Windows Reversing Basic

aaaddress1@chroot.org

台灣科技大學 資安實務

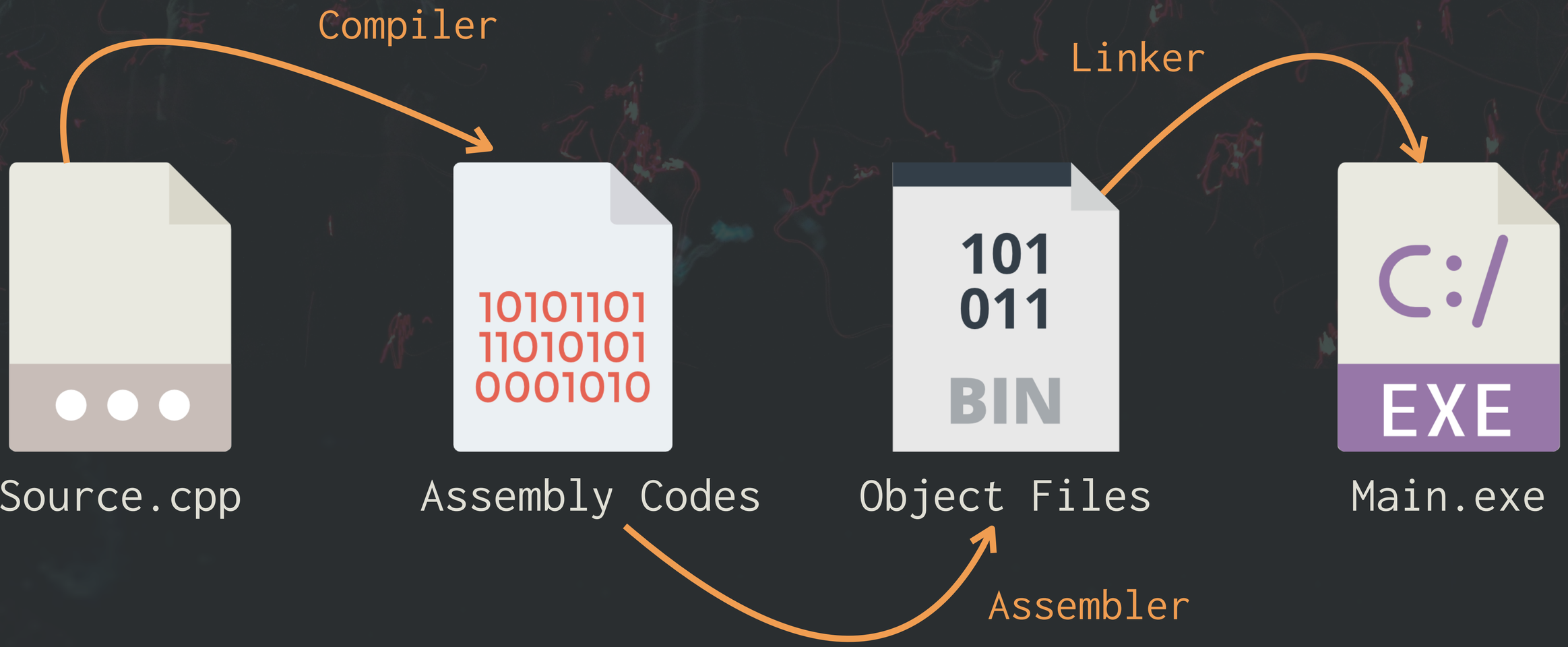
$_ whoami

- blog.30cm.tw
- aaaddress1@chroot.org
- Master degree at CSIE, NTUST
- **Security Researcher** - chr0.ot
- **Speaker** - BlackHat, DEFCON, VXCON, HITCON
- aaaddress1@chroot.org

#Windows #Reversing #Pwn #Exploit #EoP

cd Compiler

compiler

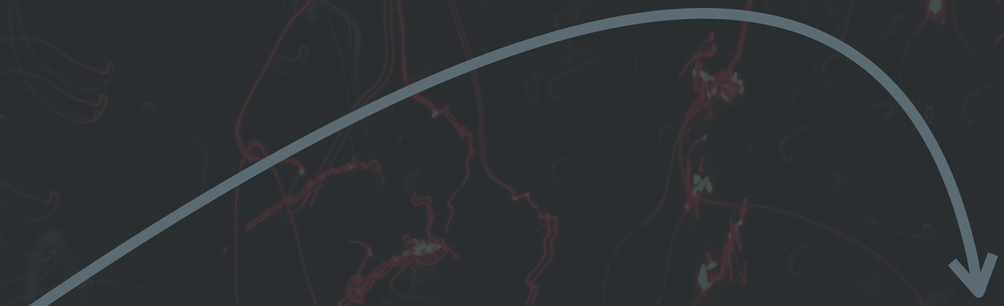

Compiler
Linker
10101101
11010101
0001010
101
011
BIN
C:/
EXE
Source.cpp
Assembly Codes
Object Files
Main.exe
Assembler

compiler

```
#include <Windows.h>
int main()
{
    MessageBoxA(
        0, "hi there.", "info", 0
    );
    return 0;
}
```

#compiler

```c
#include <Windows.h>
int main() {
  MessageBoxA(
    0,
    "hi there.",
    "info", 0
  );
  return 0;
}
```

