Value:** `<Field type=c_void_p, ofs=4, size=4>`
lpImageName	**Value:** `<Field type=c_void_p, ofs=16, size=4>`
nDebugInfoSize	**Value:** `<Field type=c_ulong, ofs=12, size=4>`

49.27 Class _UNLOAD_DLL_DEBUG_INFO

object
??._CData
_ctypes.Structure
morpher.pydbg.windows_h._UNLOAD_DLL_DEBUG_INFO

49.27.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.27.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.27.3 Class Variables

Name	Description
fields	**Value:** `[('lpBaseOfDll', <class 'ctypes.c_void_p'>)]`
lpBaseOfDll	**Value:** `<Field type=c_void_p, ofs=0, size=4>`

49.28 Class _OUTPUT_DEBUG_STRING_INFO

object
 ??._CData
 _ctypes.Structure
 morpher.pydbg.windows_h._OUTPUT_DEBUG_STRING_INFO

49.28.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

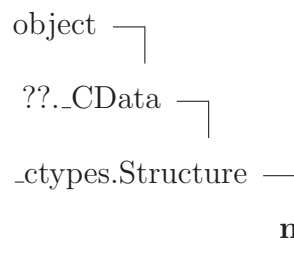
49.28.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.28.3 Class Variables

Name	Description
fields	**Value:** `[('lpDebugStringData', <class 'ctypes.LP_c_char'>), ('fUn...`
fUnicode	**Value:** `<Field type=c_ushort, ofs=4, size=2>`
lpDebugStringData	**Value:** `<Field type=LP_c_char, ofs=0, size=4>`
nDebugStringLength	**Value:** `<Field type=c_ushort, ofs=6, size=2>`

49.29 Class _RIP_INFO



morpher.pydbg.windows_h._RIP_INFO

49.29.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

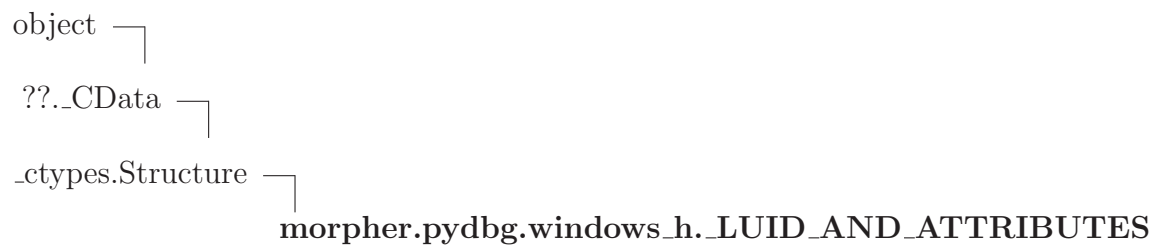
49.29.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.29.3 Class Variables

Name	Description
fields	**Value:** `[('dwError', <class 'ctypes.c_ulong'>), ('dwType', <class...`
dwError	**Value:** `<Field type=c_ulong, ofs=0, size=4>`
dwType	**Value:** `<Field type=c_ulong, ofs=4, size=4>`

49.30 Class _LUID_AND_ATTRIBUTES



49.30.1 Methods

Inherited from _ctypes.Structure

__init__(), __new__()

Inherited from ??._CData

__ctypes_from_outparam__(), __hash__(), __reduce__(), __setstate__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

49.30.2 Properties

Name	Description
Inherited from ??._CData _b_base_, _b_needsfree_	
Inherited from object __class__	

49.30.3 Class Variables

Name	Description
fields	**Value:** `[('Luid', <class 'morpher.pydbg.windows_h._LUID'>), ('Att...`
Attributes	**Value:** `<Field type=c_ulong, ofs=8, size=4>`
Luid	**Value:** `<Field type=_LUID, ofs=0, size=8>`

50 Package morpher.trace

Contains various modules used to model and store data captured from a function call.

(GRAPH)

The core components of this module build off each other to encapsulate chunks of memory captured from another process, along with enough information about types, address, etc to be able to reconstruct and replay data to a function call. One of the major design goals was to make these objects serializable so they could be stored to a file or sent through a pipe and properly reconstructed once they were deserialized.

The contents of memory are captured in the form of `Block`s, which provide an interface to read and write their contents. Collections of blocks are managed by `Memory` objects; one of their main responsibilities is to preserve pointer relationships, by recording pointers and then "patching" them on demand to point to the same objects after deserialization.

Type information is captured mainly through the use of `Tag` objects, which merely record an address and a format string similar to those used by the *struct* module. A `TypeManager` object can be used to translate these format types into *ctypes* objects suitable as function arguments for a DLL function. One of the most important aspects of these classes is to properly reconstruct *ctypes* struct and union classes from stored information about user-defined types.

`Snapshot` puts all the pieces together in order to capture a function call in it's entirety. It contains a `Memory` object with the actual data, a collection of `Tag`s giving types for that data, and a `TypeManager` that can translate those tags into meaningful classes. `Snapshot` is responsible for using all this information to reconstruct *ctypes* objects respresenting the original arguments that can be used in a function call, and in such a way that all pointers point to the same data that they did when originally captured.

Finally, `Trace` serves as a top-level object that pairs a list of `Snapshot` objects to be replayed in order, along with the `TypeManager` object used by all of those `Snapshot`s.

