High-performance logging for the rest of us

CocoaHeads / SLUG July 2024

Saagar Jha

os_log

os_log?

os_log?

High-performance logging for the rest of us

CocoaHeads / SLUG July 2024

Saagar Jha

High-performance logging for the rest of us

CocoaHeads / SLUG July 2024

Saagar Jha

High-performance logging for the rest of us

CocoaHeads / SLUG July 2024

Saagar Jha

High-performance logging for the rest of us

CocoaHeads / SLUG July 2024

Saagar Jha

logging for the rest of us

Logging for the rest of us
What is a log?

- Understand what a program is doing without stopping it
  - As opposed to a debugger
  - …or a crash log
- Meant for human consumption
- General characteristics
  - Ordered
  - Reliable
- I was going to put an alignment chart here but going over it would take too long
  - Find me afterwards if you want it

Logging for the rest of us

By example

```
print("just setting up my twttr")
```

Unstructured output

Logging for the rest of us

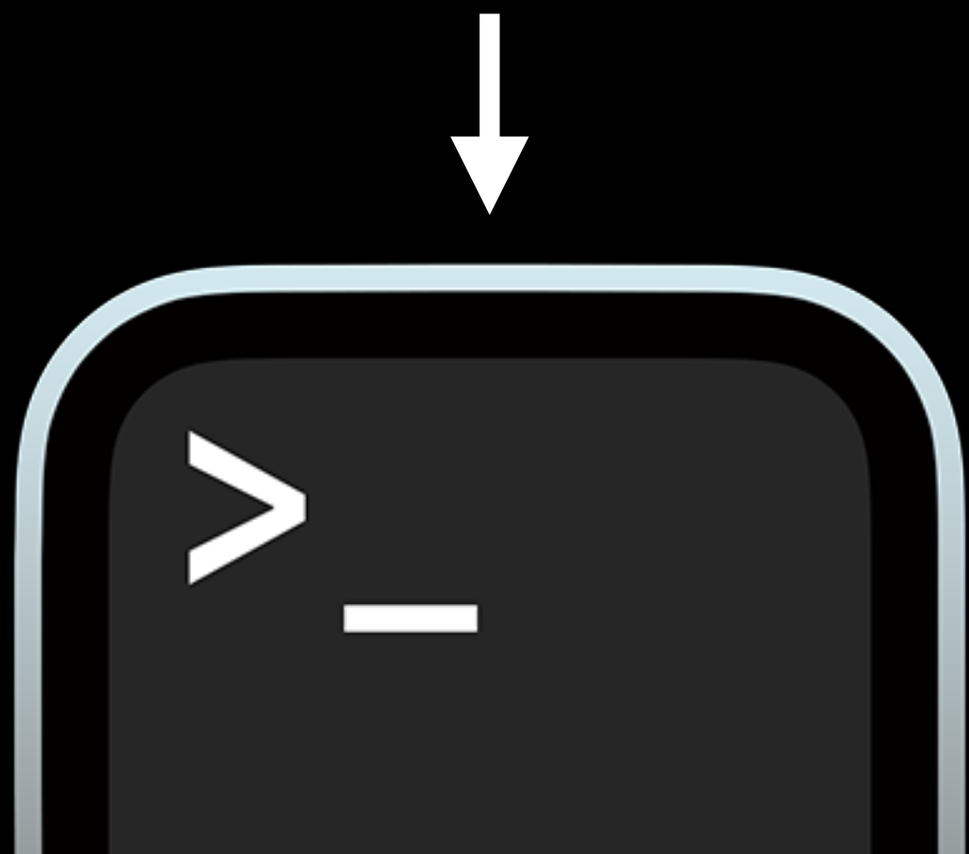
By example

```
print("just setting up my twttr")
```

Logging for the rest of us

By example

```
print("just setting up my twttr")
```



Logging for the rest of us

By example

```
print("just setting up my twttr")
```

Logging for the rest of us

By example

```
NSLog("just setting up my twttr")
```

Logging for the rest of us

By example

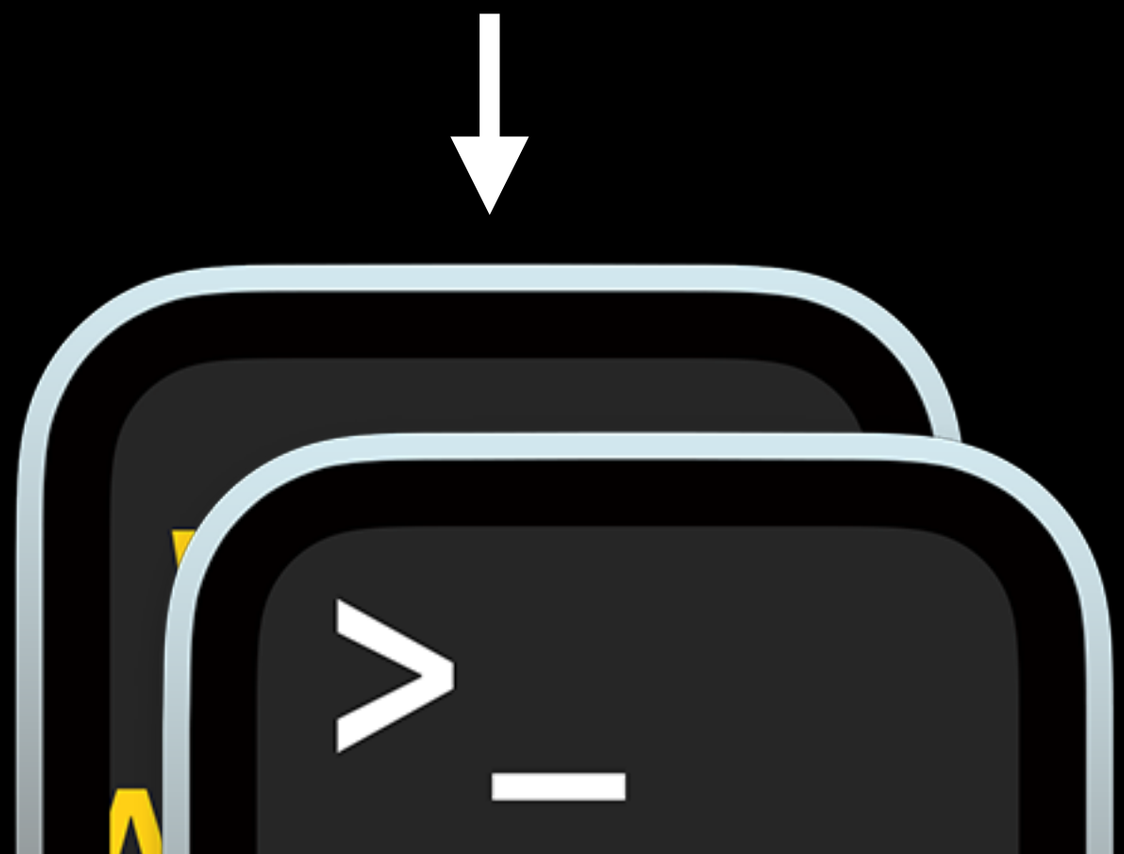
```
NSLog("just setting up my twttr")
```

Logging for the rest of us

By example

Unstructured output

```
NSLog("just setting up my twttr")
```



os_log

os_log
aka “unified logging”

- Subsumes Apple’s disparate logging frameworks
  - Centrally-managed log store (backed by disk or memory)
  - One API across userspace and kernel
- Fast
  - “Free” if turned off
  - Dynamic overhead depending on level of observation
- Structured

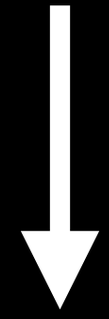
Structured logging

```
printf("My email is %s\n", user->email);
```

Structured logging

I lied this is actually unstructured logging again

```
printf("My email is %s\n", user->email);
```



My email is saagar@saagarjha.com

Structured logging

For real this time

```
AccountLog.info("User ID", account.id());
```

Structured logging

For real this time

```
AccountLog.info("User ID", account.id());
```

```
level: info
message:
- User ID
- 42069
subsystem: account
timestamp: 2007-01-09T17:41:00.000Z
```

Structured logging

For real this time

```
AccountLog.info("User ID", account.id());
```

```
level: info
message:
- User ID
- 42069
subsystem: account
timestamp: [2007-01-09T17:41:00.000Z]
```

os_log internals

os_log internals

- Lives in <os/log.h>
- …
- …let’s just look at it

```c
#include <os/log.h>

int main() {
    os_log_t log = os_log_create(
        "com.saagarjha.test.subsystem",
        "Test Category");
    os_log_debug(
        log,
        "This is a number: %d. This is a string: %s",
        42069,
        "Never gonna give you up");
}
```

```c
#include <os/log.h>

int main() {
    os_log_t log = _os_log_create(
        &__dso_handle,
        "com.saagarjha.test.subsystem",
        "Test Category");
    _os_log_internal(
        &__dso_handle,
        log,
        OS_LOG_TYPE_DEBUG,
        "This is a number: %d. This is a string: %s",
        42069,
        "Never gonna give you up");
}
```

```
import os

let number = 42
let string = "Never gonna give you up"

let log = Logger(
    subsystem: "com.saagarjha.test.subsystem",
    category: "Test Category")
log.log(
    level: .debug,
    "This is a number: \(number). This is a string: \(string)")
```

```
import os

let number = 42
let string = "Never gonna give you up"

let log = Logger(
  subsystem: "com.saagarjha.test.subsystem",
  category: "Test Category")
var interpolation = OSLogInterpolation(
  literalCapacity: /* compiler defined */,
  interpolationCount: 2)
interpolation.appendLiteral("This is a number: ")
interpolation.appendInterpolation(number)
interpolation.appendLiteral(". This is a string: ")
interpolation.appendInterpolation(string)
let message = OSLogMessage(stringInterpolation: interpolation)
log.log(level: .debug, message)
```

test — Binary Ninja 4.1.5619-test

test

Mach-O ▾ Linear ▾ Pseudo C ▾

```
0x1000030e0

1000030e0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
100003100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
100003120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
100003140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
100003160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................

__text (PURE_CODE) section started  {0x10000317c-0x100003bb8}

10000317c  int64_t _main()

10000317c  {
10000319c      void* x0 = type metadata accessor for Logger(0);
1000031b0      ___swift_allocate_value_buffer(x0, &log);
1000031bc      ___swift_project_value_buffer(x0, &log);
1000031fc      Logger.init(subsystem:category:)(-0x2fffffffffffffe4, -0x7fffffffffc2e0, 'Test Cat', 'egory\x00\x00\xed');
100003200      char debug_type_1 = static os_log_type_t.debug.getter();
100003208      os_log_t log = Logger.logObject.getter();
100003210      uint32_t debug_type = ((uint32_t)debug_type_1);
100003210
10000321c      if (_os_log_type_enabled(log, debug_type) != 0) {
100003228          uint8_t* buf = _swift_slowAlloc(0x16, -1);
100003238          int64_t x0_4 = _swift_slowAlloc(0x20, -1);
100003240          int64_t var_38 = x0_4;
10000324c          *(uint32_t*)buf = 0x8000202;
100003254          *(uint64_t*)(buf + 4) = 42;
10000325c          *(uint16_t*)(buf + 0xc) = 0x820;
100003280          *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn:)(-0xd000000000000017, -0x7fffffffffc2c0, &var_38)
1000032a4          __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…", buf, 0x16);
1000032bc          _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
1000032cc          _swift_slowDealloc(x0_4, -1, -1);
1000032dc          _swift_slowDealloc(buf, -1, -1);
10000321c      }
10000321c
1000032e4      _objc_release(log);
1000032fc      return 0;
10000317c  }

100003300  int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)

100003300  {
100003310      void* x9 = *(uint64_t*)((char*)arg1 - 8);
100003314      int32_t x8 = *(uint32_t*)((char*)x9 + 0x50);
100003314
100003318      if ((x8 & 0x20000) == 0)
```