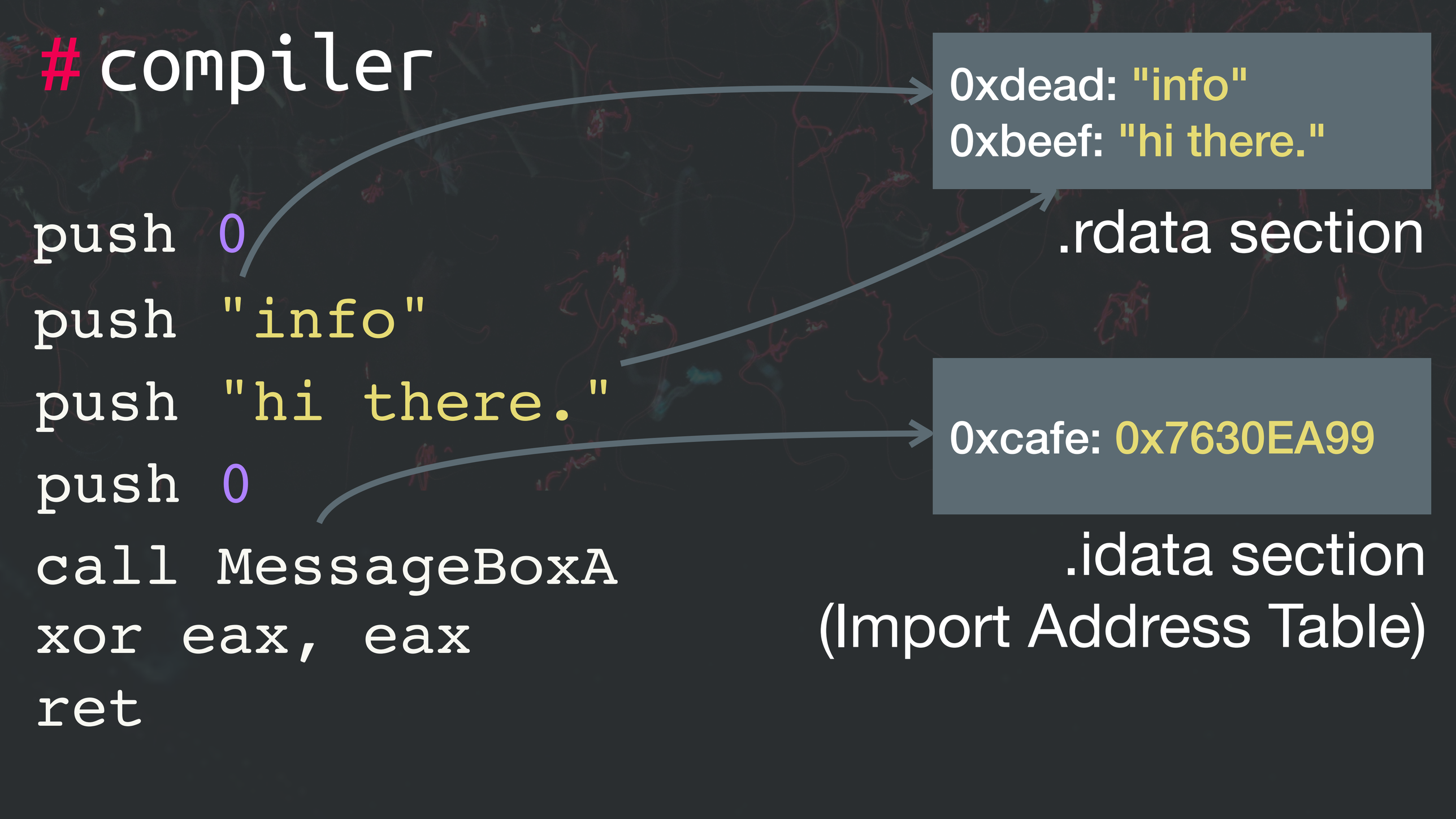


```
push 0
push "info"
push "hi there."
push 0
call MessageBoxA
xor eax, eax
ret
```

en.wikipedia.org/wiki/X86_calling_conventions

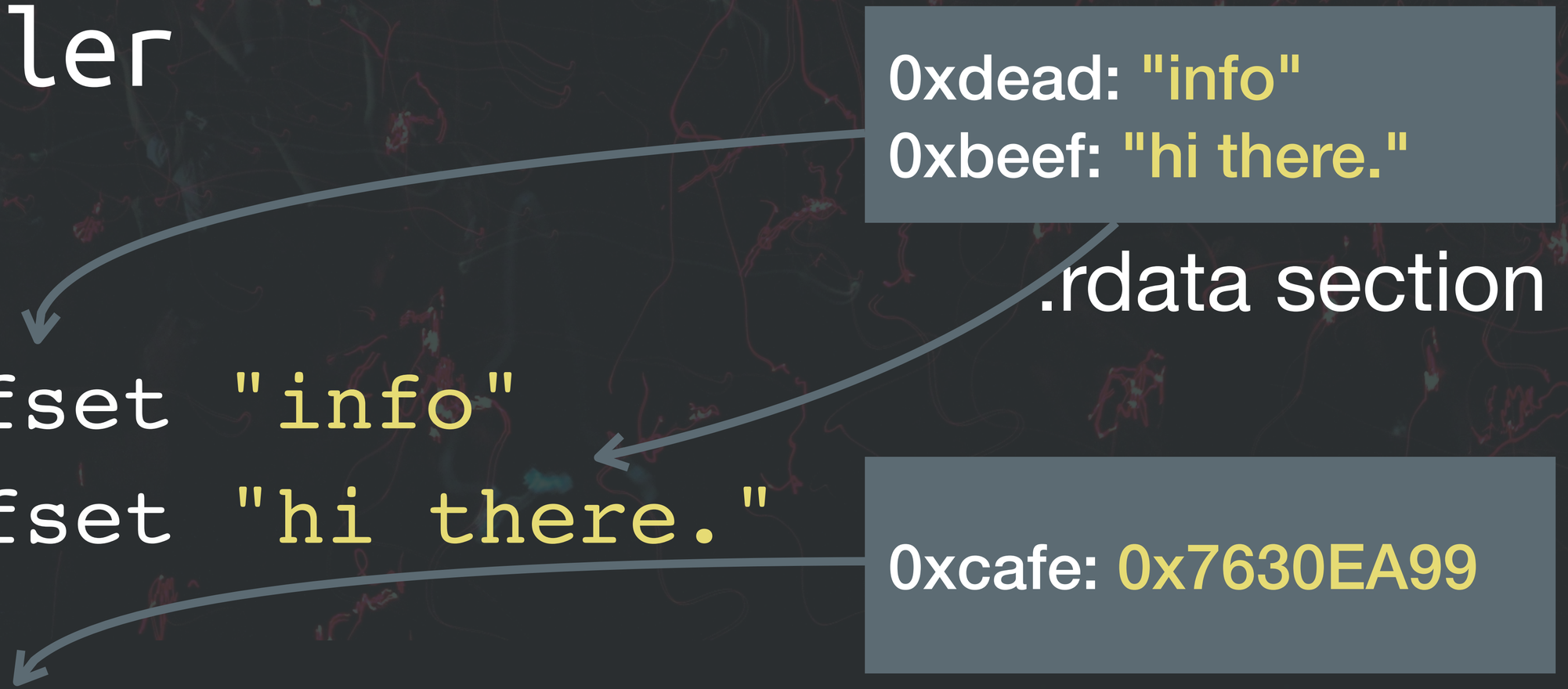
compiler

```
push 0
push "info"
push "hi there."
push 0
call MessageBoxA
xor eax, eax
ret
```