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: November 13, 2011

50.1 Modules

- **block**: Contains the `Block` class definition for maintaining a piece of memory *(Section 51, p. 309)*
- **memory**: Contains the `Memory` class for maintaining a collection of `Block`

(Section 52, p. 314)
- **snapshot**: Contains the `Snapshot` class for storing and replaying a function call *(Section 53, p. 320)*
- **tag**: Contains the `Tag` class definition for pairing addresses with types *(Section 54, p. 324)*
- **trace**: Contains the `Trace` class definition for storing a list of `Snapshot`s *(Section 55, p. 327)*
- **typemanager**: Contains the `TypeManager` class for storing and reconstructing types *(Section 56, p. 330)*

51 Module morpher.trace.block

Contains the `Block` class definition for maintaining a piece of memory

Author: Rob Waaser

Contact: robwaaser@gmail.com

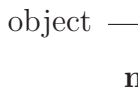
Organization: Carnegie Mellon University

Since: October 26, 2011

51.1 Variables

Name	Description
__package__	**Value:** `'morpher.trace'`

51.2 Class Block



morpher.trace.block.Block

Encapsulates a memory block and provides an interface for reading/writing that memory.

The memory block is represented as a byte string, and is stored along with the "virtual address" that the memory block starts at. Reads and writes use these virtual addresses and the `struct` module to access the contents of the byte string. The class is maintained in such a way that it can be serialized and deserialized using the *pickle* module, and provides a `translate` method that can take a virtual address and return the actual address the bytestring currently occupies.

51.2.1 Methods

__init__(*self*, *addr*, *data*)

Stores an array of raw bytes along with the virtual address it starts at. The byte string is originally stored as a ctypes string buffer, which allows us to retrieve the actual raw address of the byte string.

Parameters

`addr`: The virtual address of the beginning of the data

(type=integer)

`data`: The byte string to store

(type=byte string)

Overrides: object.__init__

setActive(*self*, *flag*)

Converts the internal byte string to and from a serializable string.

Blocks can't be pickled if they contain ctypes pointers, so this method "turns off" the block and converts the data to a pickle-friendly string if the flag is *False*, and the reverse if the flag is *True*.

Parameters

`flag`: The state the block should be set to

(type=Boolean)

read(*self*, *addr*, *size*=`None`, *fmt*=`None`)

Reads data from this `Block` as either a raw byte string or a tuple of objects

Given an address in this `Block` and a size in bytes, returns 'size' raw bytes present at that address. If an optional format of the type specified in the `struct` module is given, the size parameter is ignored and the fmt is used to unpack and return the contents as a tuple of objects

Parameters

`addr`: The virtual address to read from

(type=integer)

`size`: The number of bytes to read

(type=integer)

`fmt`: The format of the object(s) to read

(type=string)

Return Value

Raw byte string or tuple of objects

(type=byte string or tuple)

Raises

`Exception` If neither the size nor format is supplied

write(*self*, *addr*, *data*, *fmt*=`None`)

Given an address in this `Block` and a byte string or tuple of objects, updates the memory at that address.

If a `struct`-style format is given, data is interpreted as a tuple of objects that should be first packed into a byte string before being written to memory. If a format is not specified, data is interpreted as a raw byte string to be written to memory verbatim.

Parameters

`addr`: The virtual address to write to

(type=integer)

`data`: The objects or bytes to write to memory

(type=Tuple or byte string)

`fmt`: An optional format string

(type=string)

contains(*self*, *addr*, *size*)

Returns True if range [addr, addr + size - 1] is contained in this block.

Parameters

`addr`: The starting address of the range in question

(type=integer)

`size`: The size of the range in question

(type=integer)

Return Value

True if the range is entirely contained by this `Block`, False otherwise.

(type=Boolean)

translate(*self*, *addr*)

Given a virtual address in the captured block's memory space, returns the actual address in memory of the data pointed to by the virtual address.

Parameters

`addr`: The virtual address to translate

(type=integer)

Return Value

The real address corresponding to the given virtual address

(type=integer)

toString(*self*)

Creates a pretty-printed string containing the contents of this `Block` in a format suitable for display.

Return Value

A string representing this object's contents

(type=string)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

51.2.2 Properties

Name	Description
Inherited from object	

continued on next page

Name	Description
__class__	

51.2.3 Instance Variables

Name	Description
active	Boolean indicating if the byte array is in a serializable state (*False*) or not (*True*)
addr	The virtual address the byte string starts at
data	The stored byte string
size	The length of the stored byte string

52 Module morpher.trace.memory

Contains the `Memory` class for maintaining a collection of `Block`

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 26, 2011

52.1 Variables

Name	Description
__package__	**Value:** `'morpher.trace'`

52.2 Class Memory

object
└
morpher.trace.memory.Memory

Acts as an interface to a collection of `Block` objects and provides methods for patching pointers into those memory segments.

A single `Memory` object maintains a collection of `Block` objects and serves as a global interface to those blocks that hides the details of which block is actually being read to or written from. This class is designed to be serialized and deserialized, and provides methods that can update pointers after their target objects have been moved to new addresses.

Pointers are "registered" by their addresses using the `registerPointer` method. Calling the `patch` method causes the value of each pointer to be fetched, and the virtual target address to be translated to the real address that data now occupies using the `Block` translate method.

Note: It is assumed that the underlying `Block` objects contain non-overlapping and non-consecutive ranges of memory

52.2.1 Methods

__init__(*self*, *blklist*)

Takes a list of (address, data) tuples, where address is a virtual address and data is a byte string representing the memory contents to store at that address, and adds them to the internal memory map. It is assumed that these blocks contain non-overlapping and non-consecutive ranges of memory

Parameters

`blklist`: The contents of this `Memory` as a list of (address, data) pairs

(type=(integer, byte string) tuple list)

Overrides: object.__init__

Requires: blklist must consist of disjoint memory ranges

__getstate__(*self*)

The *pickle* system calls this method when dumping. Turns off all the blocks so they can be pickled, then returns this object's __dict__ attribute for serialization.

