

**Macoun**

# lldb - Debugger auf Abwegen

Oliver Bayer  
inovex GmbH

**Disclaimer: Nur weil es technisch  
geht, muss es noch lange nicht  
sinnvoll sein!**

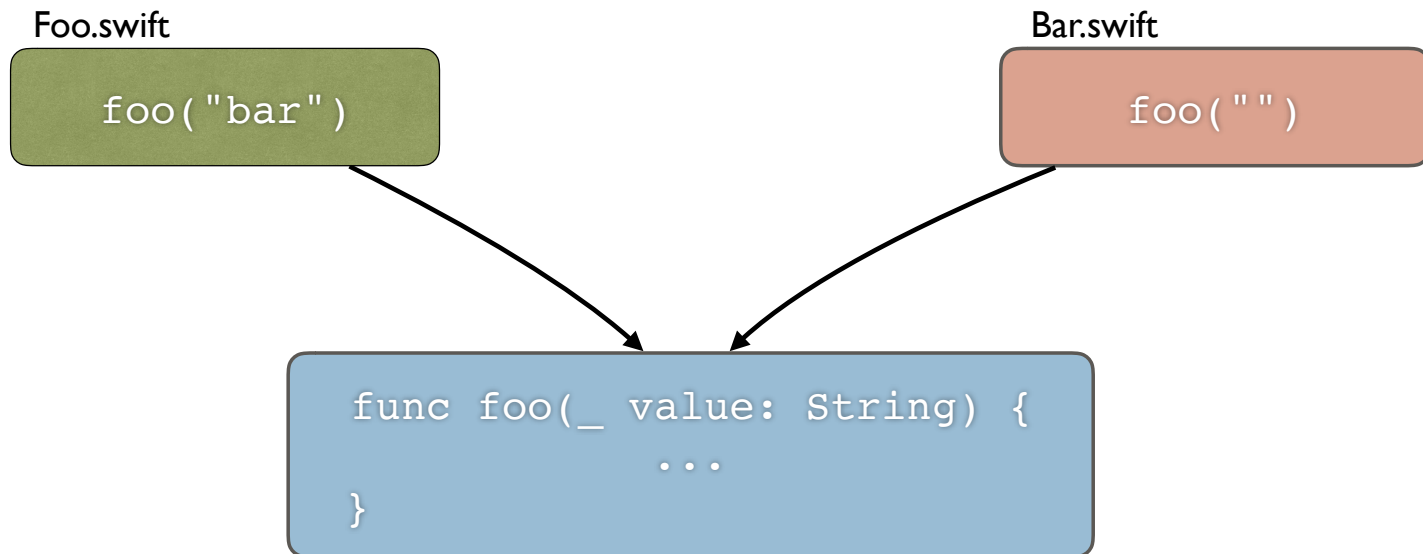
# Ablauf

- Breakpoints per CLI
- Metabreakpoints
- Kontrollfluss Manipulation / Demo

# Breakpoints

- Software Breakpoint
- Symbolic Breakpoint
- Non-Symbolic Breakpoint

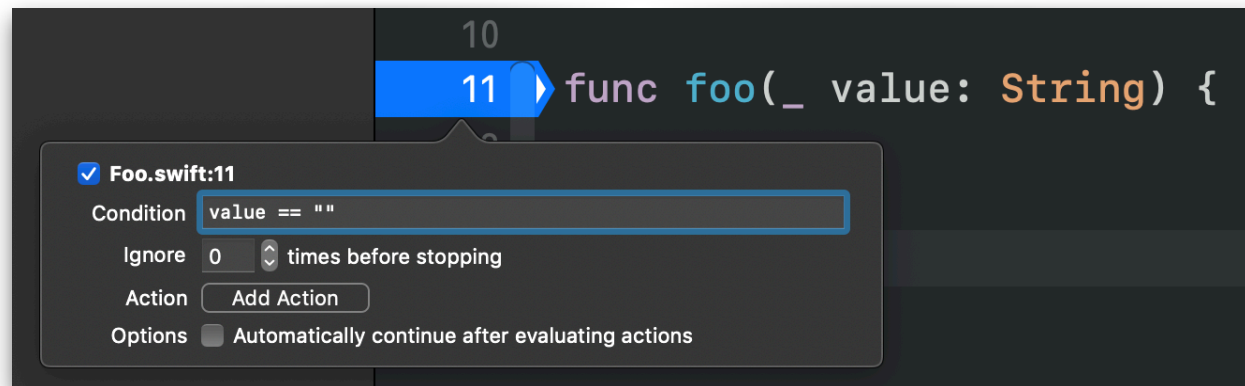
# Conditional Breakpoints



# Conditional Breakpoints

```
8  
9  import Foundation  
10  
11 func foo(_ value: String) {  
12  