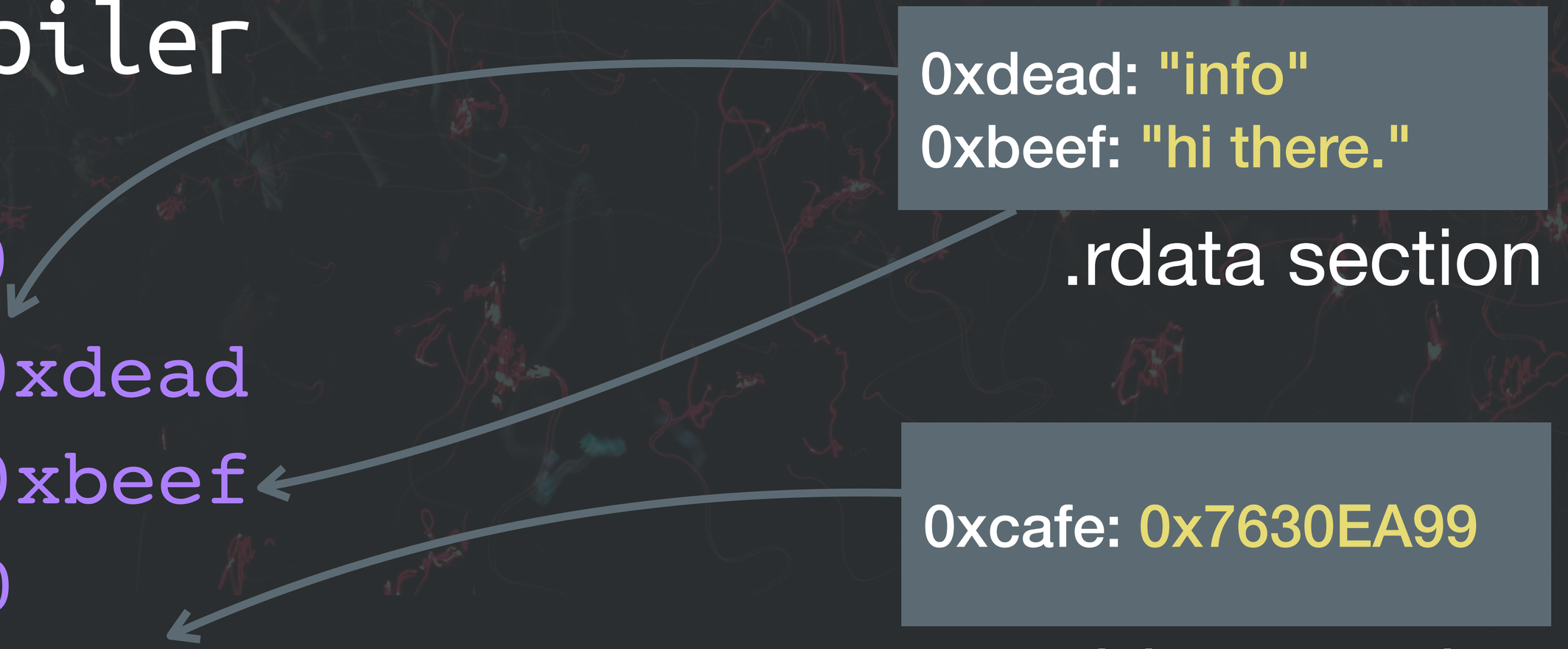
compiler

```
push 0
push offset "info"
push offset "hi there."
push 0
call MessageBoxA
xor eax, eax
ret
```



compiler

```
push 0
push 0xdead
push 0xbeef
push 0
call ds:0xcafe
xor eax, eax
ret
```



compiler

```
push    0               ; 6A 00
push    0xdead          ; 68 AD DE 00 00
push    0xbeef          ; 68 EF BE 00 00
push    0               ; 6A 00
call    ds:0xcafe       ; FF 15 FE CA 00 00
xor     eax, eax        ; 33 C0
ret                     ; C3
```

compiler

.text Section

```
6A 00
68 AD DE 00 00
68 EF BE 00 00
6A 00
FF 15 FE CA 00 00
33 C0
C3
```

Main.exe

0xdead: "info"
0xbeef: "hi there."

.rdata Section

0xcafe: 0x7630EA99

.idata Section


cd Hell_World.c

Compiler

$ gcc -S hellWorld.cpp

```
C:\Users\exploit\Desktop\TwTech_Rev
λ gcc -S -masm=intel hellWorld.cpp

C:\Users\exploit\Desktop\TwTech_Rev
λ cat hellWorld.s
        .file   "hellWorld.cpp"
        .intel_syntax noprefix
        .section        .text$_Z6printfPKcz,"x"
        .linkonce discard
        .globl  __Z6printfPKcz
        .def    __Z6printfPKcz; .scl    2;      .type   32;     .endef
__Z6printfPKcz:
LFB25:
        .cfi_startproc
        push    ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        mov     ebp, esp
```