Return Value

This object's __dict__ attribute

(type=dictionary)

Note: The blocks will automatically re-activate themselves individually when an operation is performed on them - otherwise `setActive` can force them all to reactivate right then

__setstate__(*self*, *newdict*)

Pickle calls this method when unpickling. Restores this object's __dict__ from the unserialized version given, then reactivates all of the constituent `Block` objects.

Parameters

`newdict`: The deserialized __dict__ for this object

(type=dictionary)

setActive(*self*)

Utility method to re-enable all `Block`s in this object

registerPointer(*self*, *addr*)

Stores the given address in a set of addresses pointing to pointers in memory, so that the pointer contents can be properly updated to preserve their meaning using `patch`.

Parameters

`addr`: The address of the pointer object in this memory

(type=integer)

Raises

`Exception` If the range [addr, addr + sizeof(pointer) - 1] is not contained in this memory

unregisterPointer(*self*, *addr*)

Removes an address from the internal pointer registry that was previously added using `registerPointer`.

Parameters

`addr`: The address of the pointer object in this memory

(type=integer)

read(*self*, *addr*, *size*=`None`, *fmt*=`None`)

Reads data from this `Memory` as either a raw byte string or a tuple of objects

Given an address in this `Memory` and a size in bytes, returns 'size' raw bytes present at that address. If an optional format of the type specified in the `struct` module is given, the size parameter is ignored and the fmt is used to unpack and return the contents as a tuple of objects

Parameters

`addr:` The virtual address to read from

(type=integer)

`size:` The number of bytes to read

(type=integer)

`fmt:` The format of the object(s) to read

(type=string)

Return Value

Raw byte string or tuple of objects

(type=byte string or tuple)

Raises

`Exception` If neither the size nor format is supplied

`Exception` If the given address range is not totally contained by one of the underlying `Block`s.

write(*self*, *addr*, *data*, *fmt*=`None`)

Given an address in this `Memory` and a byte string or tuple of objects, updates the memory at that address.

If a `struct`-style format is given, data is interpreted as a tuple of objects that should be first packed into a byte string before being written to memory. If a format is not specified, data is interpreted as a raw byte string to be written to memory verbatim.

Parameters

`addr`: The virtual address to write to

(type=integer)

`data`: The objects or bytes to write to memory

(type=Tuple or byte string)

`fmt`: An optional format string

(type=string)

Raises

`Exception` If the given memory range is totally contained by one of the constituent `Block`s.

patch(*self*)

For each pointer in this `Memory` whose address is registered, this method updates the pointer's value to reflect the ACTUAL address of the object it originally pointed to.

containsAddress(*self*, *addr*, *size*=`1`)

Checks whether the given range is wholly contained in this `Memory` object.

Parameters

`addr`: The virtual address the range starts at

(type=integer)

`size`: The size of the range to check

(type=integer)

Return Value

True if the range exists, *False* otherwise

(type=Boolean)

toString(*self*)

Creates a pretty-printed string containing the contents of this `Memory` in a format suitable for display.

Return Value

A string representing this object's contents

(type=string)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

52.2.2 Properties

Name	Description
Inherited from object	
__class__	

52.2.3 Instance Variables

Name	Description
mem	A dictionary mapping addresses to `Block` objects
pointers	A set containing addresses of pointer objects

53 Module morpher.trace.snapshot

Contains the `Snapshot` class for storing and replaying a function call

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: November 13, 2011

53.1 Variables

Name	Description
__package__	**Value:** `'morpher.trace'`

53.2 Class Snapshot

object
└ **morpher.trace.snapshot.Snapshot**

Contains enough information to replay a captured function call in its entirety.

A `Memory` object is combined with a set of `Tag` objects and a `TypeManager` object to act as snapshot of a function call. The `Memory` is used to store the actual argument values observed on the stack and the values they point to, while the `Tag` objects assign type information to these recorded values. A supplied `TypeManager` object can then be used to translate this type data into actual classes that can be used to instantiate objects and load them with values from the memory capture, which can then be used by the `ctypes` to replay the originally recorded function call.

To Do: Add the capability to record and replay global variables

53.2.1 Methods

__init__(*self*, *name*, *blklist*)

Stores the given function information and the contents of memory described as a list of (address, data) tuples, where address is a virtual address and data is a byte string to store at that address. Initially the tag set and argument list are empty.

Parameters

`name:` The name of the function call captured

(type=string)

`blklist:` The contents of this `Memory` as a list of (address, data) pairs

(type=(integer, byte string) tuple list)

Overrides: object.__init__

Requires: blklist must consist of disjoint memory ranges

setArgs(*self*, *args*)

Saves an ordered list of argument tags for this function call.

Parameters

`args:` A list of `Tag`s describing the function arguments in the same order they were observed

(type=`Tag` object list)

addTag(*self*, *tag*)

Adds the given `Tag` object to the internal tag set

Parameters

`tag:` The tag object to register

(type=`Tag` object)

Raises

`Exception` If the tag's address is not valid for this object

removeTag(*self*, *tag*)

Removes a `Tag` object previously given to `addTag`

Parameters

`tag:` The tag object to remove

(type=`Tag` object)

replay(*self*, *typeman*)

Patches internal pointers and extracts a list of argument objects observed for this captured function call.