13 }  
14
```

# Conditional Breakpoints





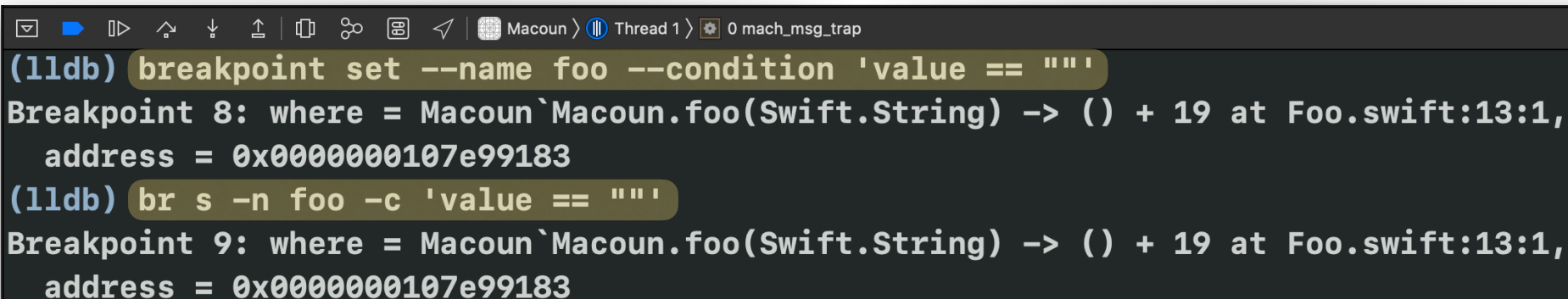
# Conditional Breakpoints

## Non-Symbolic Breakpoint

```
Macoun > Thread 1 > 0 mach_msg_trap  
(lldb) breakpoint set --file Foo.swift --line 11 --condition 'value == ""'  
Breakpoint 6: where = Macoun`Macoun.foo(Swift.String) -> () + 19 at Foo.swift:13:1,  
address = 0x0000000107e99183  
(lldb) br s -f Foo.swift -l 11 -c 'value == ""'  
Breakpoint 7: where = Macoun`Macoun.foo(Swift.String) -> () + 19 at Foo.swift:13:1,  
address = 0x0000000107e99183
```