14 mac-aarch64 0x100002dea

st ● +

▾ Linear ▾ Pseudo C ▾

000030e0

```
00030e0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00
```

text (PURE_CODE) section started {0x10000317c-0x100003bb8}

```
000317c  int64_t _main()

000317c  {
000319c      void* x0 = type metadata accessor for Logger(0);
00031b0      ___swift_allocate_value_buffer(x0, &log);
00031bc      ___swift_project_value_buffer(x0, &log);
00031fc      Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat', 'eg
0003200      char debug_type_1 = static os_log_type_t.debug.getter();
0003208      os_log_t log = Logger.logObject.getter();
0003210      uint32_t debug_type = ((uint32_t)debug_type_1);
0003210
000321c      if (_os_log_type_enabled(log, debug_type) != 0) {
0003228          uint8_t* buf = _swift_slowAlloc(0x16, -1);
0003238          int64_t x0_4 = _swift_slowAlloc(0x20, -1);
0003240          int64_t var_38 = x0_4;
000324c          *(uint32_t*)buf = 0x8000202;
0003254          *(uint64_t*)(buf + 4) = 42;
000325c          *(uint16_t*)(buf + 0xc) = 0x820;
0003280          *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn:)(-
00032a4          __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…", b
00032bc          _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
00032cc          _swift_slowDealloc(x0_4, -1, -1);
```

st ● +

▾ Linear ▾ Pseudo C ▾

000030e0

```
00030e0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00
```

text (PURE_CODE) section started {0x10000317c-0x100003bb8}

```
000317c  int64_t _main()

000317c  {
000319c      void* x0 = type metadata accessor for Logger(0);
00031b0      ___swift_allocate_value_buffer(x0, &log);
00031bc      ___swift_project_value_buffer(x0, &log);
00031fc      Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat', 'eg
0003200      char debug_type_1 = static os_log_type_t.debug.getter();
0003208      os_log_t log = Logger.logObject.getter();
0003210      uint32_t debug_type = ((uint32_t)debug_type_1);
0003210
000321c      if (_os_log_type_enabled(log, debug_type) != 0) {
0003228          uint8_t* buf = _swift_slowAlloc(0x16, -1);
0003238          int64_t x0_4 = _swift_slowAlloc(0x20, -1);
0003240          int64_t var_38 = x0_4;
000324c          *(uint32_t*)buf = 0x8000202;
0003254          *(uint64_t*)(buf + 4) = 42;
000325c          *(uint16_t*)(buf + 0xc) = 0x820;
0003280          *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn:)(-
00032a4          __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…", b
00032bc          _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
00032cc          _swift_slowDealloc(x0_4, -1, -1);
```

st ● +

▾ Linear ▾ Pseudo C ▾

000030e0

```
00030e0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00
```

text (PURE_CODE) section started {0x10000317c-0x100003bb8}

```
000317c  int64_t _main()

000317c  {
000319c      void* x0 = type metadata accessor for Logger(0);
00031b0      ___swift_allocate_value_buffer(x0, &log);
00031bc      ___swift_project_value_buffer(x0, &log);
00031fc      Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat', 'eg
0003200      char debug_type_1 = static os_log_type_t.debug.getter();
0003208      os_log_t log = Logger.logObject.getter();
0003210      uint32_t debug_type = ((uint32_t)debug_type_1);
0003210
000321c      if (_os_log_type_enabled(log, debug_type) != 0) {
0003228          uint8_t* buf = _swift_slowAlloc(0x16, -1);
0003238          int64_t x0_4 = _swift_slowAlloc(0x20, -1);
0003240          int64_t var_38 = x0_4;
000324c          *(uint32_t*)buf = 0x8000202;
0003254          *(uint64_t*)(buf + 4) = 42;
000325c          *(uint16_t*)(buf + 0xc) = 0x820;
0003280          *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn:)(-
00032a4          __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…", b
00032bc          _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
00032cc          _swift_slowDealloc(x0_4, -1, -1);
```

st ● +

▾ Linear ▾ Pseudo C ▾

000030e0

```
00030e0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00
```

text (PURE_CODE) section started {0x10000317c-0x100003bb8}

```
000317c  int64_t _main()

000317c  {
000319c      void* x0 = type metadata accessor for Logger(0);
00031b0      ___swift_allocate_value_buffer(x0, &log);
00031bc      ___swift_project_value_buffer(x0, &log);
00031fc      Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat', 'eg
0003200      char debug_type_1 = static os_log_type_t.debug.getter();
0003208      os_log_t log = Logger.logObject.getter();
0003210      uint32_t debug_type = ((uint32_t)debug_type_1);
0003210
000321c      if (_os_log_type_enabled(log, debug_type) != 0) {
0003228          uint8_t* buf = _swift_slowAlloc(0x16, -1);
0003238          int64_t x0_4 = _swift_slowAlloc(0x20, -1);
0003240          int64_t var_38 = x0_4;
000324c          *(uint32_t*)buf = 0x8000202;
0003254          *(uint64_t*)(buf + 4) = 42;
000325c          *(uint16_t*)(buf + 0xc) = 0x820;
0003280          *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn:)(-
00032a4          __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…", b
00032bc          _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
00032cc          _swift_slowDealloc(x0_4, -1, -1);
```

st ● +

▾ Linear ▾ Pseudo C ▾

000030e0

```
00030e0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 0
0003160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00
```

text (PURE_CODE) section started {0x10000317c-0x100003bb8}

```
000317c  int64_t _main()

000317c  {
000319c      void* x0 = type metadata accessor for Logger(0);
00031b0      ___swift_allocate_value_buffer(x0, &log);
00031bc      ___swift_project_value_buffer(x0, &log);
00031fc      Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat', 'eg
0003200      char debug_type_1 = static os_log_type_t.debug.getter();
0003208      os_log_t log = Logger.logObject.getter();
0003210      uint32_t debug_type = ((uint32_t)debug_type_1);
0003210
000321c      if (_os_log_type_enabled(log, debug_type) != 0) {
0003228          uint8_t* buf = _swift_slowAlloc(0x16, -1);
0003238          int64_t x0_4 = _swift_slowAlloc(0x20, -1);
0003240          int64_t var_38 = x0_4;
000324c          *(uint32_t*)buf = 0x8000202;
0003254          *(uint64_t*)(buf + 4) = 42;
000325c          *(uint16_t*)(buf + 0xc) = 0x820;
0003280          *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn:)(-
00032a4          __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…", b
00032bc          _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
00032cc          _swift_slowDealloc(x0_4, -1, -1);
```

```
int64_t _main()

{
    void* x0 = type metadata accessor for Logger(0);
    ___swift_allocate_value_buffer(x0, &log);
    ___swift_project_value_buffer(x0, &log);
    Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat',
    char debug_type_1 = static os_log_type_t.debug.getter();
    os_log_t log = Logger.logObject.getter();
    uint32_t debug_type = ((uint32_t)debug_type_1);

    if (_os_log_type_enabled(log, debug_type) != 0) {
        uint8_t* buf = _swift_slowAlloc(0x16, -1);
        int64_t x0_4 = _swift_slowAlloc(0x20, -1);
        int64_t var_38 = x0_4;
        *(uint32_t*)buf = 0x8000202;
        *(uint64_t*)(buf + 4) = 42;
        *(uint16_t*)(buf + 0xc) = 0x820;
        *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn
        __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…
        _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
        _swift_slowDealloc(x0_4, -1, -1);
        _swift_slowDealloc(buf, -1, -1);
    }

    _objc_release(log);
    return 0;
}
```

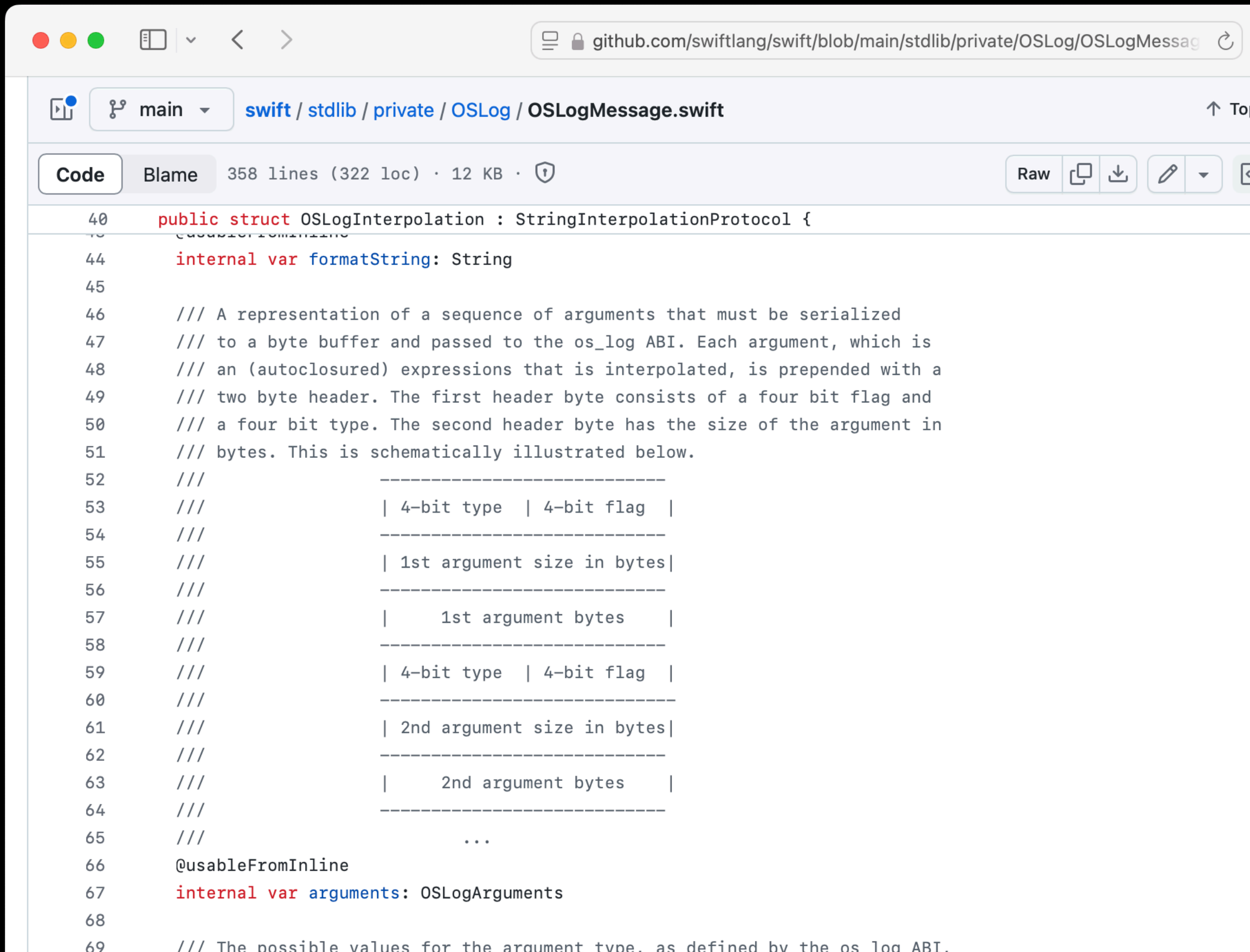
```
int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)
```

```
int64_t _main()

{
    void* x0 = type metadata accessor for Logger(0);
    ___swift_allocate_value_buffer(x0, &log);
    ___swift_project_value_buffer(x0, &log);
    Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat',
    char debug_type_1 = static os_log_type_t.debug.getter();
    os_log_t log = Logger.logObject.getter();
    uint32_t debug_type = ((uint32_t)debug_type_1);

    if (_os_log_type_enabled(log, debug_type) != 0) {
        uint8_t* buf = _swift_slowAlloc(0x16, -1);
        int64_t x0_4 = _swift_slowAlloc(0x20, -1);
        int64_t var_38 = x0_4;
        *(uint32_t*)buf = 0x8000202;
        *(uint64_t*)(buf + 4) = 42;
        *(uint16_t*)(buf + 0xc) = 0x820;
        *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn
        __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…
        _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
        _swift_slowDealloc(x0_4, -1, -1);
        _swift_slowDealloc(buf, -1, -1);
    }

    _objc_release(log);
    return 0;
}

int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)
```

```
int64_t _main()

{
    void* x0 = type metadata accessor for Logger(0);
    ___swift_allocate_value_buffer(x0, &log);
    ___swift_project_value_buffer(x0, &log);
    Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat',
    char debug_type_1 = static os_log_type_t.debug.getter();
    os_log_t log = Logger.logObject.getter();
    uint32_t debug_type = ((uint32_t)debug_type_1);

    if (_os_log_type_enabled(log, debug_type) != 0) {
        uint8_t* buf = _swift_slowAlloc(0x16, -1);
        int64_t x0_4 = _swift_slowAlloc(0x20, -1);
        int64_t var_38 = x0_4;
        *(uint32_t*)buf = 0x8000202;
        *(uint64_t*)(buf + 4) = 42;
        *(uint16_t*)(buf + 0xc) = 0x820;
        *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn
        __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…
        _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
        _swift_slowDealloc(x0_4, -1, -1);
        _swift_slowDealloc(buf, -1, -1);
    }

    _objc_release(log);
    return 0;
}
```

```
int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)
```

```
int64_t _main()

{
    void* x0 = type metadata accessor for Logger(0);
    ___swift_allocate_value_buffer(x0, &log);
    ___swift_project_value_buffer(x0, &log);
    Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat',
    char debug_type_1 = static os_log_type_t.debug.getter();
    os_log_t log = Logger.logObject.getter();
    uint32_t debug_type = ((uint32_t)debug_type_1);

    if (_os_log_type_enabled(log, debug_type) != 0) {
        uint8_t* buf = _swift_slowAlloc(0x16, -1);
        int64_t x0_4 = _swift_slowAlloc(0x20, -1);
        int64_t var_38 = x0_4;
        *(uint32_t*)buf = 0x8000202;
        *(uint64_t*)(buf + 4) = 42;
        *(uint16_t*)(buf + 0xc) = 0x820;
        *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn
        __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…
        _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
        _swift_slowDealloc(x0_4, -1, -1);
        _swift_slowDealloc(buf, -1, -1);
    }

    _objc_release(log);
    return 0;
}

int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)
```