Any registered pointers are changed so they point to the actual addresses of the objects they originally referred to. The supplied `TypeManager` object is then used to construct `ctypes` classes that represent the equivalent C types for each object that was captured as an argument to this function call. The stored argument values are then used to create equivalent objects from the `ctypes` classes and returned in an ordered list, which can be used in a call to the same function loaded using `ctypes`.

Parameters

`typeman`: A `TypeManager` object used to interpret format strings used to tag objects in this `Snapshot`

(type=`TypeManager` object)

toString(*self*)

Creates a pretty-printed string containing the contents of this `Snapshot` in a format suitable for display.

Return Value

A string representing this object's contents

(type=string)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

53.2.2 Properties

Name	Description
Inherited from object	
__class__	

53.2.3 Instance Variables

Name	Description
args	Ordered list of `Tag` objects describing function arguments
mem	Internal `Memory` object for storing data

continued on next page

Name	Description
name	The name of the function call that was captured
tags	Set of `Tag` objects associating types with data
type_manager	Used to temporarily store a `TypeManager` used during a function call

54 Module morpher.trace.tag

Contains the `Tag` class definition for pairing addresses with types

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 30, 2011

54.1 Variables

Name	Description
__package__	**Value:** `None`

54.2 Class Tag

object
└
morpher.trace.tag.Tag

Pairs an address with a format string representing the type of the object located at that address, as an immutable object. The class also overrides the `__hash__` and `__eq__` methods so that, when added to a set, the set will recognize two Tags with equivalent information as the same object.

54.2.1 Methods

__init__(*self*, *addr*, *fmt*)

Initializes this tag with the given address/format pair

Parameters

`addr`: The address for this tag

(type=integer)

`fmt`: The format string representing the associated tag

(type=string)

Overrides: object.__init__

__**eq**__(*self*, *other*)

Compare this object to another `Tag` object

Parameters

`other`: A `Tag` object to compare to

(type=`Tag` object)

Return Value

True if this object matches the given one

(type=Boolean)

__**hash**__(*self*)

Returns a hash constructed from the address and format

Return Value

A unique hash for this object

(type=integer)

Overrides: object.__hash__

__**setattr**__(*self*, **args*)

Raise an exception if a change to the object is attempted

Raises

`TypeError` Always

Overrides: object.__setattr__

__**delattr**__(*self*, **args*)

Raise an exception if a change to the object is attempted

Raises

`TypeError` Always

Overrides: object.__delattr__

Inherited from object

__format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __sizeof__(), __str__(), __subclasshook__()

54.2.2 Properties

Name	Description
Inherited from object	

continued on next page

Name	Description
__class__	

54.2.3 Instance Variables

Name	Description
addr	The address being tagged
fmt	The format string associated with the address

55 Module morpher.trace.trace

Contains the `Trace` class definition for storing a list of `Snapshot`s

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: November 13, 2011

55.1 Variables

Name	Description
__package__	**Value:** `'morpher.trace'`

55.2 Class Trace

object
└
morpher.trace.trace.Trace

Pairs a `TypeManager` object with a list of `Snapshot` objects.

The `Snapshot` objects can be replayed in order using the shared `TypeManager` object, which allows this `Trace` to replay an entire series of function calls exactly as they were first observed.

Note: `Trace` is built entirely from serializable objects, allowing a `Trace` object to be saved using the *pickle* module and restored without any noticeable problems

55.2.1 Methods

__init__(*self*, *snapshots*, *usertypes*={})

Uses the supplied usertypes information to create a `TypeManager` object and stores it along with the ordered list of `Snapshot` objects.

Parameters

`snapshots`: A list of `Snapshot` objects in the order they were captured

*(type=**`Snapshot`** object list)*

`usertypes`: Optional dictionary mapping format strings to pairs of type strings and lists of fields' formats

(type=dictionary of string : (string, string list) pairs)

Overrides: object.__init__

replay(*self*)

Acts as a Python generator function that returns enough information to recreate each function call in the trace in order.

Returns a series of pairs, consisting of the function ordinal to be replayed and a list of arguments for that function.

Return Value

function ordinals paired with argument lists

(type=(ordinal, ctypes object list))

toString(*self*)

Creates a pretty-printed string containing the contents of this `Trace` in a format suitable for display.

Return Value

A string representing this object's contents

(type=string)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

55.2.2 Properties

Name	Description
Inherited from object	
__class__	

55.2.3 Instance Variables

Name	Description
snapshots	list of `Snapshot` making up the trace
type_manager	The `TypeManager` object

56 Module morpher.trace.typemanager

Contains the `TypeManager` class for storing and reconstructing types

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: November 13, 2011

56.1 Variables

Name	Description
__package__	**Value:** `'morpher.trace'`

56.2 Class TypeManager

object
└
morpher.trace.typemanager.TypeManager

Stores type information and can be serialized and reconstructed

Maintains a map of format strings to classes representing the equivalent type. Format strings match the same types used in the *struct* module and can also specify a numeric ID of a user-defined type. The information for user-defined types is constructed from a supplied dictionary mapping user type ids to a tuple, where the tuple contains the type ("struct" or "union") and a list of format strings representing each field of that type. This information is used to construct a matching `ctypes` Structure or Union class on-the-fly and add it to the map.

This class is also used to retrieve size and alignment information for any type represented by a format string, by using the associated `ctypes` methods on their representative classes. These operations are memoized to improve performance, but any information that cannot be serialized by the *pickle* module is discarded upon serialization, and the information is reconstructed again after deserialization.