# Conditional Breakpoints

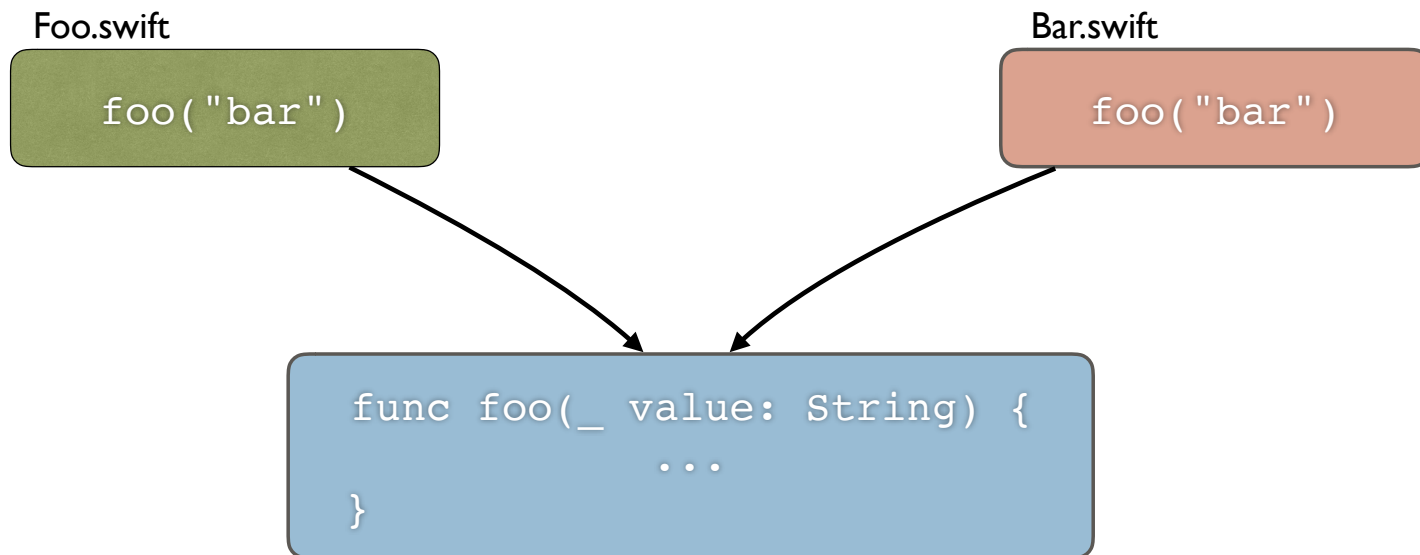
## Symbolic Breakpoint



The screenshot shows a debugger window with a dark theme. The title bar at the top contains several icons and text: a dropdown menu, a blue play button, a home icon, a downward arrow, an upward arrow, a document icon, a magnifying glass, and the text 'Macoun > Thread 1 > 0 mach\_msg\_trap'. Below the title bar, the first command is '(lldb) breakpoint set --name foo --condition 'value == '''. This is followed by the output 'Breakpoint 8: where = Macoun`Macoun.foo(Swift.String) -> () + 19 at Foo.swift:13:1, address = 0x0000000107e99183'. The second command is '(lldb) br s -n foo -c 'value == '''. This is followed by the output 'Breakpoint 9: where = Macoun`Macoun.foo(Swift.String) -> () + 19 at Foo.swift:13:1, address = 0x0000000107e99183'.

```
(lldb) breakpoint set --name foo --condition 'value == ''
Breakpoint 8: where = Macoun`Macoun.foo(Swift.String) -> () + 19 at Foo.swift:13:1,
address = 0x0000000107e99183
(lldb) br s -n foo -c 'value == ''
Breakpoint 9: where = Macoun`Macoun.foo(Swift.String) -> () + 19 at Foo.swift:13:1,
address = 0x0000000107e99183
```

# Metabreakpoints



# Metabreakpoints

```
br s -n foo -N macoun -d
```

Bar.swift

```
foo("bar")
```

```
func foo(_ value: String) {  
    ...  
}
```

# Metabreakpoints

```
br set -n foo -N macoun -d  
br s -f Bar.swift -l 42 -C "br en macoun" -G  
true
```

Bar.swift

foo("bar")



```
func foo(_ value: String) {  
    ...  
}
```

# Metabreakpoints

```
func seiteneffekteFuer100() {  
  print("1")  
  do(  
    print("2")  
    something()  
    print("3")  
    forMe()  
  }  
}
```

# Metabreakpoints

```
func seiteneffekteFuer100() {  
    do()  
    something()  
    forMe()  
}
```

```
br s -p . -X seiteneffekteFuer100 -f Foo.swift -G true  
br command add -s python  
print("Line: {}".format(frame.GetLineEntry().GetLine()))  
DONE
```

# Metabreakpoints

```
func seiteneffekteFuer100() {  
    do()  
    something()  
    forMe()  
}
```

```
br s -p . -X seiteneffekteFuer100 -f Foo.swift -G true  
br command add -s python  
print("Line: {}".format(frame.GetLineEntry().GetLine()))  
DONE
```



# Metabreakpoints

```
func seiteneffekteFuer100() {  
    do()  
    something()  
    forMe()  
}
```

```
br s -p . -X seiteneffekteFuer100 -f Foo.swift -G true  
br command add -s python  
print("Line: {}".format(frame.GetLineEntry().GetLine()))  
DONE
```