github.com/swiftlang/swift/blob/main/stdlib/private/OSLog/OSLogMessag

main

swift / stdlib / private / OSLog / **OSLogMessage.swift**

To

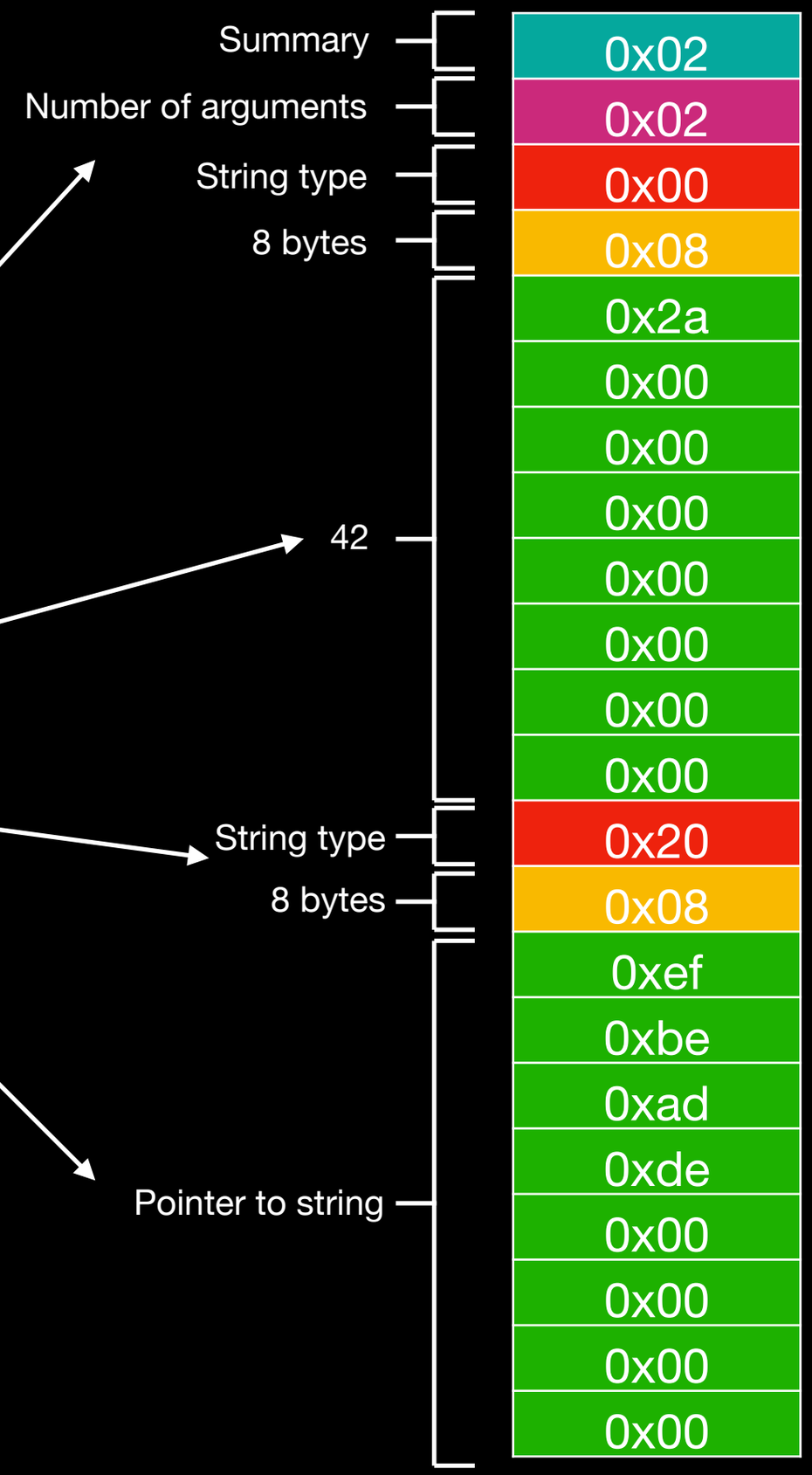
Code | Blame | 358 lines (322 loc) · 12 KB ·

Raw

```
40    public struct OSLogInterpolation : StringInterpolationProtocol {
44      internal var formatString: String
45
46      /// A representation of a sequence of arguments that must be serialized
47      /// to a byte buffer and passed to the os_log ABI. Each argument, which is
48      /// an (autoclosured) expressions that is interpolated, is prepended with a
49      /// two byte header. The first header byte consists of a four bit flag and
50      /// a four bit type. The second header byte has the size of the argument in
51      /// bytes. This is schematically illustrated below.
52      ///                  ----------------------------
53      ///                  | 4-bit type  | 4-bit flag  |
54      ///                  ----------------------------
55      ///                  | 1st argument size in bytes|
56      ///                  ----------------------------
57      ///                  |     1st argument bytes    |
58      ///                  ----------------------------
59      ///                  | 4-bit type  | 4-bit flag  |
60      ///                  ----------------------------
61      ///                  | 2nd argument size in bytes|
62      ///                  ----------------------------
63      ///                  |     2nd argument bytes    |
64      ///                  ----------------------------
65      ///                           ...
66      @usableFromInline
67      internal var arguments: OSLogArguments
68
69      /// The possible values for the argument type, as defined by the os_log ABI.
```

```
int64_t _main()

{
    void* x0 = type metadata accessor for Logger(0);
    ___swift_allocate_value_buffer(x0, &log);
    ___swift_project_value_buffer(x0, &log);
    Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat',
    char debug_type_1 = static os_log_type_t.debug.getter();
    os_log_t log = Logger.logObject.getter();
    uint32_t debug_type = ((uint32_t)debug_type_1);

    if (_os_log_type_enabled(log, debug_type) != 0) {
        uint8_t* buf = _swift_slowAlloc(0x16, -1);
        int64_t x0_4 = _swift_slowAlloc(0x20, -1);
        int64_t var_38 = x0_4;
        *(uint32_t*)buf = 0x8000202;
        *(uint64_t*)(buf + 4) = 42;
        *(uint16_t*)(buf + 0xc) = 0x820;
        *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn
        __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…
        _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
        _swift_slowDealloc(x0_4, -1, -1);
        _swift_slowDealloc(buf, -1, -1);
    }

    _objc_release(log);
    return 0;
}

int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)
```

```
int64_t _main()

{
    void* x0 = type metadata accessor for Logger(0);
    ___swift_allocate_value_buffer(x0, &log);
    ___swift_project_value_buffer(x0, &log);
    Logger.init(subsystem:category:)(-0x2fffffffffff
    char debug_type_1 = static os_log_type_t.debug.ge
    os_log_t log = Logger.logObject.getter();
    uint32_t debug_type = ((uint32_t)debug_type_1);

    if (_os_log_type_enabled(log, debug_type) != 0) {
        uint8_t* buf = _swift_slowAlloc(0x16, -1);
        int64_t x0_4 = _swift_slowAlloc(0x20, -1);
        int64_t var_38 = x0_4;
        *(uint32_t*)buf = 0x8000202;
        *(uint64_t*)(buf + 4) = 42;
        *(uint16_t*)(buf + 0xc) = 0x820;
        *(uint64_t*)(buf + 0xe) = getNullTerminatedU
        __os_log_impl(&__macho_header, log, debug_typ
        _swift_arrayDestroy(x0_4, 1, (type metadata f
        _swift_slowDealloc(x0_4, -1, -1);
        _swift_slowDealloc(buf, -1, -1);
    }

    _objc_release(log);
    return 0;
}

int64_t* ___swift_allocate_value_buffer(void* arg1,
```



```
int64_t _main()

{
    void* x0 = type metadata accessor for Logger(0);
    ___swift_allocate_value_buffer(x0, &log);
    ___swift_project_value_buffer(x0, &log);
    Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat',
    char debug_type_1 = static os_log_type_t.debug.getter();
    os_log_t log = Logger.logObject.getter();
    uint32_t debug_type = ((uint32_t)debug_type_1);

    if (_os_log_type_enabled(log, debug_type) != 0) {
        uint8_t* buf = _swift_slowAlloc(0x16, -1);
        int64_t x0_4 = _swift_slowAlloc(0x20, -1);
        int64_t var_38 = x0_4;
        *(uint32_t*)buf = 0x8000202;
        *(uint64_t*)(buf + 4) = 42;
        *(uint16_t*)(buf + 0xc) = 0x820;
        *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn
        __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…
        _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
        _swift_slowDealloc(x0_4, -1, -1);
        _swift_slowDealloc(buf, -1, -1);
    }

    _objc_release(log);
    return 0;
}

int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)
```

```
int64_t _main()

{
    void* x0 = type metadata accessor for Logger(0);
    ___swift_allocate_value_buffer(x0, &log);
    ___swift_project_value_buffer(x0, &log);
    Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat',
    char debug_type_1 = static os_log_type_t.debug.getter();
    os_log_t log = Logger.logObject.getter();
    uint32_t debug_type = ((uint32_t)debug_type_1);

    if (_os_log_type_enabled(log, debug_type) != 0) {
        uint8_t* buf = _swift_slowAlloc(0x16, -1);
        int64_t x0_4 = _swift_slowAlloc(0x20, -1);
        int64_t var_38 = x0_4;
        *(uint32_t*)buf = 0x8000202;
        *(uint64_t*)(buf + 4) = 42;
        *(uint16_t*)(buf + 0xc) = 0x820;
        *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn
        __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…
        _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
        _swift_slowDealloc(x0_4, -1, -1);
        _swift_slowDealloc(buf, -1, -1);
    }

    _objc_release(log);
    return 0;
}
```