56.2.1 Methods

__init__(*self*, *usertypes*={})

Fills the table mapping format strings to `ctypes` classes with the mappings for the basic types, and stores the information for constructing user-defined types if supplied.

Parameters

`usertypes`: Optional dictionary mapping format strings to pairs of type strings and lists of fields' formats

(type=dictionary of string : (string, string list) pairs)

Overrides: object.__init__

__getstate__(*self*)

The *pickle* system calls this method when dumping. Prevents the type table from being serialized (saves space and time reading in the file)

Return Value

The __dict__ attibute of this object

(type=dictionary)

__setstate__(*self*, *newdict*)

Pickle calls this method when unpickling. Restores the table to the basic state, just like `__init__`

Parameters

`newdict`: The state object unserialized by pickle (__dict__)

(type=dictionary)

getClass(*self*, *mytype*)

Given a format string such as "PPi" or "12", etc, returns a `ctypes` class object corresponding to its type, creating the class on the fly with the stored usertypes data if necessary. The function is memoized for improved performance but the memoization table is deleted when the object is serialized.

Format strings can be of the types defined by the *struct* module, in which case only the first character is used, or the text can represent a number, in which case the usertype with the matching id is used.

Parameters

`mytype`: The format string to translate to a class

(type=string)

Return Value

The `ctypes` class corresponding to the type represented by the given format string

(type=`ctypes` class)

getFormat(*self*, *objclass*)

Performs the reverse of the `getClass` function.

Parameters

`objclass`: The {ctypes} class to translate to a format string

(type=`ctypes` class)

Return Value

The format string matching the supplied class as described in the `getClass` function

(type=string)

getInfo(*self*, *fmt*)

Takes a format string and returns a (size, alignment) tuple describing the size and alignment of the corresponding C type

Parameters

`fmt`: The format string to translate to a class

(type=string)

Return Value

A tuple containing the (size, alignment) of the C type

(type=(integer, integer) tuple)

align(*self*, *address*, *alignment*)

Utility function to align an address by adding padding

Parameters

`address:` The address to align

(type=integer)

`alignment:` The alignment requirement

(type=integer)

Return Value

The original address plus enough padding that it fits the alignment requirement

(type=integer)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

56.2.2 Properties

Name	Description
Inherited from object	
__class__	

56.2.3 Instance Variables

Name	Description
infotable	The memoization table for the `getInfo` method
table	The dictionary mapping format strings to `ctypes` classes
usertypes	The dictionary storing information about user-defined types such as C Structs and Unions

57 Package morpher.utils

57.1 Modules

58 Module morpher.utils.crash_binning

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

58.1 Variables

Name	Description
__package__	**Value:** `'morpher.utils'`

58.2 Class __crash_bin_struct__

58.2.1 Class Variables

Name	Description
exception_module	**Value:** `None`
exception_address	**Value:** `0`
write_violation	**Value:** `0`
violation_address	**Value:** `0`
violation_thread_id	**Value:** `0`
context	**Value:** `None`
context_dump	**Value:** `None`
disasm	**Value:** `None`
disasm_around	**Value:** `[]`
stack_unwind	**Value:** `[]`
seh_unwind	**Value:** `[]`
extra	**Value:** `None`

58.3 Class crash_binning

To Do: Add MySQL import/export.

58.3.1 Methods

__init__(*self*)

record_crash(*self*, *pydbg*, *extra*=`None`)

Given a PyDbg instantiation that at the current time is assumed to have "crashed" (access violation for example) record various details such as the disassemly around the violating address, the ID of the offending thread, the call stack and the SEH unwind. Store the recorded data in an internal dictionary, binning them by the exception address.

Parameters

`pydbg`: Instance of pydbg

(type=pydbg)

`extra`: (Optional, Def=None) Whatever extra data you want to store with this bin

(type=Mixed)

crash_synopsis(*self*, *crash*=`None`)

For the supplied crash, generate and return a report containing the disassemly around the violating address, the ID of the offending thread, the call stack and the SEH unwind. If not crash is specified, then call through to last_crash_synopsis() which returns the same information for the last recorded crash.

Parameters

`crash`: (Optional, def=None) Crash object to generate report on

(type=__crash_bin_struct__)

Return Value

Crash report

(type=String)

See Also: crash_synopsis()

export_file(*self*, *file_name*)

Dump the entire object structure to disk.

Parameters

`file_name`: File name to export to

(type=String)

Return Value

self

(type=crash_binning)

See Also: import_file()

import_file(*self*, *file_name*)

Load the entire object structure from disk.

Parameters

`file_name`: File name to import from

(type=String)

Return Value

self

(type=crash_binning)

See Also: export_file()

last_crash_synopsis(*self*)

For the last recorded crash, generate and return a report containing the disassemly around the violating address, the ID of the offending thread, the call stack and the SEH unwind.