# Metabreakpoints

```
func seiteneffekteFuer100() {  
    do()  
    something()  
    forMe()  
}
```

```
br s -p . -X seiteneffekteFuer100 -f Foo.swift -G true  
br command add -s python  
print("Line: {}".format(frame.GetLineEntry().GetLine()))  
DONE
```

# Metabreakpoints

```
func do() {  
  ...  
  let player = AVPlayer(playerItem: item)  
  ...  
  player.play()  
  ...  
}
```

# Metabreakpoints

```
func do() {  
    ...  
    let player = AVPlayer(playerItem: item)  
    ...  
}
```

```
br s -f Player.swift -l 4 -C "call player.isMuted = false"  
-G true
```

# Metabreakpoints

```
func do() {  
    ...  
    let player = AVPlayer(playerItem: item)  
    // Hier der Breakpoint  
    ...  
}
```

```
br s -f Player.swift -p "// Hier der Breakpoint" -X do  
-C "call player.isMuted = true" -G true
```

# Metabreakpoints

- Auf dem Stack<sup>1</sup> steht die Rücksprungadresse des Aufrufenden
- Nach dem Rücksprung steht die Adresse der *AVPlayer* Instanz in Register *rax*<sup>1</sup>

[1] x86 Architekturen

# Metabreakpoints

```
br s -S "-[AVPlayer init]" -K false  
br command add  
settings set target.language objc  
br s -a `(unsigned long)*(unsigned long *)$rsp` -o true  
-C "exp -l objc -- [(AVPlayer *)$rax setMuted: @YES]" -G  
true  
settings set target.language swift  
continue
```

# Metabreakpoints

```
br s -S "-[AVPlayer init]" -K false
br command add
settings set target.language objc
br s -a `(unsigned long)*(unsigned long *)$rsp` -o true
-C "exp -l objc -- [(AVPlayer *)$rax setMuted: @YES]" -G
true
settings set target.language swift
continue
```



# Metabreakpoints

```
br s -S "-[AVPlayer init]" -K false  
br command add  
settings set target.language objc  
br s -a `(unsigned long)*(unsigned long *)$rsp` -o true  
-C "exp -l objc -- [(AVPlayer *)$rax setMuted: @YES]" -G  
true  
settings set target.language swift  
continue
```

# Metabreakpoints

```
br s -S "-[AVPlayer init]" -K false
br command add
settings set target.language objc
br s -a `(unsigned long)*(unsigned long *)$rsp` -o true
-C "exp -l objc -- [(AVPlayer *)$rax setMuted: @YES]" -G
true
settings set target.language swift
continue
```

# Metabreakpoints

```
br s -S "-[AVPlayer init]" -K false
br command add
settings set target.language objc
br s -a `(unsigned long)*(unsigned long *)$rsp` -o true
-C "exp -l objc -- [(AVPlayer *)$rax setMuted: @YES]" -G
true
settings set target.language swift
continue
```

# Metabreakpoints

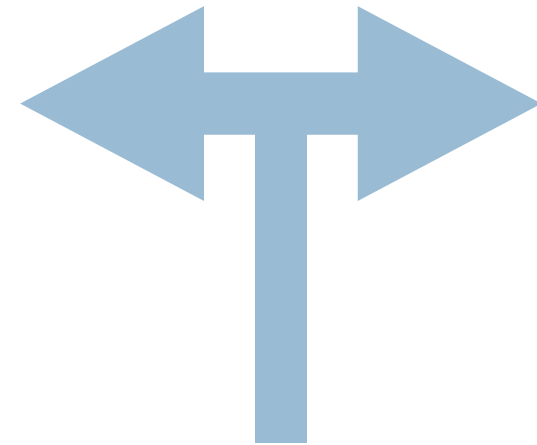
```
br s -S "-[AVPlayer init]" -K false  
br command add  
settings set target.language objc  
br s -a `(unsigned long)*(unsigned long *)$rsp` -o true  
-C "exp -l objc -- [(AVPlayer *)$rax setMuted: @YES]" -G  
true  
settings set target.language swift  
continue
```