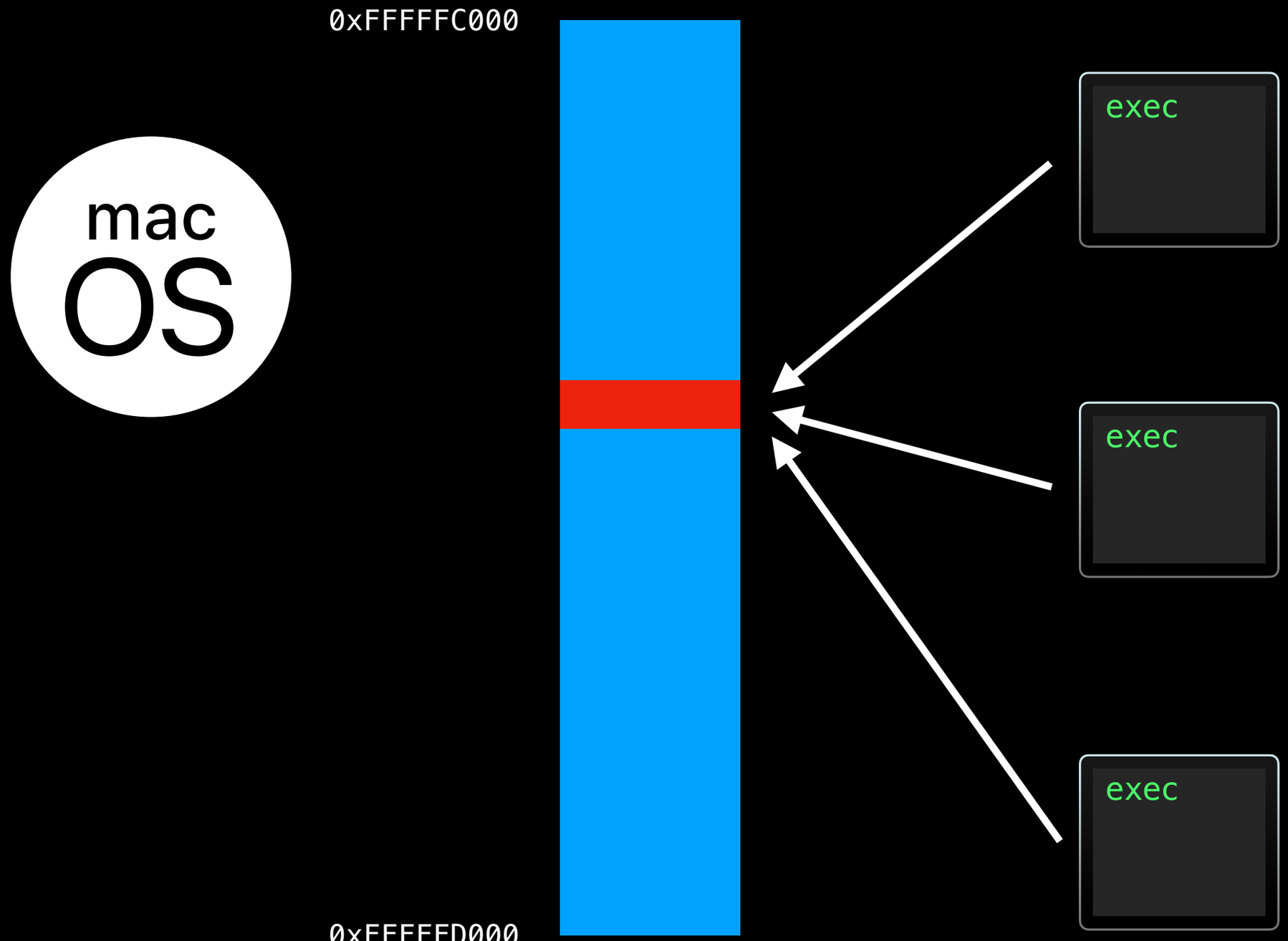
```
int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)
```

```
int64_t _main()

{
    void* x0 = type metadata accessor for Logger(0);
    ___swift_allocate_value_buffer(x0, &log);
    ___swift_project_value_buffer(x0, &log);
    Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat',
    char debug_type_1 = static os_log_type_t.debug.getter();
    os_log_t log = Logger.logObject.getter();
    uint32_t debug_type = ((uint32_t)debug_type_1);

    if (_os_log_type_enabled(log, debug_type) != 0) {
        uint8_t* buf = _swift_slowAlloc(0x16, -1);
        int64_t x0_4 = _swift_slowAlloc(0x20, -1);
        int64_t var_38 = x0_4;
        *(uint32_t*)buf = 0x8000202;
        *(uint64_t*)(buf + 4) = 42;
        *(uint16_t*)(buf + 0xc) = 0x820;
        *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn
        __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…
        _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
        _swift_slowDealloc(x0_4, -1, -1);
        _swift_slowDealloc(buf, -1, -1);
    }

    _objc_release(log);
    return 0;
}
```

```
int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)
```