Return Value

Crash report

(type=String)

See Also: crash_synopsis()

58.3.2 Class Variables

Name	Description
bins	**Value:** {}
last_crash	**Value:** `None`
pydbg	**Value:** `None`

59 Module morpher.utils.hooking

Author: Pedram Amini

License: GNU General Public License 2.0 or later

Contact: pedram.amini@gmail.com

Organization: www.openrce.org

59.1 Variables

Name	Description
AF_INET	**Value:** 2
AF_INET6	**Value:** 23
CONTEXT_CONTROL	**Value:** 65537
CONTEXT_DEBUG_REGISTERS	**Value:** 65552
CONTEXT_FULL	**Value:** 65543
CREATE_NEW_CONSOLE	**Value:** 16
CREATE_PROCESS_DEBUG_EVENT	**Value:** 3
CREATE_THREAD_DEBUG_EVENT	**Value:** 2
DBG_CONTINUE	**Value:** 65538
DBG_EXCEPTION_HANDLED	**Value:** 65537
DBG_EXCEPTION_NOT_HANDLED	**Value:** 2147549185
DEBUG_ONLY_THIS_PROCESS	**Value:** 2
DEBUG_PROCESS	**Value:** 1
DEFAULT_MODE	**Value:** 0
EFLAGS_RF	**Value:** 65536
EFLAGS_TRAP	**Value:** 256
ERROR_NO_MORE_FILES	**Value:** 18
EXCEPTION_ACCESS_VIOLATION	**Value:** 3221225477
EXCEPTION_BREAKPOINT	**Value:** 2147483651
EXCEPTION_DEBUG_EVENT	**Value:** 1

continued on next page

Name	Description
EXCEPTION_GUARD_PAGE	**Value:** 2147483649
EXCEPTION_SINGLE_STEP	**Value:** 2147483652
EXIT_PROCESS_DEBUG_EVENT	**Value:** 5
EXIT_THREAD_DEBUG_EVENT	**Value:** 4
FILE_MAP_READ	**Value:** 4
FORMAT_MESSAGE_ALLOCATE_BUFFER	**Value:** 256
FORMAT_MESSAGE_FROM_SYSTEM	**Value:** 4096
GetLastError	**Value:** <_FuncPtr object at 0x0253AA08>
HW_ACCESS	**Value:** 3
HW_EXECUTE	**Value:** 0
HW_WRITE	**Value:** 1
INVALID_HANDLE_VALUE	**Value:** 4294967295
LOAD_DLL_DEBUG_EVENT	**Value:** 6
MEM_COMMIT	**Value:** 4096
MEM_DECOMMIT	**Value:** 16384
MEM_IMAGE	**Value:** 16777216
MEM_RELEASE	**Value:** 32768
MIB_TCP_STATE_LISTEN	**Value:** 2
OUTPUT_DEBUG_STRING_EVENT	**Value:** 8
PAGE_EXECUTE	**Value:** 16
PAGE_EXECUTE_READ	**Value:** 32
PAGE_EXECUTE_READWRITE	**Value:** 64
PAGE_EXECUTE_WRITECOPY	**Value:** 128
PAGE_GUARD	**Value:** 256
PAGE_NOACCESS	**Value:** 1
PAGE_NOCACHE	**Value:** 512
PAGE_READONLY	**Value:** 2
PAGE_READWRITE	**Value:** 4
PAGE_WRITECOMBINE	**Value:** 1024

continued on next page

Name	Description
PAGE_WRITECOPY	**Value:** 8
PROCESS_ALL_ACCESS	**Value:** 2035711
RIP_EVENT	**Value:** 9
RTLD_GLOBAL	**Value:** 0
RTLD_LOCAL	**Value:** 0
SE_PRIVILEGE_ENABLED	**Value:** 2
SW_SHOW	**Value:** 5
SysDbgReadMsr	**Value:** 16
SysDbgWriteMsr	**Value:** 17
TCP_TABLE_OWNER_PID_ALL	**Value:** 5
TH32CS_INHERIT	**Value:** 2147483648
TH32CS_SNAPALL	**Value:** 15
TH32CS_SNAPHEAPLIST	**Value:** 1
TH32CS_SNAPMODULE	**Value:** 8
TH32CS_SNAPPROCESS	**Value:** 2
TH32CS_SNAPTHREAD	**Value:** 4
THREAD_ALL_ACCESS	**Value:** 2032639
TOKEN_ADJUST_PRIVILEGES	**Value:** 32
UDP_TABLE_OWNER_PID	**Value:** 1
UNLOAD_DLL_DEBUG_EVENT	**Value:** 7
USER_CALLBACK_DEBUG_EVENT	**Value:** 3735928559
VIRTUAL_MEM	**Value:** 12288
__package__	**Value:** 'morpher.utils'
c_types	**Value:** (<type '_ctypes.Structure'>, <class 'ctypes.c_char'>, <cl...
cdll	**Value:** <ctypes.LibraryLoader object at 0x0249FB90>
memmove	**Value:** <CFunctionType object at 0x0253AA80>
memset	**Value:** <CFunctionType object at 0x0253AAF8>
oledll	**Value:** <ctypes.LibraryLoader object at 0x0249FC30>
pydll	**Value:** <ctypes.LibraryLoader object at 0x0249FBB0>

continued on next page

Name	Description
pythonapi	**Value:** <PyDLL 'python dll', handle 1e000000 at 249fbd0>
windll	**Value:** <ctypes.LibraryLoader object at 0x0249FBF0>

59.2 Class hook_container

The purpose of this class is to provide an easy interface for hooking the entry and return points of arbitrary API calls. The hooking of one or both of the points is optional. Example usage:

```
def CreateFileA_on_entry (dbg, args):
    pass

def CreateFileA_on_return (dbg, args, return_value):
    pass

h = hooks(dbg)
h.add(dbg.func_resolve("kernel32", "CreateFileA"), 7, CreateFileA_on_entry, CreateFile
```

This class transparently takes care of various thread-related race conditions.