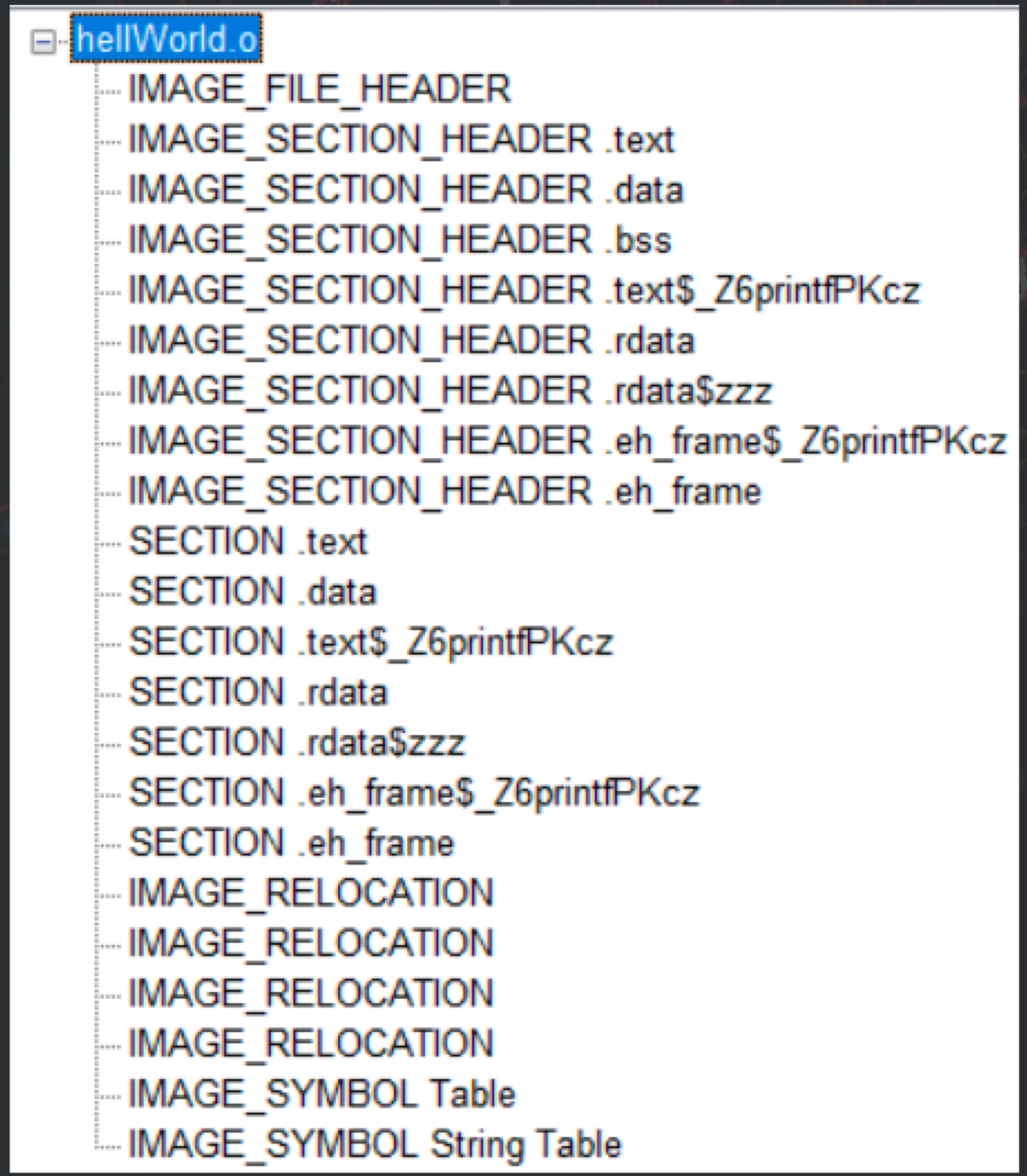
Assembler $ gcc -O0 -c hellWorld.s

```
C:\Users\exploit\Desktop\TwTech_Rev
λ gcc -c hellWorld.s

C:\Users\exploit\Desktop\TwTech_Rev
λ file hellWorld.o
hellWorld.o: Intel 80386 COFF object file,
bols

C:\Users\exploit\Desktop\TwTech_Rev
λ
```

COFF File?


helloWorld.o
IMAGE_FILE_HEADER
IMAGE_SECTION_HEADER .text
IMAGE_SECTION_HEADER .data
IMAGE_SECTION_HEADER .bss
IMAGE_SECTION_HEADER .text$_Z6printfPKcz
IMAGE_SECTION_HEADER .rdata
IMAGE_SECTION_HEADER .rdata$zzz
IMAGE_SECTION_HEADER .eh_frame$_Z6printfPKcz
IMAGE_SECTION_HEADER .eh_frame
SECTION .text
SECTION .data
SECTION .text$_Z6printfPKcz
SECTION .rdata
SECTION .rdata$zzz
SECTION .eh_frame$_Z6printfPKcz
SECTION .eh_frame
IMAGE_RELOCATION
IMAGE_RELOCATION
IMAGE_RELOCATION
IMAGE_RELOCATION
IMAGE_SYMBOL Table
IMAGE_SYMBOL String Table

COFF File

```
C:\Users\exploit\Desktop\TwTech_Rev
λ readCoff.exe hellWorld.o
.text: 00000154
.data: 000001ec
.bss: 00000000
/4: 000001f8
.rdata: 00000224
/24: 0000023c
/35: 0000027c
/59: 000002b8
```