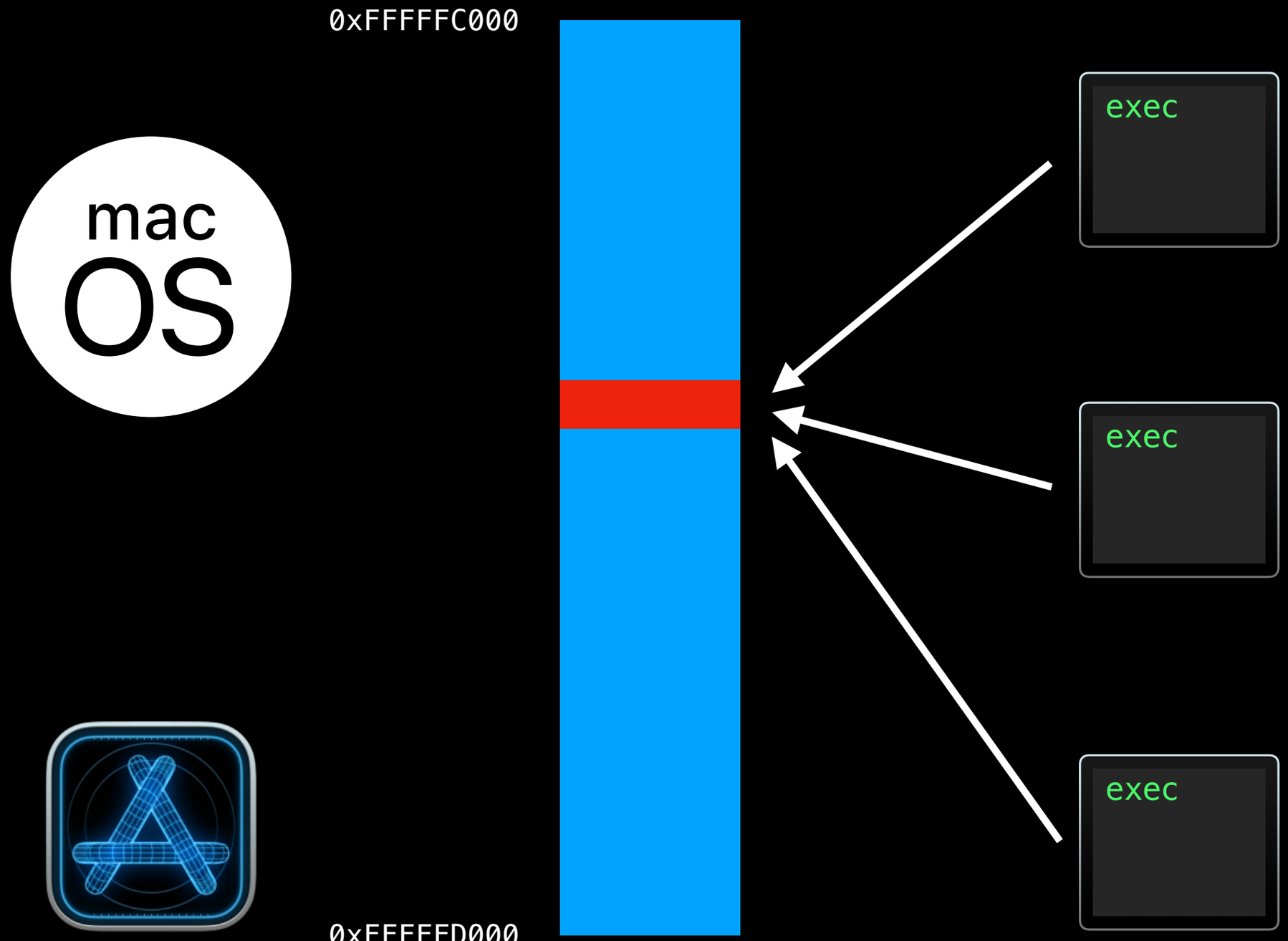
_os_log_type_enabled

commpage crash course



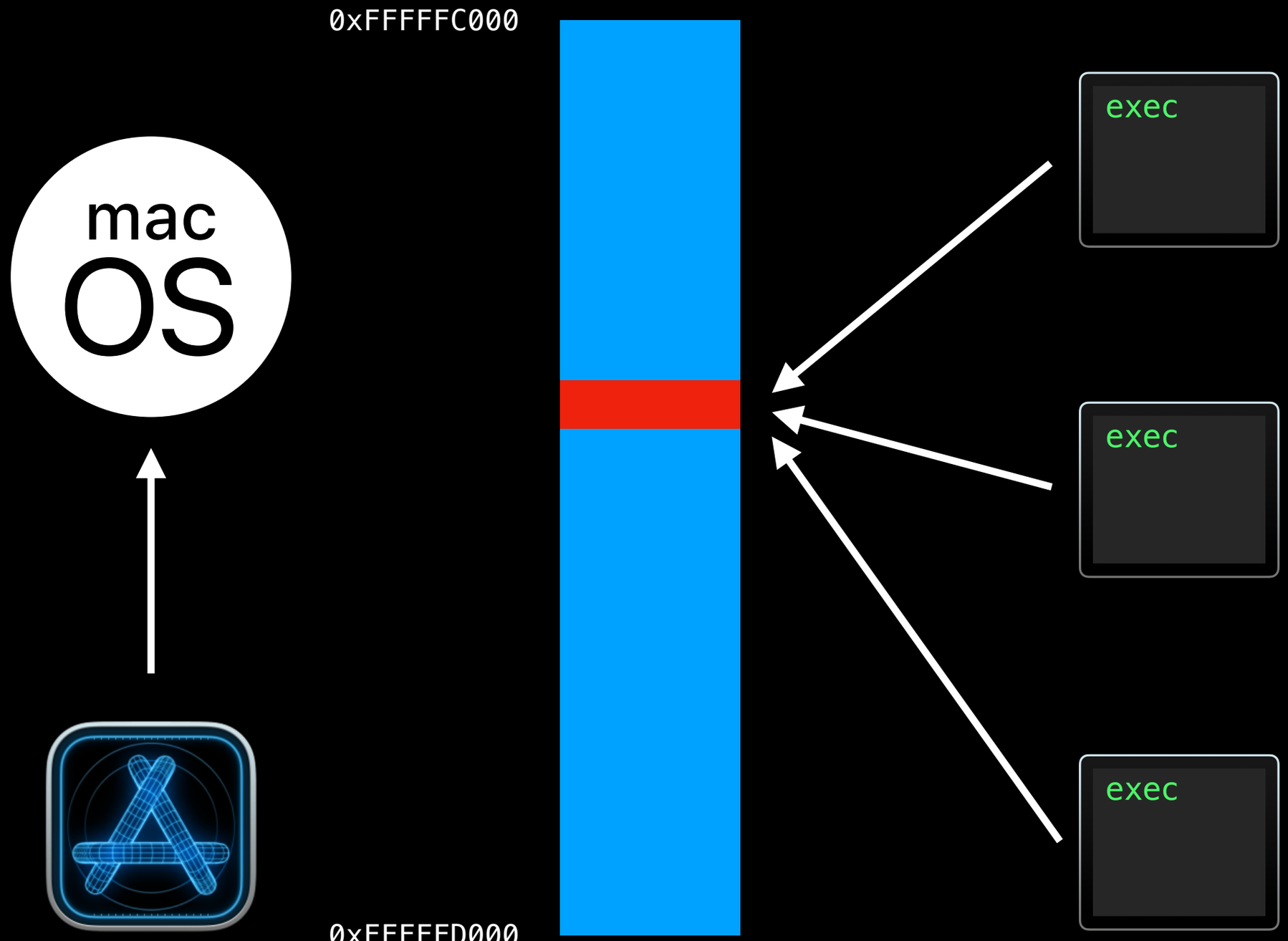
_os_log_type_enabled

commpage crash course



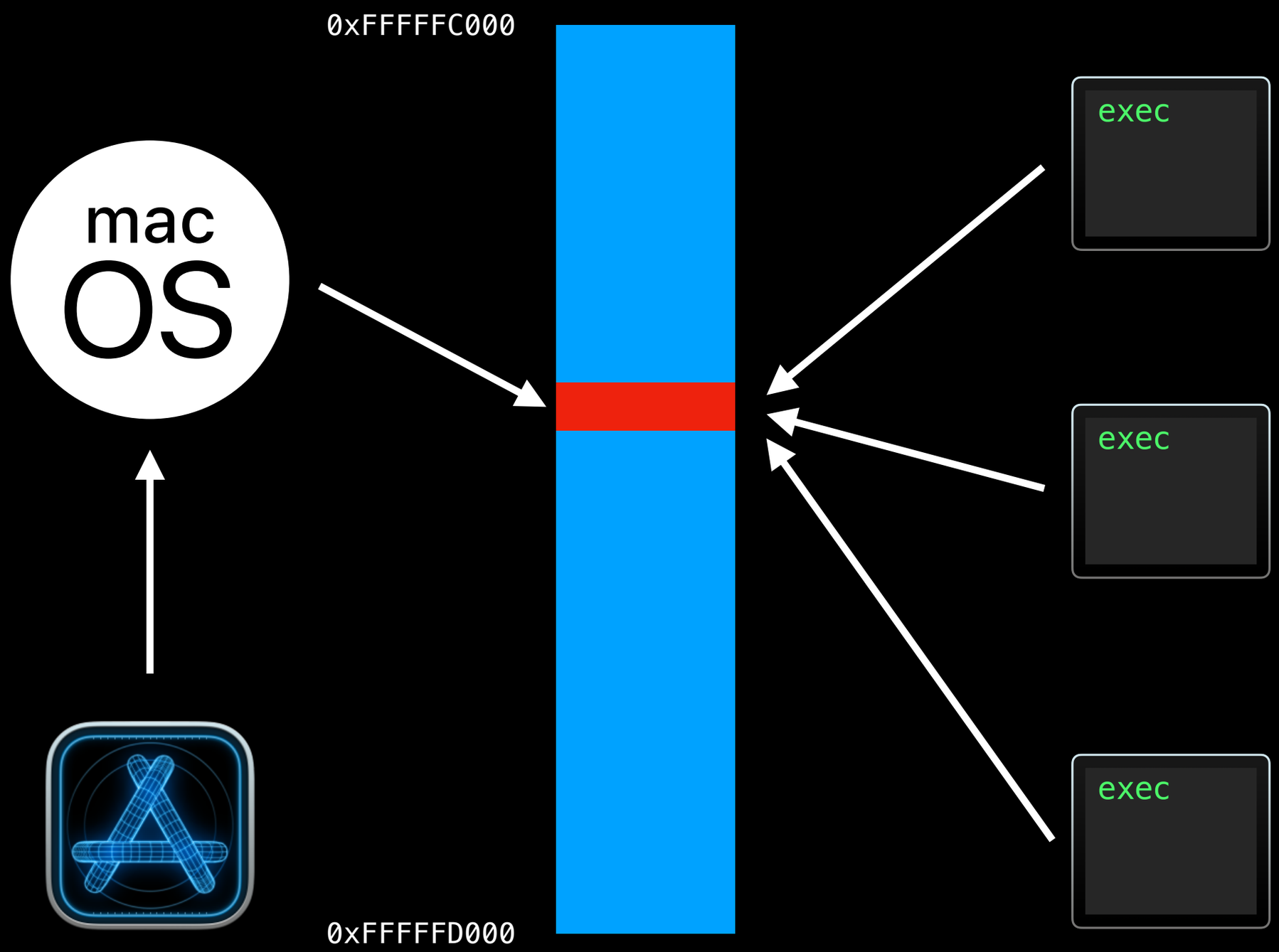
_os_log_type_enabled

commpage crash course



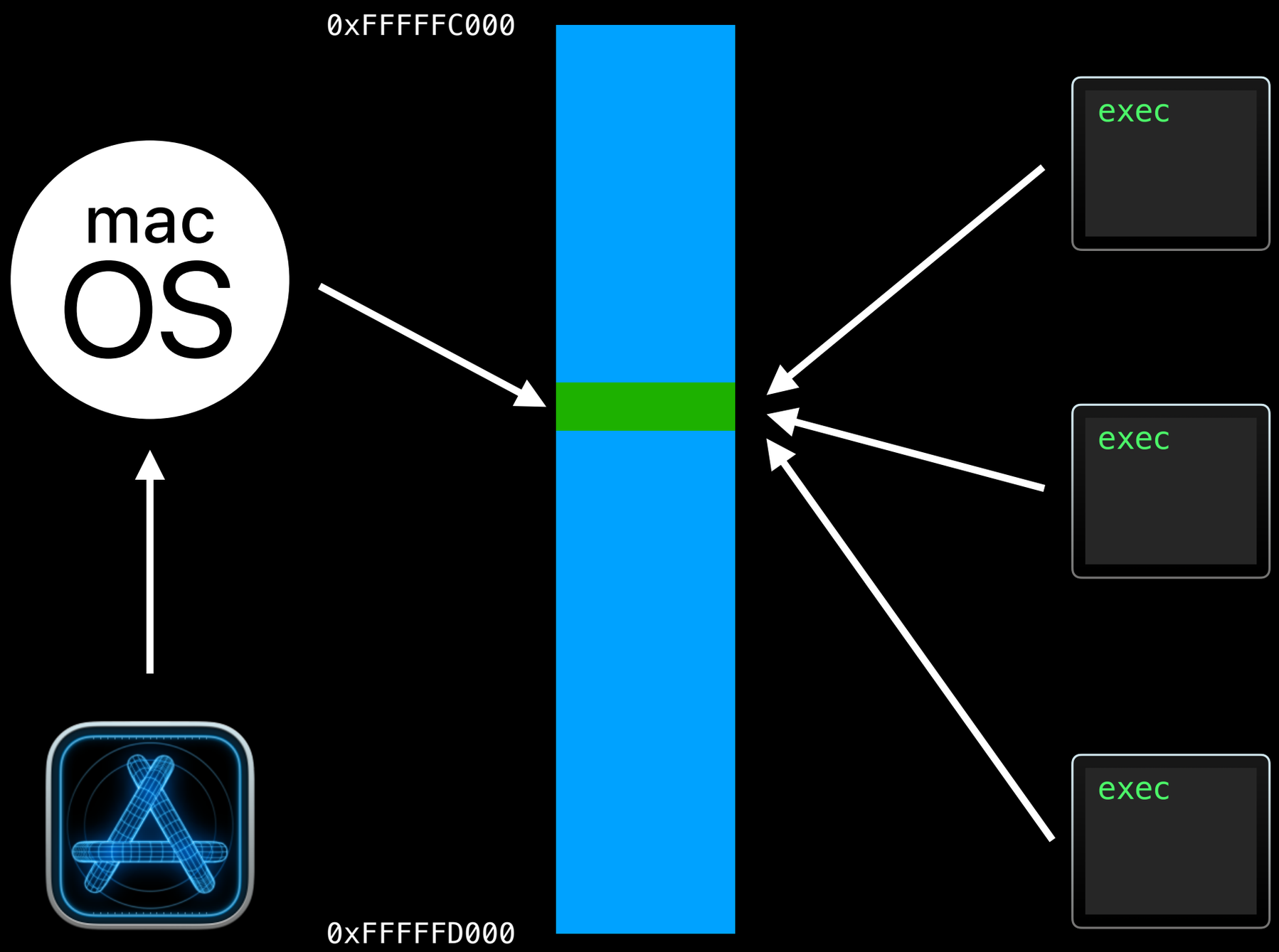
_os_log_type_enabled

commpage crash course



_os_log_type_enabled

commpage crash course



os_log limitations

os_log limitations

The downsides of unified logging

- App developers care about *their* logs
- System services are spammy
  - Hiding chatty logs isn’t enough–new ones keep getting added
- Logs go to a system store
  - Requires cross-process synchronization
  - No control over persistence

os_log limitations

Collecting logs is difficult

- “Please attach a sysdiagnose”
  - System-level action not appropriate for apps
  - Privacy implications
- APIs to get at your own logs are very limited
  - Or broken!

iOS has very limited facilities for reading the system log. Currently, an iOS app can only read entries created by that specific process, using `.currentProcessIdentifier` scope. This is annoying if, say, the app crashed and you want to know what it was doing before the crash. What you need is a way to get all log entries written by your app (r. 57880434).

There are two known bugs with the `.currentProcessIdentifier` scope. The first is that the `.reverse` option doesn’t work (r. 87622922). You always get log entries in forward order. The second is that the `getEntries(with:at:matching:)` method doesn’t honour its position argument (r. 87416514). You always get all available log entries.

os_log limitations

You are Apple's guest

- Very clearly an internal API exposed to third party developers
- Serves Apple's internal needs
- Only gets improvements with OS updates
- No way to add new functionality
  - Wrapping logs
  - Collecting additional diagnostics

Logging for the rest of us

Logging for the rest of us

What do we actually want?