# Metabreakpoints

```
br s -S "-[AVPlayer init]" -K false
br command add
settings set target.language objc
br s -a `(unsigned long)*(unsigned long *)$rsp` -o true
-C "exp -l objc -- [(AVPlayer *)$rax setMuted: @YES]" -G
true
settings set target.language swift
continue
```

# Neue Wege gehen mit *thread*

- Codepfade zur Laufzeit manipulieren
- Methoden kurzschließen
- Rückgabewerte ändern



Demo

# Neue Wege gehen mit *thread*

- Nachteil an *thread return* et al. → bremst das Ausführen des Codes aus
- Idee: (Binary) Patch → genauso schnell wie vorher



# Binary Patch Breakpoint

```
private func isExpectedInput(_ value: String) -> Bool {  
    if value.lowercased() == secret {  
        return true  
    } else {  
        return false  
    }  
}
```

# Binary Patch Breakpoint

```
private func isExpectedInput(_ value: String) -> Bool {  
    return true  
}
```

# Binary Patch Breakpoint

```
push    rbp
mov     rbp, rsp
push    0x1
pop     rax
pop     rbp
ret
```

```
'\x55\x6a\x01\x58\x5d\xc3'
```

# Binary Patch Breakpoint

```
br s -n isExpectedInput -o true -K false
br command add -s python
patch = '\x55\xB8\x01\x00\x00\x00\x5d\xc3'
addr = bp_loc.GetLoadAddress()
error = lldb.SBError()
proc = frame.GetThread().GetProcess()
result = proc.WriteMemory(addr, patch, error)
proc.Continue()
DONE
```

# Binary Patch Breakpoint

```
br s -n isExpectedInput -o true -K false
br command add -s python
patch = '\x55\xB8\x01\x00\x00\x00\x5d\xc3'
addr = bp_loc.GetLoadAddress()
error = lldb.SBError()
proc = frame.GetThread().GetProcess()
result = proc.WriteMemory(addr, patch, error)
proc.Continue()
DONE
```

# Binary Patch Breakpoint

```
br s -n isExpectedInput -o true -K false
br command add -s python
patch = '\x55\xB8\x01\x00\x00\x00\x5d\xc3'
addr = bp_loc.GetLoadAddress()
error = lldb.SBError()
proc = frame.GetThread().GetProcess()
result = proc.WriteMemory(addr, patch, error)
proc.Continue()
DONE
```

# Binary Patch Breakpoint

```
br s -n isExpectedInput -o true -K false  
br command add -s python  
patch = '\x55\xB8\x01\x00\x00\x00\x5d\xc3'  
addr = bp_loc.GetLoadAddress()  
error = lldb.SBError()  
proc = frame.GetThread().GetProcess()  
result = proc.WriteMemory(addr, patch, error)  
proc.Continue()  
DONE
```

# Binary Patch Breakpoint

```
br s -n isExpectedInput -o true -K false
br command add -s python
patch = '\x55\xB8\x01\x00\x00\x00\x5d\xc3'
addr = bp_loc.GetLoadAddress()
error = lldb.SBError()
proc = frame.GetThread().GetProcess()
result = proc.WriteMemory(addr, patch, error)
proc.Continue()
DONE
```



# Binary Patch Breakpoint

```
br s -n isExpectedInput -o true -K false
br command add -s python
patch = '\x55\xB8\x01\x00\x00\x00\x5d\xc3'
addr = bp_loc.GetLoadAddress()
error = lldb.SBError()
proc = frame.GetThread().GetProcess()
result = proc.WriteMemory(addr, patch, error)
proc.Continue()
DONE
```

# Binary Patch Breakpoint

```
br s -n isExpectedInput -o true -K false  
br command add -s python  
patch = '\x55\xB8\x01\x00\x00\x00\x5d\xc3'  
addr = bp_loc.GetLoadAddress()  
error = lldb.SBError()  
proc = frame.GetThread().GetProcess()  
result = proc.WriteMemory(addr, patch, error)  
proc.Continue()  
DONE
```