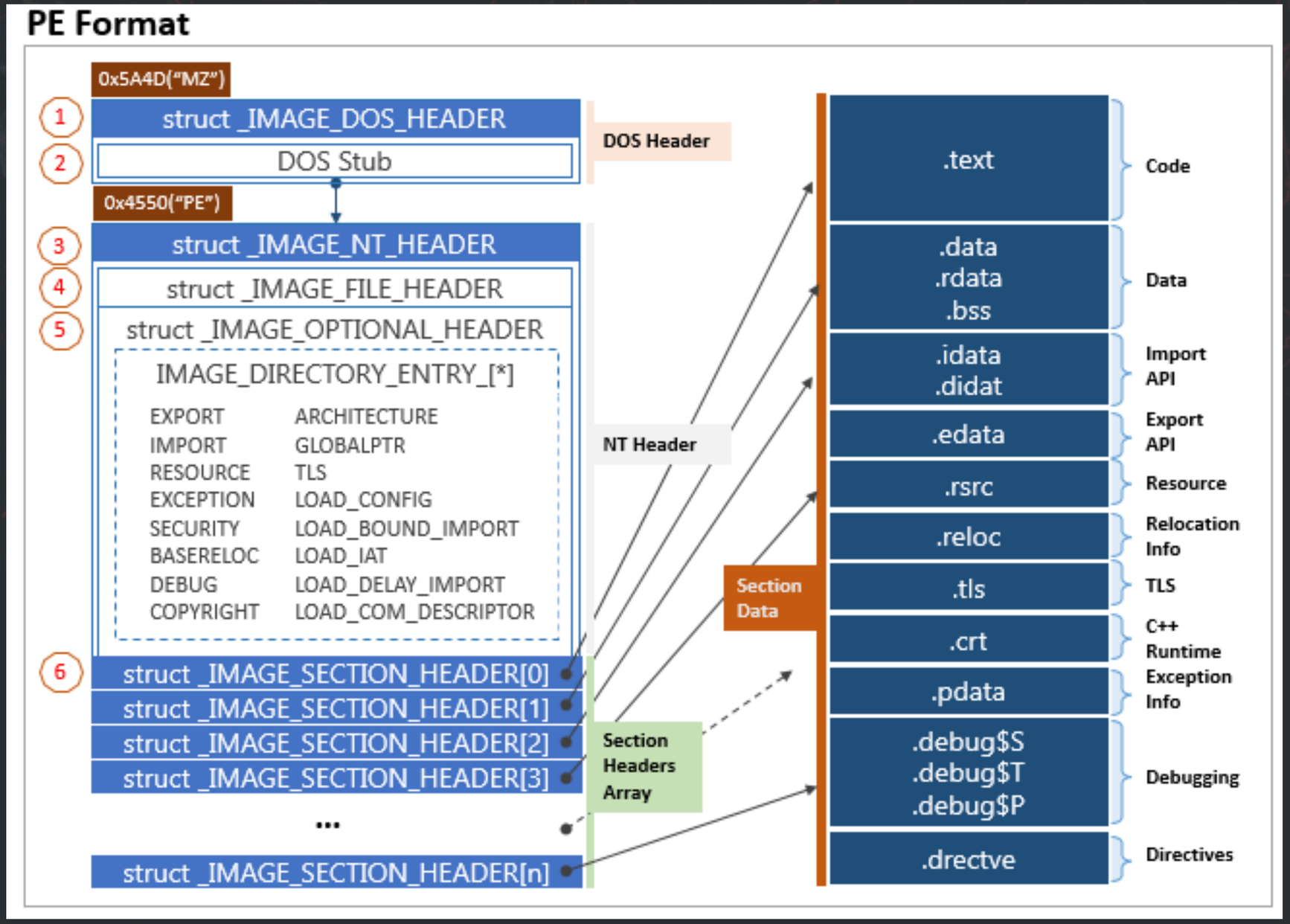
Linker

```
C:\Users\exploit\Desktop\TwTech_Rev
λ gcc -m32 hellWorld.o -o whatTheHell.exe

C:\Users\exploit\Desktop\TwTech_Rev
λ .\whatTheHell.exe
Hola, Hell World 123456.

C:\Users\exploit\Desktop\TwTech_Rev
λ
```

Linker


PE Format
0x5A4D("MZ")
1
struct _IMAGE_DOS_HEADER
2
DOS Stub
DOS Header
0x4550("PE")
3
struct _IMAGE_NT_HEADER
4
struct _IMAGE_FILE_HEADER
5
struct _IMAGE_OPTIONAL_HEADER
IMAGE_DIRECTORY_ENTRY_[*]
EXPORT
IMPORT
RESOURCE
EXCEPTION
SECURITY
BASERELOC
DEBUG
COPYRIGHT
ARCHITECTURE
GLOBALPTR
TLS
LOAD_CONFIG
LOAD_BOUND_IMPORT
LOAD_IAT
LOAD_DELAY_IMPORT
LOAD_COM_DESCRIPTOR
NT Header
6
struct _IMAGE_SECTION_HEADER[0]
struct _IMAGE_SECTION_HEADER[1]
struct _IMAGE_SECTION_HEADER[2]
struct _IMAGE_SECTION_HEADER[3]
...
struct _IMAGE_SECTION_HEADER[n]
Section Headers Array
Section Data
.text
Code
.data
.rdata
.bss
Data
.idata
.didat
Import API
.edata
Export API
.rsrc
Resource
.reloc
Relocation Info
.tls
TLS
.crt
C++ Runtime Exception Info
.pdata
.debug$S
.debug$T
.debug$P
Debugging
.drectve
Directives

cat ./a.o

COFF Overview

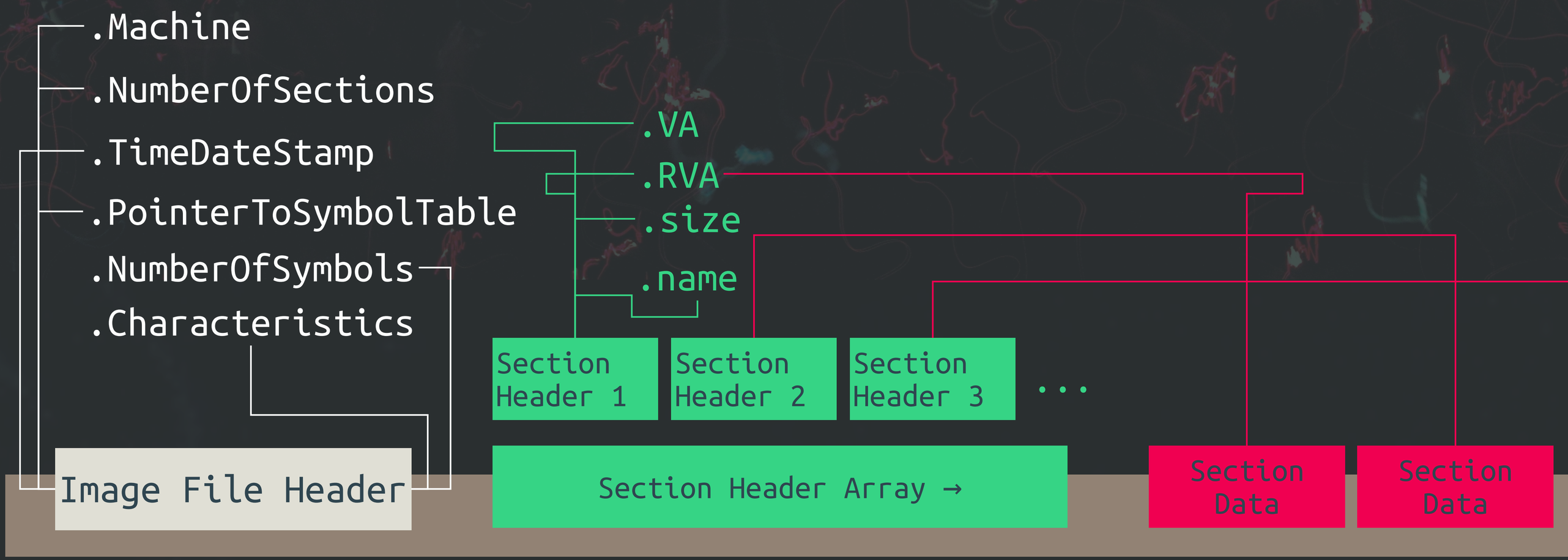
```
struct _IMAGE_FILE_HEADER {
  WORD  Machine;
  WORD  NumberOfSections;
  DWORD TimeDateStamp;
  DWORD PointerToSymbolTable;
  DWORD NumberOfSymbols;
  WORD  SizeOfOptionalHeader;
  WORD  Characteristics;
} IMAGE_FILE_HEADER;
```

```
typedef struct _IMAGE_SECTION_HEADER {
  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
  union {
    DWORD PhysicalAddress;
    DWORD VirtualSize;
  } Misc;
  DWORD VirtualAddress;
  DWORD SizeOfRawData;
  DWORD PointerToRawData;
  DWORD PointerToRelocations;
  DWORD PointerToLinenumbers;
  WORD  NumberOfRelocations;
  WORD  NumberOfLinenumbers;
  DWORD Characteristics;
} IMAGE_SECTION_HEADER;
```