- Control
  - We should own the logs
- Fast and persistent, pick two
  - Scale to millions of writes per second
  - Saved even after crashes
- Structured
- Open source?
  - We can improve it!

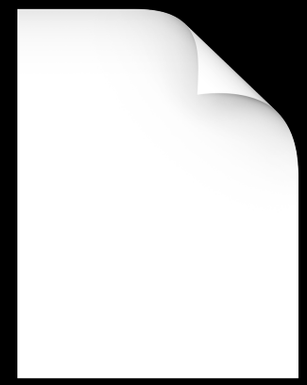
Chronicle

Chronicle

Design

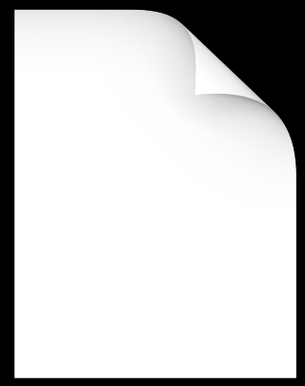
- Fast
  - Overhead measured in nanoseconds
- Structured
  - Typed log format
- Persistent
  - Memory mapped ring buffer

Memory mapping

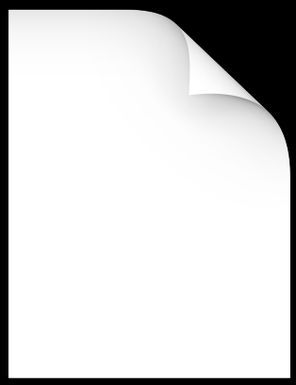


Memory mapping

Memory mapping



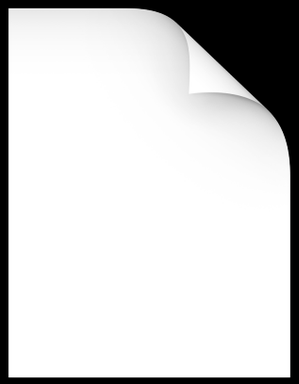
Memory mapping



Memory mapping

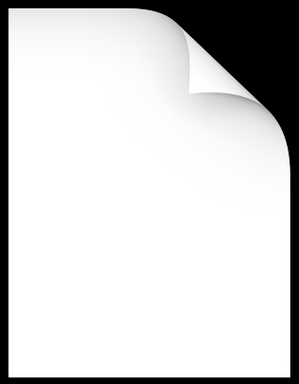
Memory mapping

Memory mapping



Memory mapping

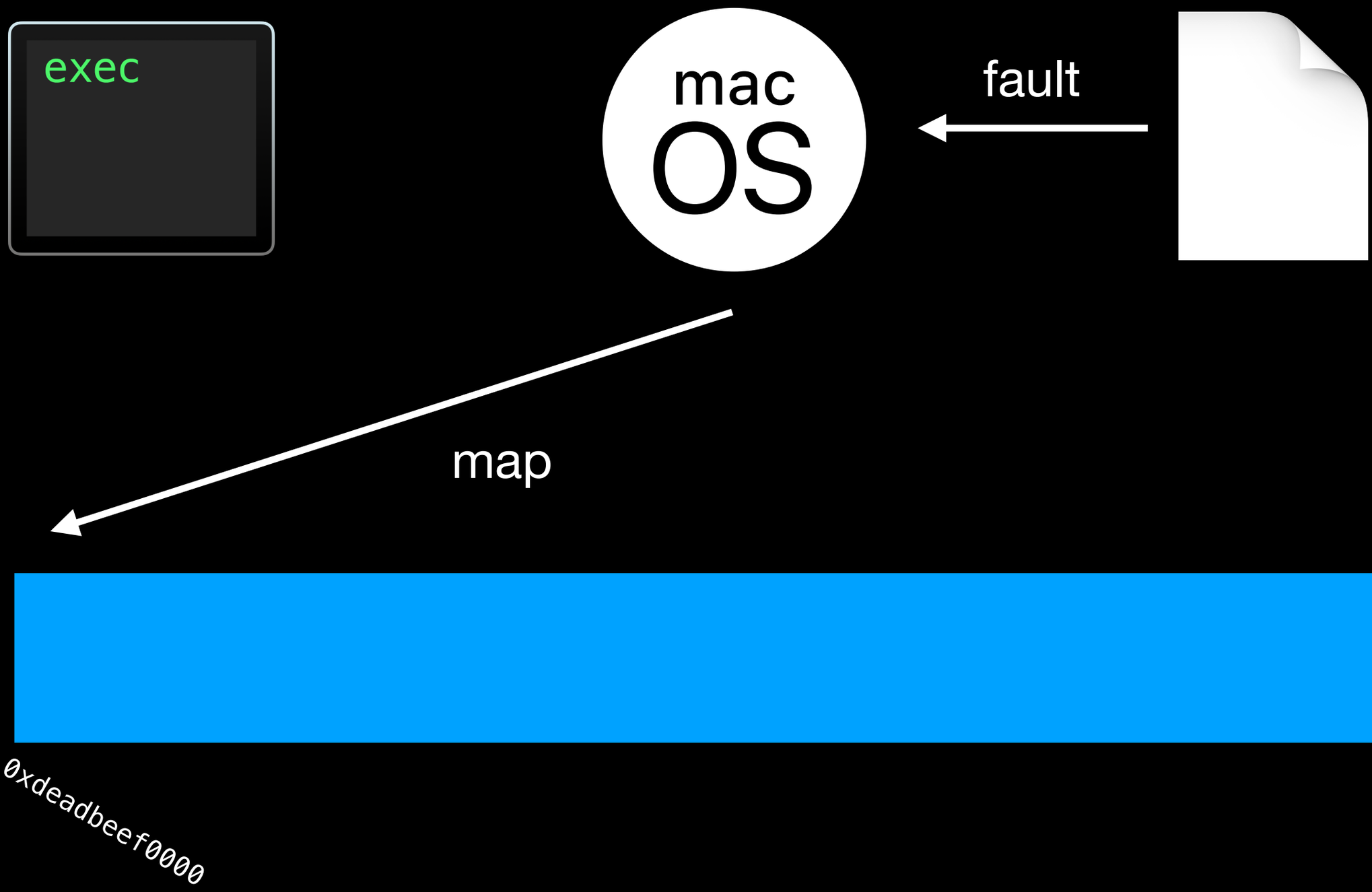
Memory mapping



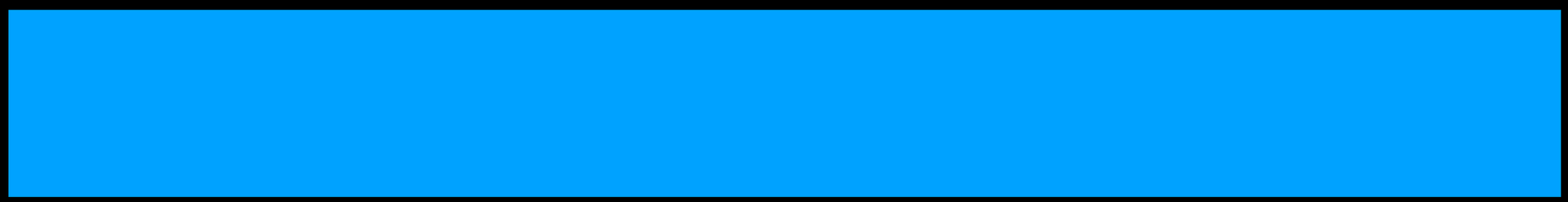
Memory mapping

Memory mapping

Memory mapping

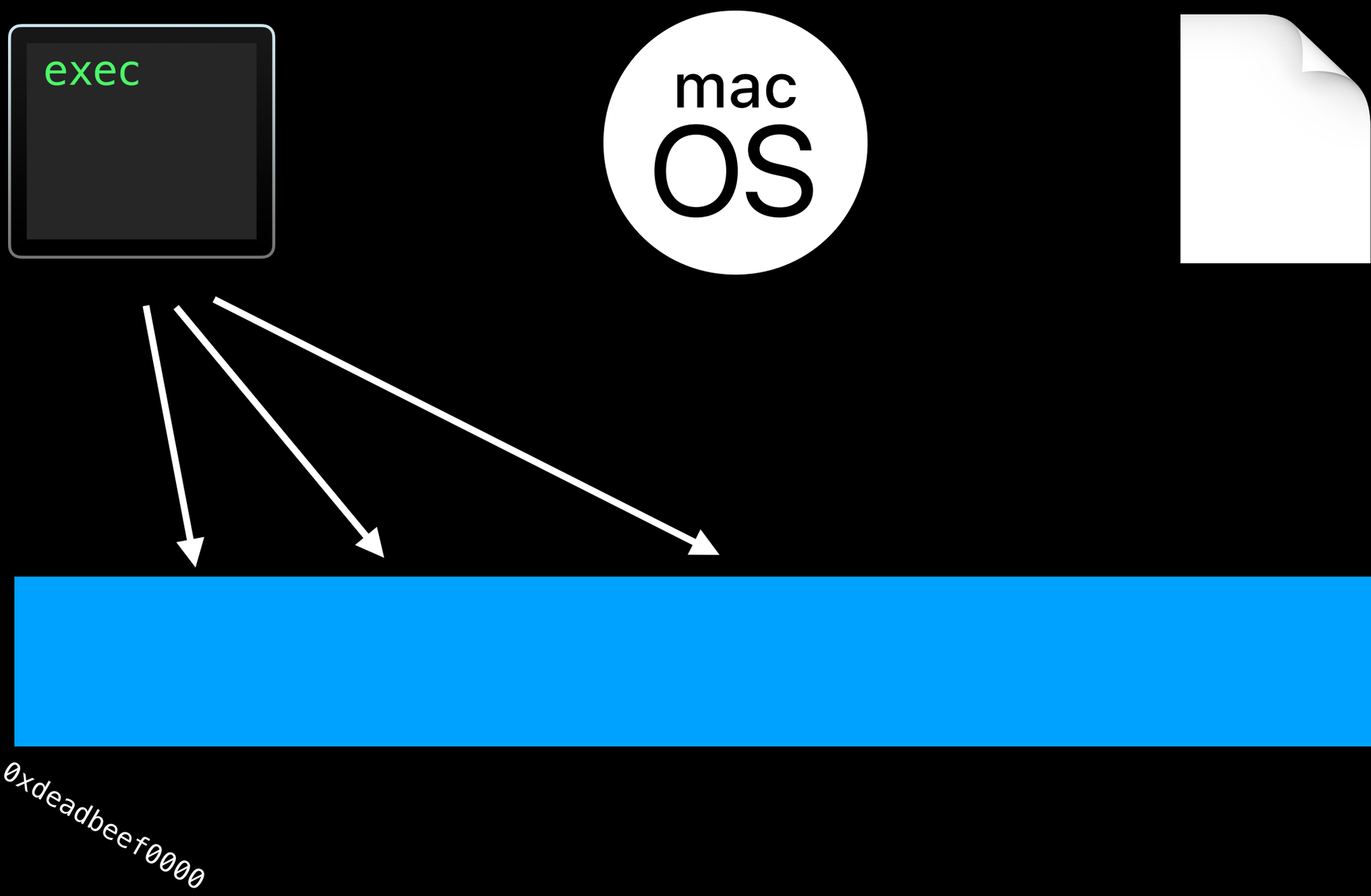


Memory mapping

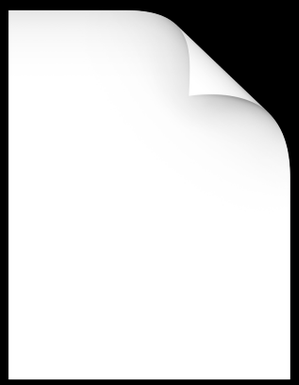


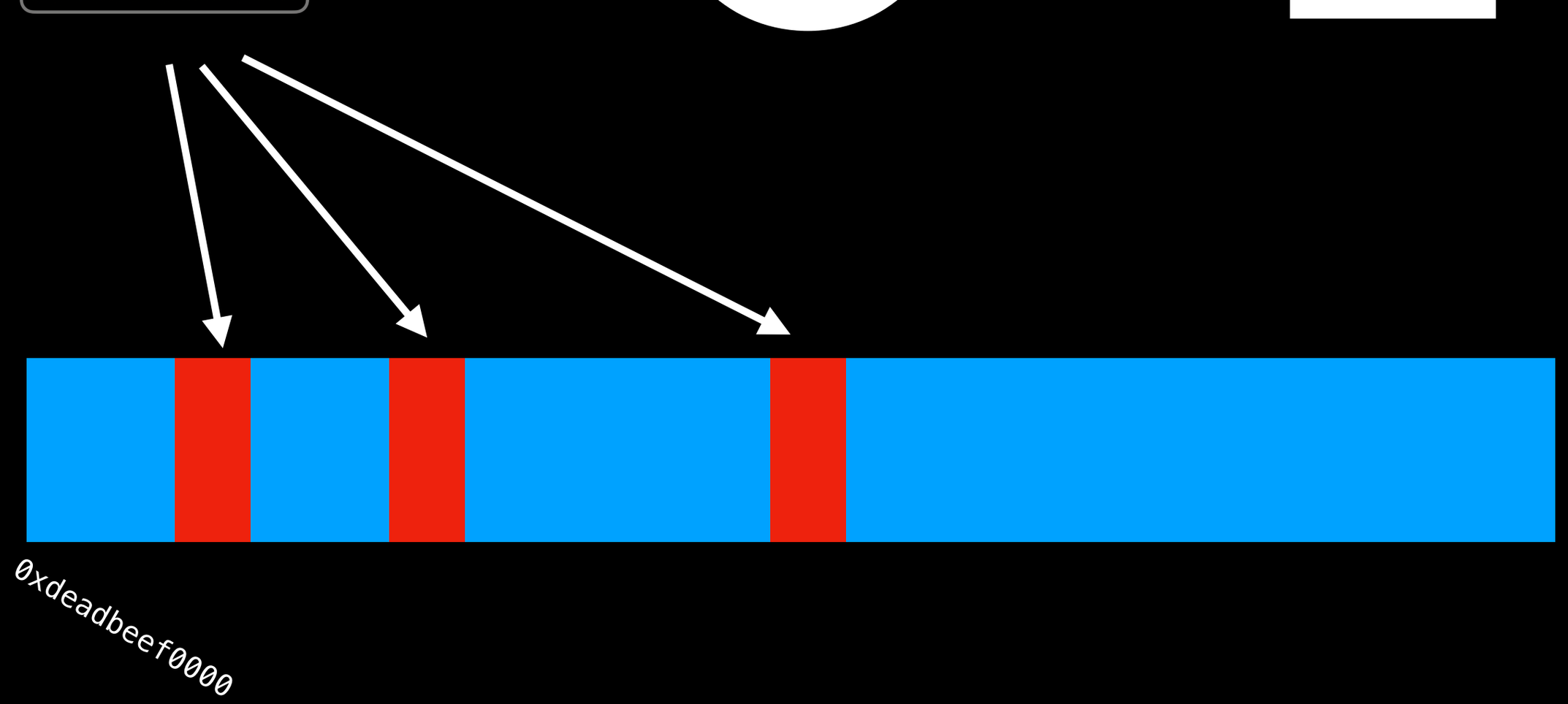
0xdeadbeef0000

Memory mapping

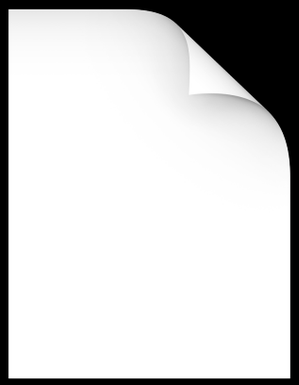


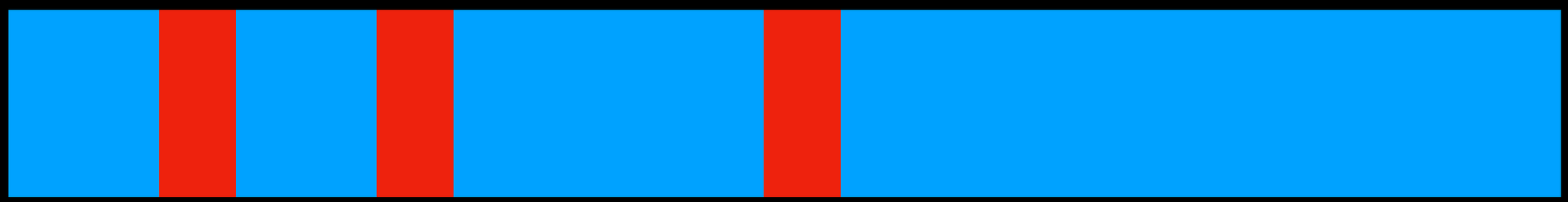
Memory mapping





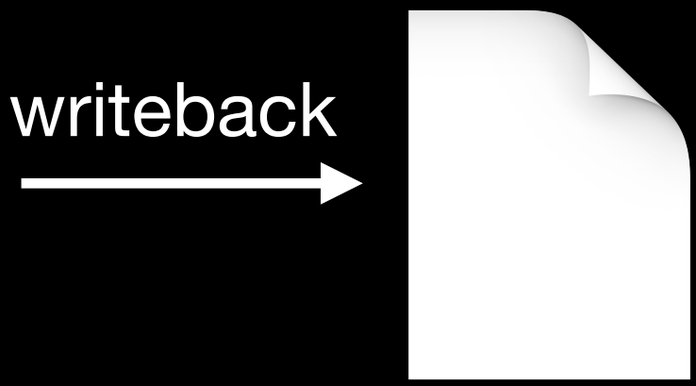
Memory mapping

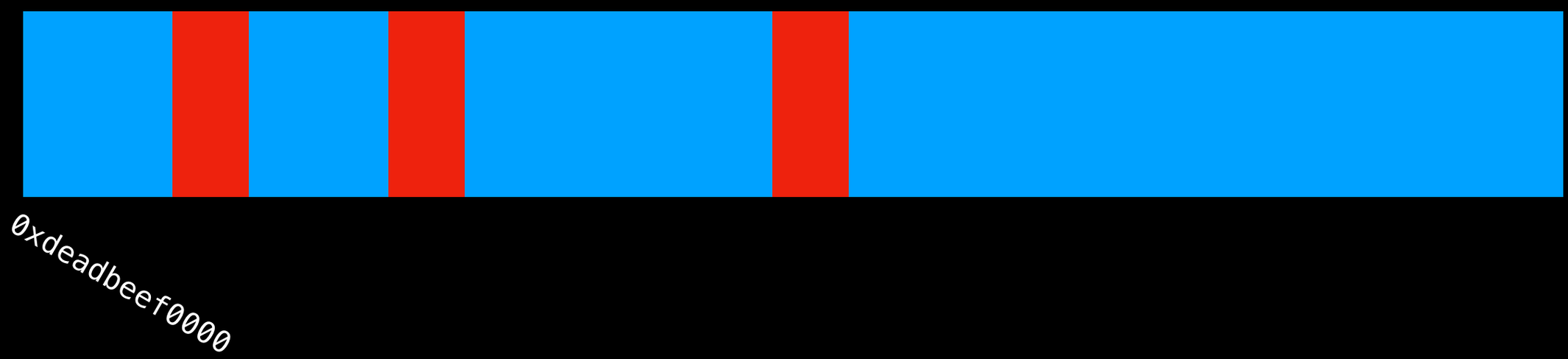




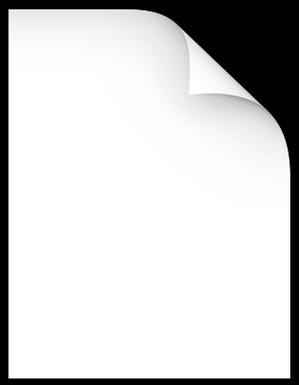
0xdeadbeef0000

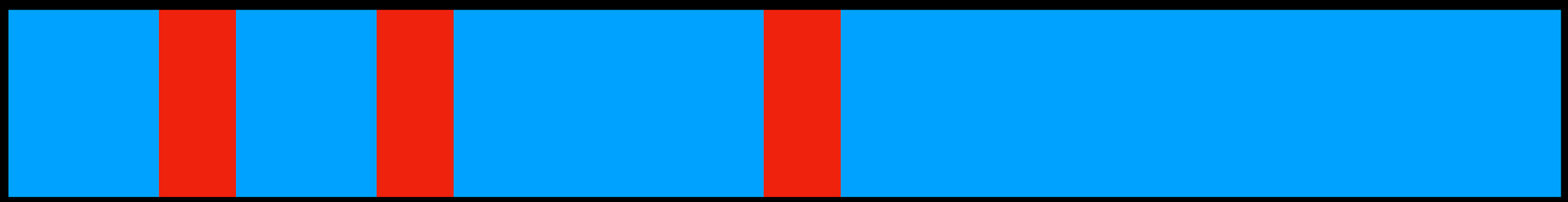
Memory mapping





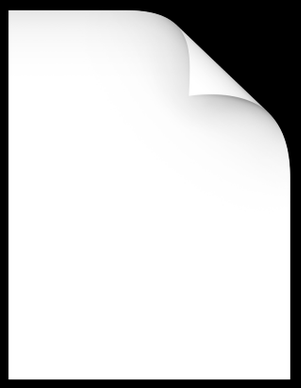
Memory mapping

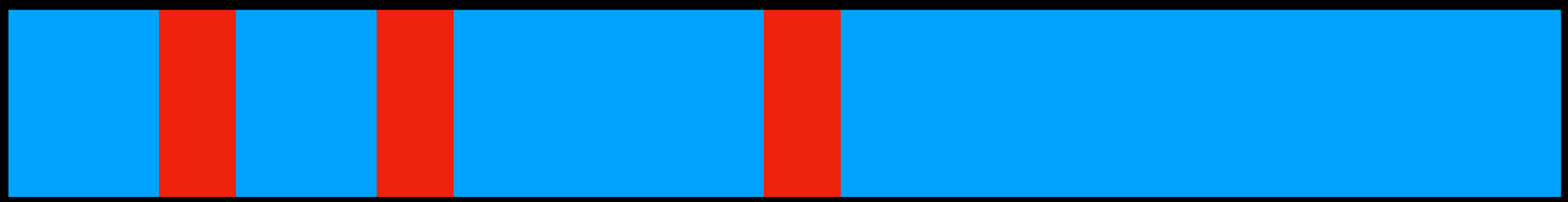




0xdeadbeef0000

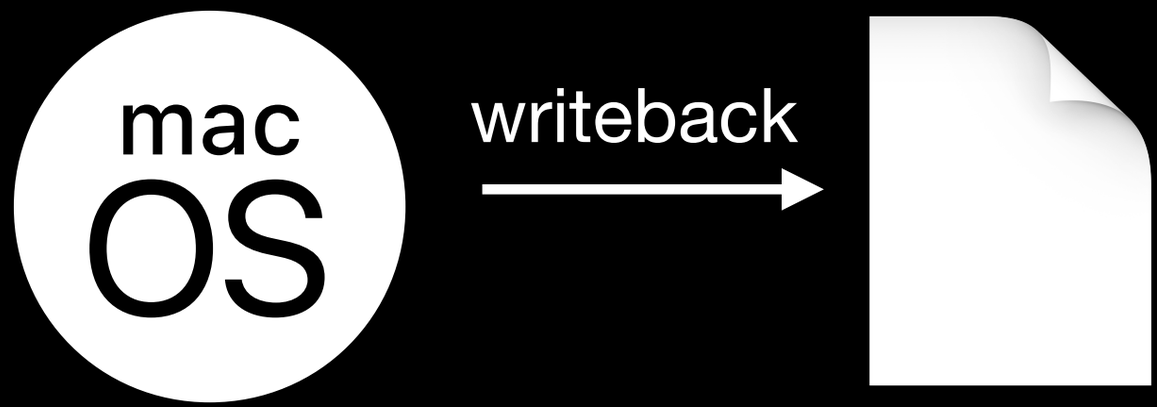
Memory mapping

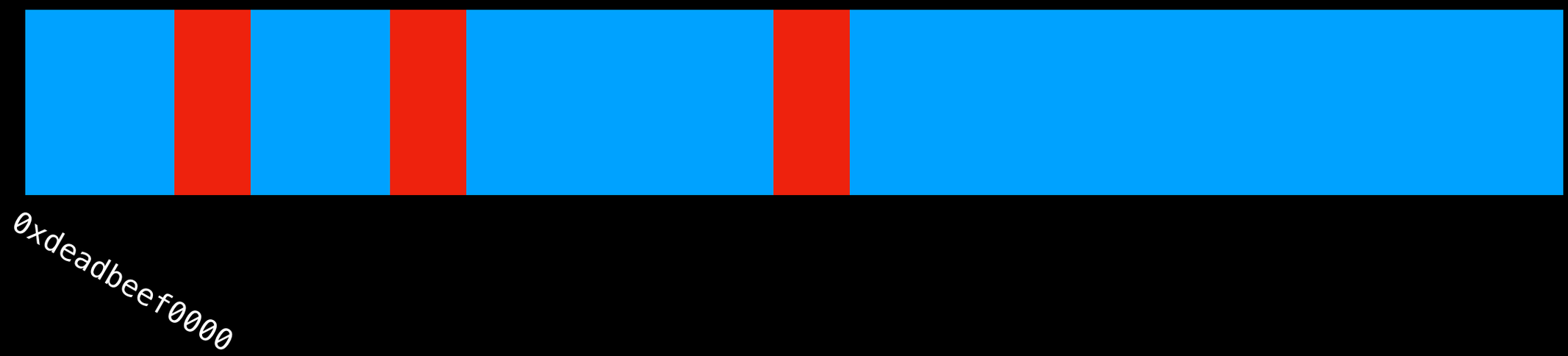




0xdeadbeef0000

Memory mapping





Layout
Real artists steal

- Similar layout to os_log
  - Metadata
  - Type information
  - Data

Type	Encoding
`Bool`	b
`Int8`	1
`Int16`	2
`Int32`	4
`Int64`	8
`Int`	i
`UInt8`	!
`UInt16`	@
`UInt32`	$
`UInt64`	*
`UInt`	I
`Float`	f
`Double`	F
`String`	s
`StaticString`	S

`StaticString` optimizations

What is that `__dso_handle`, anyway?

I have `/private/var/db/uuidtext/dsc` directory full of files. Directory size is >100GB and constantly growing. I've deleted all the files and also tried to reboot my mac, but it doesn't help. I've found little info about this directory, but it looks like that these files are log files, but I'm not able to figure out which process causes such a logging.

Files in /private/var/db/uuidtext/dsc reported infected

Hello,

After an Avast analyze, some files in /private/var/db/uuidtext/dsc are reported infected with this message "ELF:MiraiDowloader-OG(Drp)". Is this a true threat or a false positive ?

How to stop /private/var/db/uuidtext/dsc keep growing

From this thread: logd causes /private/var/db/uuidtext/dsc to grow until disk is full
Solution You need to reinstall your OS X!
1. Reboot your mac
2. Hold (Command + R) keys to Disk Utility.
3. Select Reinstall OS X.
This will be install OS X "in place" of your old one without losing any data.
Problem solved. 😉

HD being filled with garbage log (?) files (self.mac)
submitted 6 years ago * by Mmngmf_almost_therrr

See update at bottom.

Sometime during the night, the directory /private/var/db/uuidtext/dsc started filling with 40.9MB files until the disk was