59.2.1 Methods

__init__(*self*)

add(*self*, *pydbg*, *address*, *num_args*, *entry_hook*=`None`, *exit_hook*=`None`)

Add a new hook on the specified API which accepts the specified number of arguments. Optionally specify callback functions for hooked API entry / exit events. The entry / exit callback prototypes are:

```
entry(dbg, args)
```

Where entry receives the active PyDbg instance as well as a list of the arguments passed to the hooked routine:

```
exit (dbg, args, return_value)
```

Where exit received the active PyDbg instance, a list of the arguments passed to the hooked routine and the return value from the hooked routine.

Parameters

`pydbg:` PyDbg Instance

(type=PyDbg Instance)

`address:` Address of function to hook

(type=Long)

`num_args:` (Optional, Def=0) Number of arguments in function to hook

(type=Integer)

`entry_hook:` (Optional, Def=None) Function to call on hooked API entry

(type=Function Pointer)

`exit_hook:` (Optional, Def=None) Function to call on hooked API exit

(type=Function Pointer)

Return Value

Self

(type=hooks)

remove(*self*, *pydbg*, *address*)

De-activate and remove the hook from the specified API address.

Parameters

`pydbg:` PyDbg Instance

(type=PyDbg Instance)

`address:` Address of function to remove hook from

(type=Long)

Return Value

Self

(type=hooks)

iterate(*self*, *address*)

A simple iterator function that can be used to iterate through all hooks.
Yielded objects are of type hook().

Return Value

Iterated hook entries.

(type=hook)

59.2.2 Class Variables

Name	Description
hooks	**Value:** {}

59.3 Class hook

This helper class abstracts the activation/deactivation of individual hooks. The class is responsible for maintaining the various state variables requires to prevent race conditions.

59.3.1 Methods

__init__(*self*, *address*, *num_args*, *entry_hook*=`None`, *exit_hook*=`None`)

Initialize the object with the specified parameters.

Parameters

`address`: Address of function to hook

(type=Long)

`num_args`: (Optional, Def=0) Number of arguments in function to hook

(type=Integer)

`entry_hook`: (Optional, def=None) Function to call on hooked API entry

(type=Function Pointer)

`exit_hook`: (Optional, def=None) Function to call on hooked API exit

(type=Function Pointer)

hook(*self*, *pydbg*)

Activate the hook by setting a breakpoint on the previously specified address. Breakpoint callbacks are proxied through an internal routine that determines and passes further needed information such as function arguments and return value.

Parameters

`pydbg`: PyDbg Instance

(type=PyDbg Instance)

unhook(*self*, *pydbg*)

De-activate the hook by by removing the breakpoint on the previously specified address.

Parameters

`pydbg`: PyDbg Instance

(type=PyDbg Instance)

59.3.2 Class Variables

Name	Description
hooks	**Value:** `None`

continued on next page

Name	Description
address	**Value:** `0`
num_args	**Value:** `0`
entry_hook	**Value:** `None`
exit_hook	**Value:** `None`
arguments	**Value:** {}
exit_bps	**Value:** {}

60 Module morpher.utils.injection

Author: Justin Seitz

License: GNU General Public License 2.0 or later

Contact: jms@bughunter.ca

Organization: www.openrce.org

60.1 Variables

Name	Description
kernel32	**Value:** <WinDLL 'kernel32', handle 76a70000 at 249fc70>
AF_INET	**Value:** 2
AF_INET6	**Value:** 23
CONTEXT_CONTROL	**Value:** 65537
CONTEXT_DEBUG_REGISTERS	**Value:** 65552
CONTEXT_FULL	**Value:** 65543
CREATE_NEW_CONSOLE	**Value:** 16
CREATE_PROCESS_DEBUG_EVENT	**Value:** 3
CREATE_THREAD_DEBUG_EVENT	**Value:** 2
DBG_CONTINUE	**Value:** 65538
DBG_EXCEPTION_HANDLED	**Value:** 65537
DBG_EXCEPTION_NOT_HANDLED	**Value:** 2147549185
DEBUG_ONLY_THIS_PROCESS	**Value:** 2
DEBUG_PROCESS	**Value:** 1
DEFAULT_MODE	**Value:** 0
EFLAGS_RF	**Value:** 65536
EFLAGS_TRAP	**Value:** 256
ERROR_NO_MORE_FILES	**Value:** 18
EXCEPTION_ACCESS_VIOLATION	**Value:** 3221225477
EXCEPTION_BREAKPOINT	**Value:** 2147483651

continued on next page

Name	Description
EXCEPTION_DEBUG_EVENT	**Value:** 1
EXCEPTION_GUARD_PAGE	**Value:** 2147483649
EXCEPTION_SINGLE_STEP	**Value:** 2147483652
EXIT_PROCESS_DEBUG_EVENT	**Value:** 5
EXIT_THREAD_DEBUG_EVENT	**Value:** 4
FILE_MAP_READ	**Value:** 4
FORMAT_MESSAGE_ALLOCATE_BUFFER	**Value:** 256
FORMAT_MESSAGE_FROM_SYSTEM	**Value:** 4096
GetLastError	**Value:** <_FuncPtr object at 0x0253AA08>
HW_ACCESS	**Value:** 3
HW_EXECUTE	**Value:** 0
HW_WRITE	**Value:** 1
INVALID_HANDLE_VALUE	**Value:** 4294967295
LOAD_DLL_DEBUG_EVENT	**Value:** 6
MEM_COMMIT	**Value:** 4096
MEM_DECOMMIT	**Value:** 16384
MEM_IMAGE	**Value:** 16777216
MEM_RELEASE	**Value:** 32768
MIB_TCP_STATE_LISTEN	**Value:** 2
OUTPUT_DEBUG_STRING_EVENT	**Value:** 8
PAGE_EXECUTE	**Value:** 16
PAGE_EXECUTE_READ	**Value:** 32
PAGE_EXECUTE_READWRITE	**Value:** 64
PAGE_EXECUTE_WRITECOPY	**Value:** 128
PAGE_GUARD	**Value:** 256
PAGE_NOACCESS	**Value:** 1
PAGE_NOCACHE	**Value:** 512
PAGE_READONLY	**Value:** 2
PAGE_READWRITE	**Value:** 4