aaaddress1@chroot.org

cat ./a.o

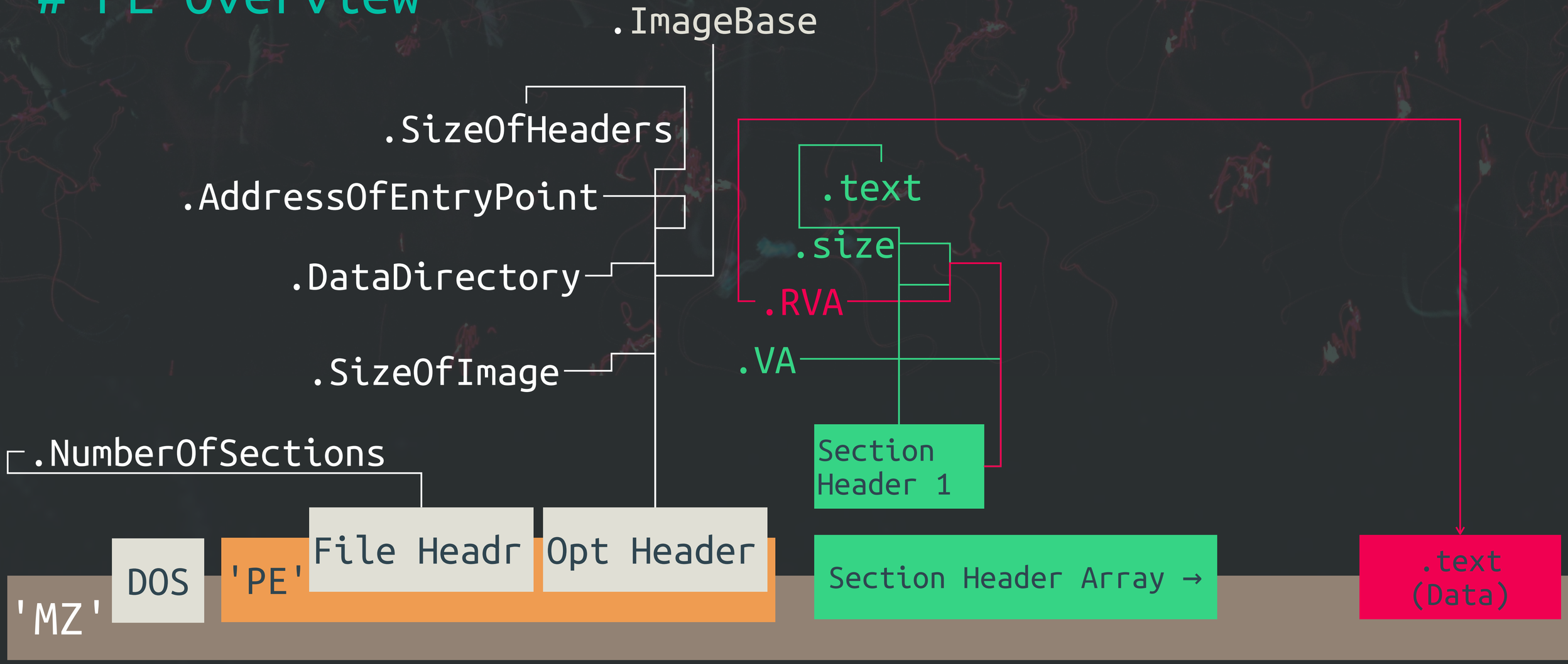
COFF Overview



aaaddress1@chroot.org

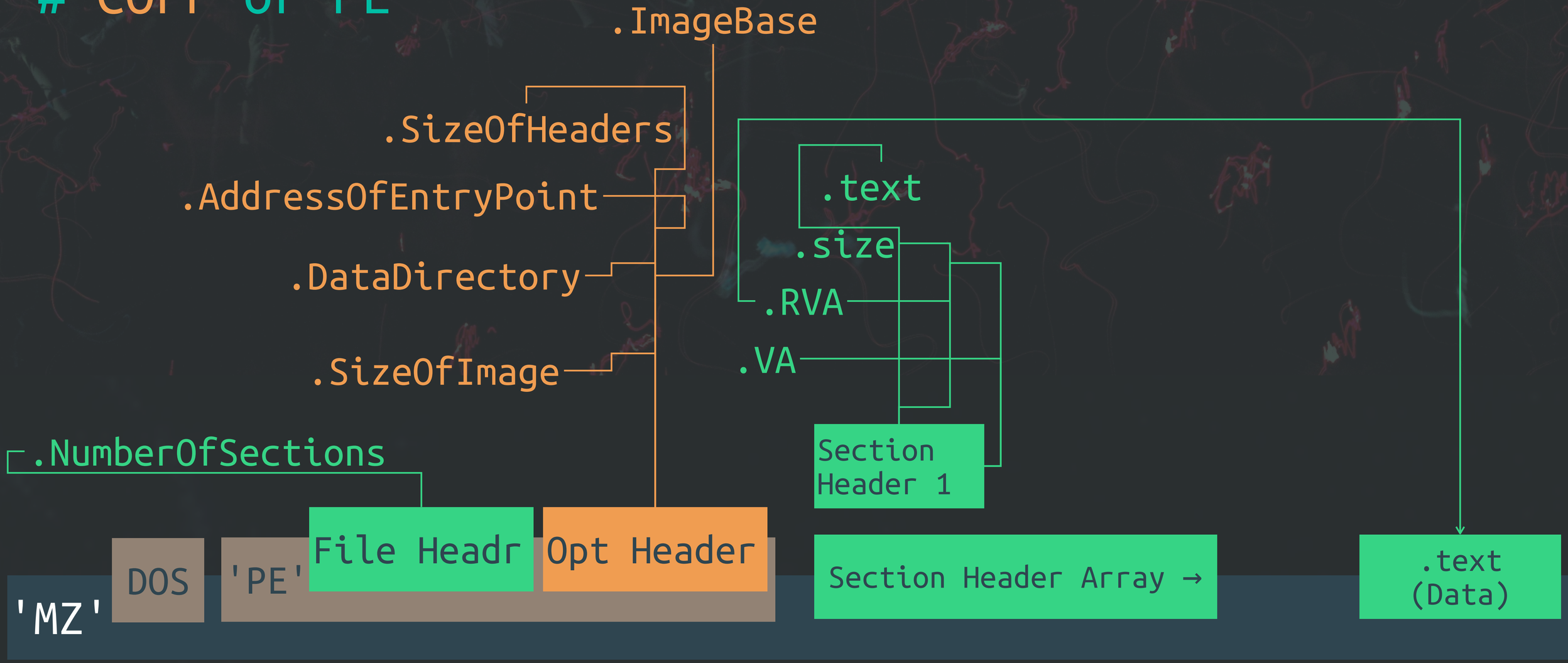
cat ./PE

PE Overview



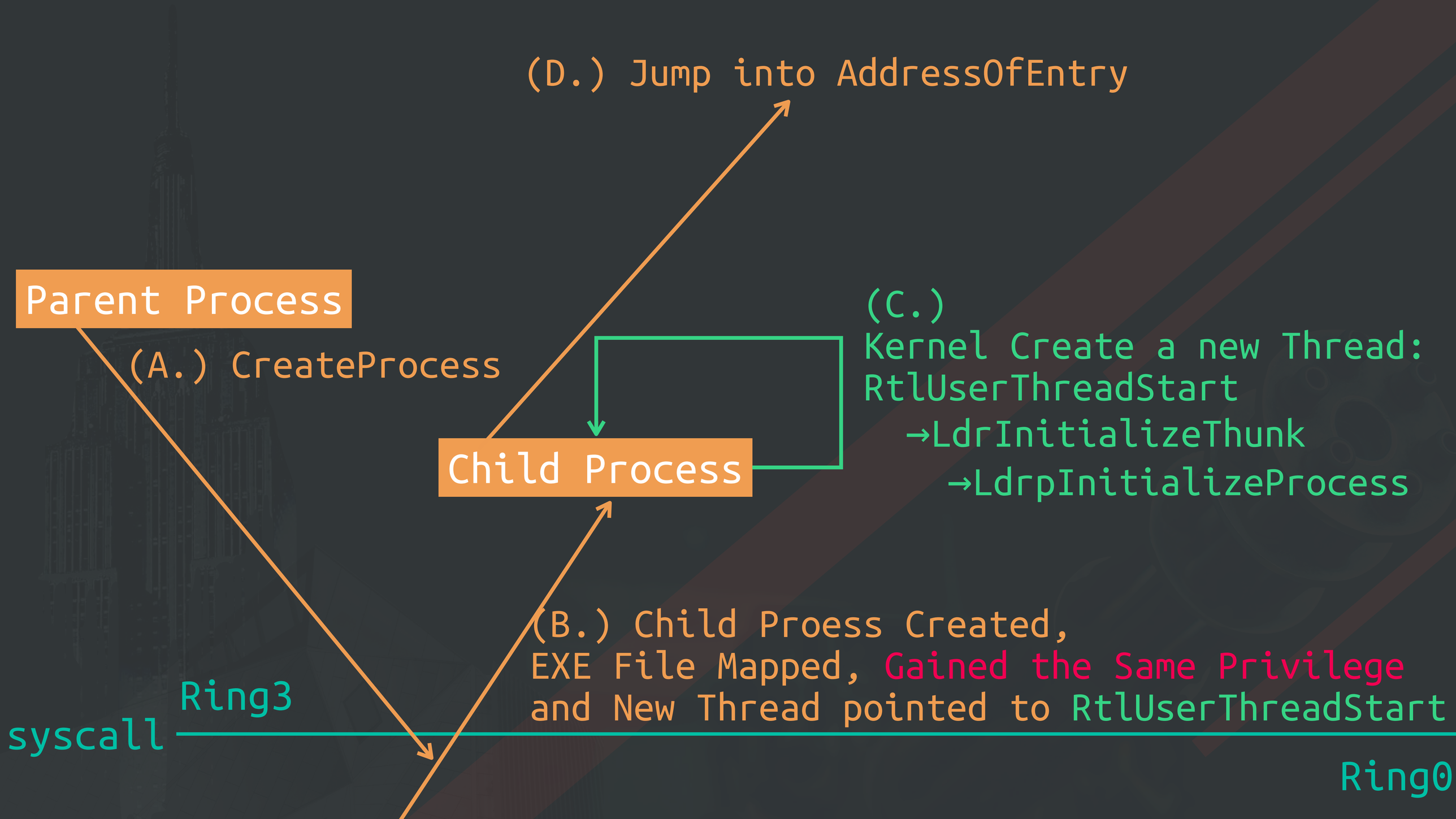
cat ./PE

COFF or PE

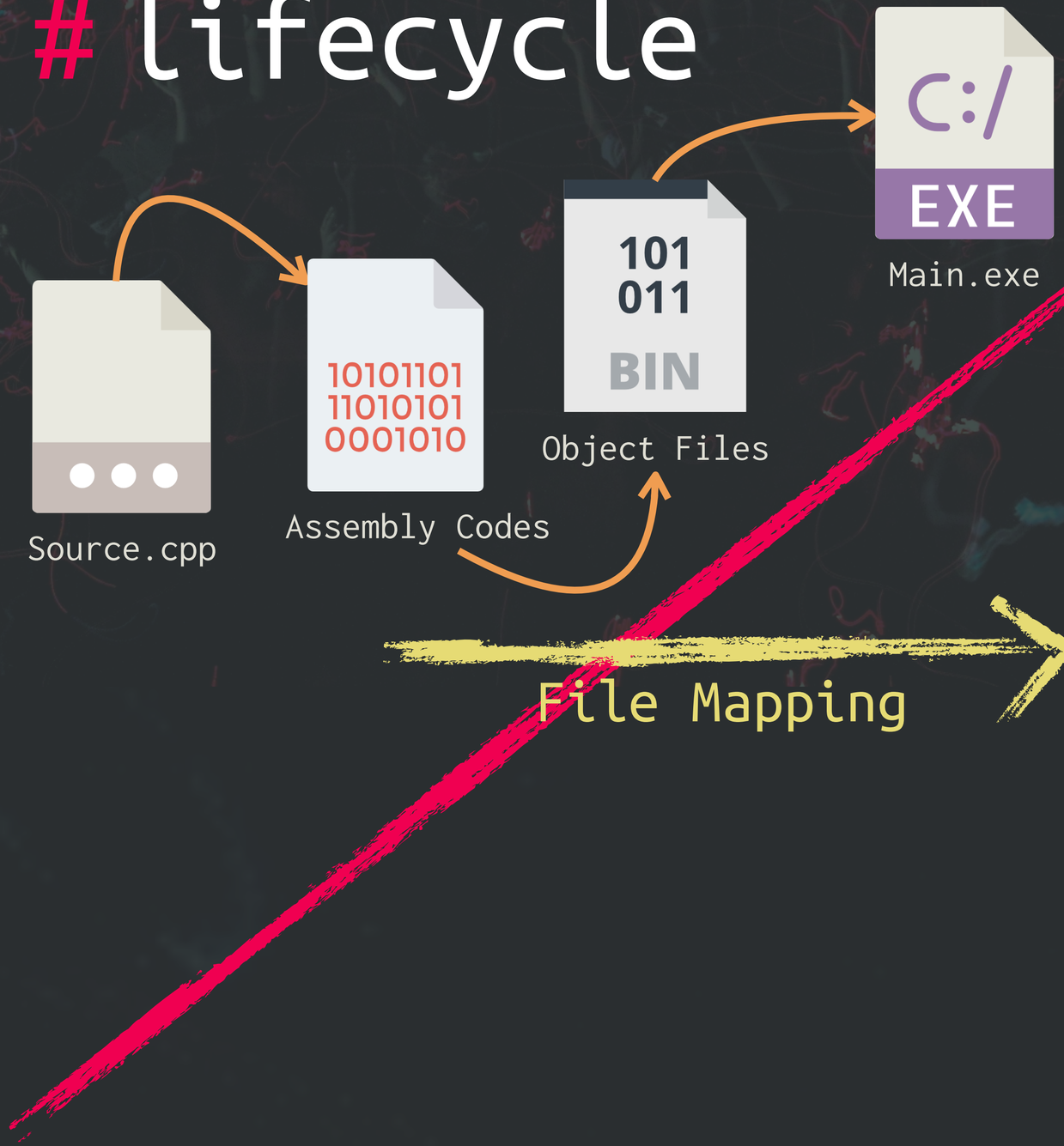


aaaddress1@chroot.org

cd Win32 Process


(D.) Jump into AddressOfEntry
Parent Process
(A.) CreateProcess
Child Process
(C.)