The service launchd appears to be overactive, but I'm not sure if that's a symptom or a cause. I'm running EtreCheck b

`StaticString` optimizations
Our own janky `__TEXT,__oslogstring`

- os_log doesn't bother copying constant strings
  - Stores base address and offset
  - `__TEXT,__oslogstring` stores all the strings in one place
  - This gets copied to /var/db/uuidtext
- Storing things in this section requires compiler support
- Just copy all the strings (in `__TEXT,__cstring`)
  - We can optimize this by only copying strings from our binary

test — Binary Ninja 4.1.5619-test

test

Mach-O ▾ Linear ▾ Pseudo C ▾

```
0x1000030e0

1000030e0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
100003100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
100003120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
100003140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
100003160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................

__text (PURE_CODE) section started  {0x10000317c-0x100003bb8}

10000317c  int64_t _main()

10000317c  {
10000319c      void* x0 = type metadata accessor for Logger(0);
1000031b0      ___swift_allocate_value_buffer(x0, &log);
1000031bc      ___swift_project_value_buffer(x0, &log);
1000031fc      Logger.init(subsystem:category:)(-0x2fffffffffffffe4, -0x7fffffffffc2e0, 'Test Cat', 'egory\x00\x00\xed');
100003200      char debug_type_1 = static os_log_type_t.debug.getter();
100003208      os_log_t log = Logger.logObject.getter();
100003210      uint32_t debug_type = ((uint32_t)debug_type_1);
100003210
10000321c      if (_os_log_type_enabled(log, debug_type) != 0) {
100003228          uint8_t* buf = _swift_slowAlloc(0x16, -1);
100003238          int64_t x0_4 = _swift_slowAlloc(0x20, -1);
100003240          int64_t var_38 = x0_4;
10000324c          *(uint32_t*)buf = 0x8000202;
100003254          *(uint64_t*)(buf + 4) = 42;
10000325c          *(uint16_t*)(buf + 0xc) = 0x820;
100003280          *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn:)(-0xd000000000000017, -0x7fffffffffc2c0, &var_38)
1000032a4          __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…", buf, 0x16);
1000032bc          _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
1000032cc          _swift_slowDealloc(x0_4, -1, -1);
1000032dc          _swift_slowDealloc(buf, -1, -1);
10000321c      }
10000321c
1000032e4      _objc_release(log);
1000032fc      return 0;
10000317c  }

100003300  int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)

100003300  {
100003310      void* x9 = *(uint64_t*)((char*)arg1 - 8);
100003314      int32_t x8 = *(uint32_t*)((char*)x9 + 0x50);
100003314
100003318      if ((x8 & 0x20000) == 0)
```

14 mac-aarch64 0x100002dea

```
int64_t _main()

{
    void* x0 = type metadata accessor for Logger(0);
    ___swift_allocate_value_buffer(x0, &log);
    ___swift_project_value_buffer(x0, &log);
    Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat',
    char debug_type_1 = static os_log_type_t.debug.getter();
    os_log_t log = Logger.logObject.getter();
    uint32_t debug_type = ((uint32_t)debug_type_1);

    if (_os_log_type_enabled(log, debug_type) != 0) {
        uint8_t* buf = _swift_slowAlloc(0x16, -1);