# Binary Patch Breakpoint

```
br s -n isExpectedInput -o true -K false  
br command add -s python  
patch = '\x55\xB8\x01\x00\x00\x00\x5d\xc3'  
addr = bp_loc.GetLoadAddress()  
error = lldb.SBError()  
proc = frame.GetThread().GetProcess()  
result = proc.WriteMemory(addr, patch, error)  
proc.Continue()  
DONE
```

# Patch Breakpoint

```
private let secret = "foo" -> "macoun"
```

# Patch Breakpoint

```
private let secret = "foo" // -> "macoun"
```

```
br s -F "Swift.String.init(_builtinStringLiteral:  
Builtin.RawPointer, utf8CodeUnitCount: Builtin.Word,  
isASCII: Builtin.Int1) -> Swift.String" -c "0 ==  
(int)strcmp(\"foo\", (char *)$rdi)" -o true  
br command add  
exp -- char * $value = (char *)"macoun"  
register write $rdi `(unsigned long *)$value`  
register write $rsi 6  
continue  
DONE
```

# Patch Breakpoint

```
private let secret = "foo" // -> "macoun"
```

```
br s -F "Swift.String.init(_builtinStringLiteral:  
Builtin.RawPointer, utf8CodeUnitCount: Builtin.Word,  
isASCII: Builtin.Int1) -> Swift.String" -c "0 ==  
(int)strcmp(\"foo\", (char *)$rdi)" -o true  
br command add  
exp -- char * $value = (char *)"macoun"  
register write $rdi `(unsigned long *)$value`  
register write $rsi 6  
continue  
DONE
```

# Patch Breakpoint

```
private let secret = "foo" // -> "macoun"
```

```
br s -F "Swift.String.init(_builtinStringLiteral:  
Builtin.RawPointer, utf8CodeUnitCount: Builtin.Word,  
isASCII: Builtin.Int1) -> Swift.String" -c "0 ==  
(int)strcmp(\"foo\", (char *)$rdi)" -o true  
br command add  
exp -- char * $value = (char *)"macoun"  
register write $rdi `(unsigned long *)$value`  
register write $rsi 6  
continue  
DONE
```

# Patch Breakpoint

```
private let secret = "foo" // -> "macoun"
```

```
br s -F "Swift.String.init(_builtinStringLiteral:  
Builtin.RawPointer, utf8CodeUnitCount: Builtin.Word,  
isASCII: Builtin.Int1) -> Swift.String" -c "0 ==  
(int)strcmp(\"foo\", (char *)$rdi)" -o true  
br command add  
exp -- char * $value = (char *)"macoun"  
register write $rdi `(unsigned long *)$value`  
register write $rsi 6  
continue  
DONE
```



# Patch Breakpoint

```
private let secret = "foo" // -> "macoun"
```

```
br s -F "Swift.String.init(_builtinStringLiteral:  
Builtin.RawPointer, utf8CodeUnitCount: Builtin.Word,  
isASCII: Builtin.Int1) -> Swift.String" -c "0 ==  
(int)strcmp(\"foo\", (char *)$rdi)" -o true  
br command add  
exp -- char * $value = (char *)"macoun"  
register write $rdi `(unsigned long *)$value`  
register write $rsi 6  
continue  
DONE
```

# Patch Breakpoint

```
private let secret = "foo" // -> "macoun"
```

```
br s -F "Swift.String.init(_builtinStringLiteral:  
Builtin.RawPointer, utf8CodeUnitCount: Builtin.Word,  
isASCII: Builtin.Int1) -> Swift.String" -c "0 ==  
(int)strcmp(\"foo\", (char *)$rdi)" -o true  
br command add  
exp -- char * $value = (char *)"macoun"  
register write $rdi `(unsigned long *)$value`  
register write $rsi 6  
continue  
DONE
```

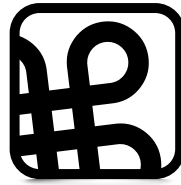
# Schlussworte

- Automatismen
  - <https://github.com/obayer/Trampoline>
  - .lldbinit
  - *command source <filename>*
- für alles andere: *lldb help*

Fragen?

# Vielen Dank

Oliver Bayer  
inovex GmbH



**Macoun**