continued on next page

Name	Description
PAGE_WRITECOMBIN-E	**Value:** 1024
PAGE_WRITECOPY	**Value:** 8
PROCESS_ALL_ACCESS	**Value:** 2035711
RIP_EVENT	**Value:** 9
RTLD_GLOBAL	**Value:** 0
RTLD_LOCAL	**Value:** 0
SE_PRIVILEGE_ENABL-ED	**Value:** 2
SW_SHOW	**Value:** 5
SysDbgReadMsr	**Value:** 16
SysDbgWriteMsr	**Value:** 17
TCP_TABLE_OWNER_P-ID_ALL	**Value:** 5
TH32CS_INHERIT	**Value:** 2147483648
TH32CS_SNAPALL	**Value:** 15
TH32CS_SNAPHEAPLIS-T	**Value:** 1
TH32CS_SNAPMODULE	**Value:** 8
TH32CS_SNAPPROCESS	**Value:** 2
TH32CS_SNAPTHREAD	**Value:** 4
THREAD_ALL_ACCESS	**Value:** 2032639
TOKEN_ADJUST_PRIVI-LEGES	**Value:** 32
UDP_TABLE_OWNER_P-ID	**Value:** 1
UNLOAD_DLL_DEBUG_-EVENT	**Value:** 7
USER_CALLBACK_DEB-UG_EVENT	**Value:** 3735928559
VIRTUAL_MEM	**Value:** 12288
__package__	**Value:** 'morpher.utils'
advapi32	**Value:** <WinDLL 'advapi32', handle 77460000 at 23d6570>
c_types	**Value:** (<type '_ctypes.Structure'>, <class 'ctypes.c_char'>, <cl...
cdll	**Value:** <ctypes.LibraryLoader object at 0x0249FB90>
iphlpapi	**Value:** <WinDLL 'iphlpapi', handle 73820000 at 23dbbb0>
memmove	**Value:** <CFunctionType object at 0x0253AA80>

continued on next page

Name	Description
memset	**Value:** <CFunctionType object at 0x0253AAF8>
ntdll	**Value:** <WinDLL 'ntdll', handle 77a40000 at 23d6e50>
oledll	**Value:** <ctypes.LibraryLoader object at 0x0249FC30>
psapi	**Value:** <WinDLL 'psapi', handle 76cb0000 at 23ed450>
pydll	**Value:** <ctypes.LibraryLoader object at 0x0249FBB0>
pythonapi	**Value:** <PyDLL 'python dll', handle 1e000000 at 249fbd0>
windll	**Value:** <ctypes.LibraryLoader object at 0x0249FBF0>

60.2 Class inject

This class abstracts the ability to inject and eject a DLL into a remote process.

60.2.1 Methods

__init__(*self*)

inject_dll(*self*, *dll_path*, *pid*)

Inject a DLL of your choice into a running process.

Parameters

`dll_path`: The path to the DLL you wish to inject
(type=String)

`pid`: The process ID that you wish to inject into
(type=Integer)

Raises

`pdx` An exception is raised on failure.

eject_dll(*self*, *dll_name*, *pid*)

Eject a loaded DLL from a running process.

Parameters

`dll_name`: The name of the DLL you wish to eject
(type=String)

`pid`: The process ID that you want to eject a DLL from
(type=Integer)

Raises

`pdx` An exception is raised on failure.

get_module_info(*self*, *dll_name*, *pid*)

Helper function to retrieve the necessary information for the DLL we wish to eject.

Parameters

`dll_name`: The name of the DLL you wish to eject
(type=String)

`pid`: The process ID that you want to eject a DLL from
(type=Integer)

Raises

`pdx` An exception is raised on failure.

61 Module run

Contains a script that serves as a command-line interface to the `Morpher` fuzzing tool.

`Morpher` is implemented as an object that is mostly controlled by the contents of a configuration file, but it can have some of the config file overriden by options specified when the `Morpher` object is instantiated. This script parses those options from the command line, passes them to the `Morpher` object it instantiates, and sets the entire fuzzing process running. In addition, this script can start a `Trace` replay process useful for analyzing the results of a fuzzing run instead of starting the `Morpher` process.

Author: Rob Waaser

Contact: robwaaser@gmail.com

Organization: Carnegie Mellon University

Since: October 25, 2011

61.1 Functions

playback(*filename*)

Can play back a trace manually, allowing the user to attach a debugger and step through the trace at their own leisure.

This function was developed to aid a reverse engineer after they have run `Morpher` and wish to investigate the reported crashs and hangs in more detail. `Morpher` stores a copy of the `Trace` file that caused a crash or hang along with the crash report. This function takes the trace file and replays the `Snapshot`s one at a time, reporting the PID so the engineer can attach a debugger of his own and follow along.

Parameters

`filename`: The path to the `Trace` file to be replayed

(type=string)

61.2 Variables

Name	Description
__package__	**Value:** `None`

Index