        int64_t x0_4 = _swift_slowAlloc(0x20, -1);
        int64_t var_38 = x0_4;
        *(uint32_t*)buf = 0x8000202;
        *(uint64_t*)(buf + 4) = 42;
        *(uint16_t*)(buf + 0xc) = 0x820;
        *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn
        __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…
        _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
        _swift_slowDealloc(x0_4, -1, -1);
        _swift_slowDealloc(buf, -1, -1);
    }

    _objc_release(log);
    return 0;
}
```

```
int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)
```

```
int64_t _main()

{
    void* x0 = type metadata accessor for Logger(0);
    ___swift_allocate_value_buffer(x0, &log);
    ___swift_project_value_buffer(x0, &log);
    Logger.init(subsystem:category:)(-0x2ffffffffffffe4, -0x7ffffffefffc2e0, 'Test Cat',
    char debug_type_1 = static os_log_type_t.debug.getter();
    os_log_t log = Logger.logObject.getter();
    uint32_t debug_type = ((uint32_t)debug_type_1);

    if (_os_log_type_enabled(log, debug_type) != 0) {
        uint8_t* buf = _swift_slowAlloc(0x16, -1);
        int64_t x0_4 = _swift_slowAlloc(0x20, -1);
        int64_t var_38 = x0_4;
        *(uint32_t*)buf = 0x8000202;
        *(uint64_t*)(buf + 4) = 42;
        *(uint16_t*)(buf + 0xc) = 0x820;
        *(uint64_t*)(buf + 0xe) = getNullTerminatedUTF8PointerImpl(_:storingStringOwnersIn
        __os_log_impl(&__macho_header, log, debug_type, "This is a number: %ld. This is a…
        _swift_arrayDestroy(x0_4, 1, (type metadata for Any + 8));
        _swift_slowDealloc(x0_4, -1, -1);
        _swift_slowDealloc(buf, -1, -1);
    }

    _objc_release(log);
    return 0;
}
```

```
int64_t* ___swift_allocate_value_buffer(void* arg1, int64_t* arg2)
```

Minimizing work

…without custom compiler passes

- Reduce the amount of work done as much as possible
- Source transformations are more powerful than compiler passes
  - Macros!
- Minimize copies

main.swift

test 〉 Sources 〉 test 〉 main.swift 〉 No Selection

```swift
import Chronicle
import Foundation

let chronicle = try Chronicle(url: URL(fileURLWithPath: "log.chronicle"), bufferSize: 1 << 20)
let logger = try chronicle.logger(name: "test")

#log(logger, "Invocation: \(String(cString: getprogname())) [\(CommandLine.argc) arguments]")
#log(logger, "It's been \(Date().timeIntervalSince1970) seconds since the epoch")

```

Minimizing work

Avoiding copies and allocations

- For each message:
  - Calculate total size of all arguments
  - Reserve section of ring buffer to write to
  - Write data directly into reservation
  - Release reservation
- Strings are complicated
  - UTF-8 strings can be copied directly
  - Bridged strings need to be converted

Crash tolerance

Things don’t only go wrong when you want them to

- Writes may be interrupted if the process crashes early
  - We need to do a two-step process for any write
    - Mark in header that we will begin a write of n bytes
      - When parsing, the following data cannot be trusted…
      - …until we finish the write and update the header
- Ring buffers are hard
  - Messages actually need to be in a doubly-linked list

Questions?