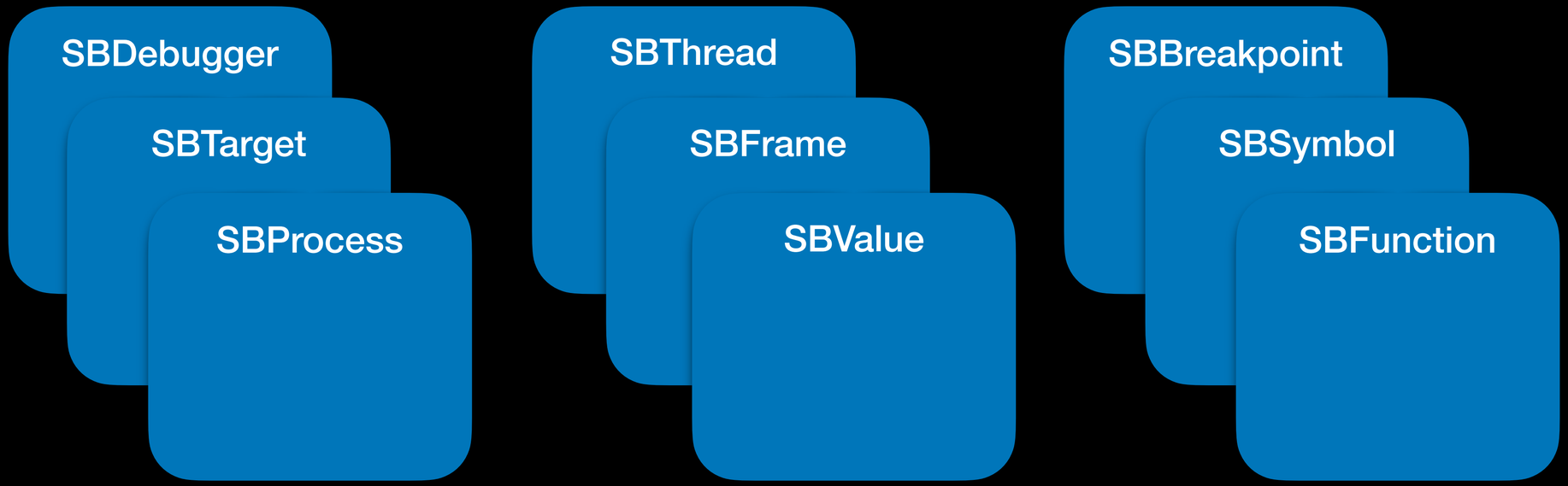
Extending LLDB to More Scripting Languages

Jonas Devlieghere, Apple

C++ API

Scripting Bridge API

Scripting Bridge API



Python

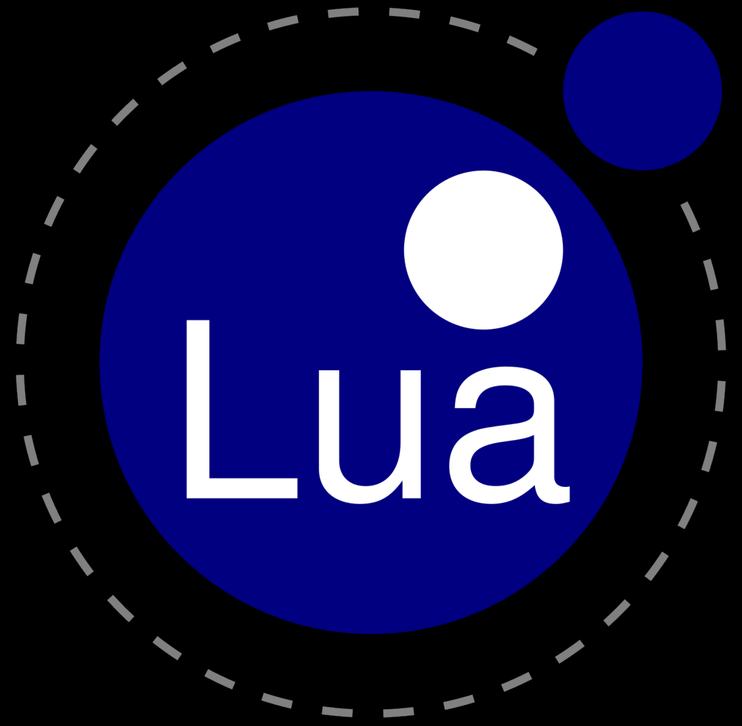
Python

```
#!/usr/bin/python
import lldb
dbg = lldb.SBDebugger.Create()
target = dbg.CreateTarget("a.out")
target.BreakpointCreateByName("foo")
process = target.LaunchSimple(...)
```

```
$ lldb
(lldb) script
>>> frame = lldb.frame
>>> print(frame.IsInlined())
False
(lldb) command script import foo.py
(lldb) foo
I'm a Python script!
```

Another Scripting Language

Another Scripting Language



Script Interpreter

- Execute a single line of code
- Run an interactive interpreter
- Load a module

```
(lldb) script
>>> io.stdout:write("Hello, World!\n")
Hello, World!
(lldb) command script import foo.lua
(lldb) foo
I'm a Lua script!
```

SWIG

SWIG

- C#
- D
- Go
- Guile
- Java
- JavaScript
- Lua
- Racket
- OCaml
- Octave
- Perl
- PHP
- Python
- R
- Ruby
- Scilab
- Tcl/Tk

Language Bindings

Language Bindings

```
(lldb) script --language python
>>> frame = lldb.frame
>>> print(frame.IsInlined())
False
```

```
(lldb) script --language lua
>>> frame = lldb.frame
>>> print(frame:IsInlined())
false
```

Future Work

- Breakpoint/Watchpoint callbacks
- Lua Type Summaries

Conclusion

- Designed for multiple scripting languages
- Stable SB API
- Generated language bindings