Kernel Create a new Thread:
RtlUserThreadStart
→LdrInitializeThunk
→LdrpInitializeProcess
(B.) Child Proess Created,
EXE File Mapped, Gained the Same Privilege
and New Thread pointed to RtlUserThreadStart
Ring3
syscall
Ring0

lifecycle



Process

Stack Memory

PE Module

NT Header

File Header

Optional Header

Section Header Array

Section[0]: .text

Section[1]: .data

Section[2]: .rdata

. . .

[DATA] .text

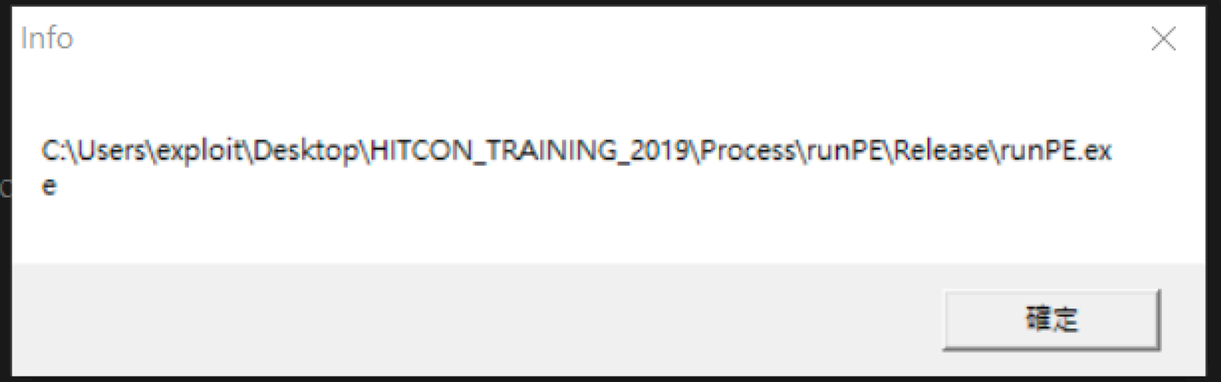
[DATA] .data

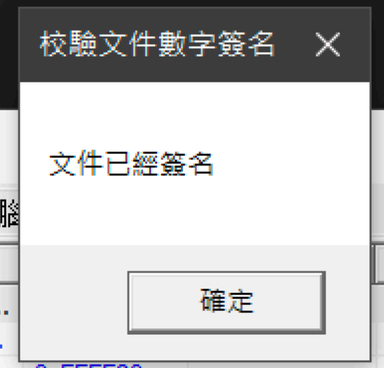
[DATA] .idata

cat ./RunPE

Process Hollowing

```
int CALLBACK WinMain(
    HINSTANCE   hInstance,
    HINSTANCE   hPrevInstanc
    LPSTR       lpCmdLine,
    int         nCmdShow
)
{
    char CurrentFilePath[MAX_PATH + 1];
    GetModuleFileNameA(0, CurrentFilePath, MAX_PATH);
    MessageBoxA(0, CurrentFilePath, "Info", 0);
```





lifecycle

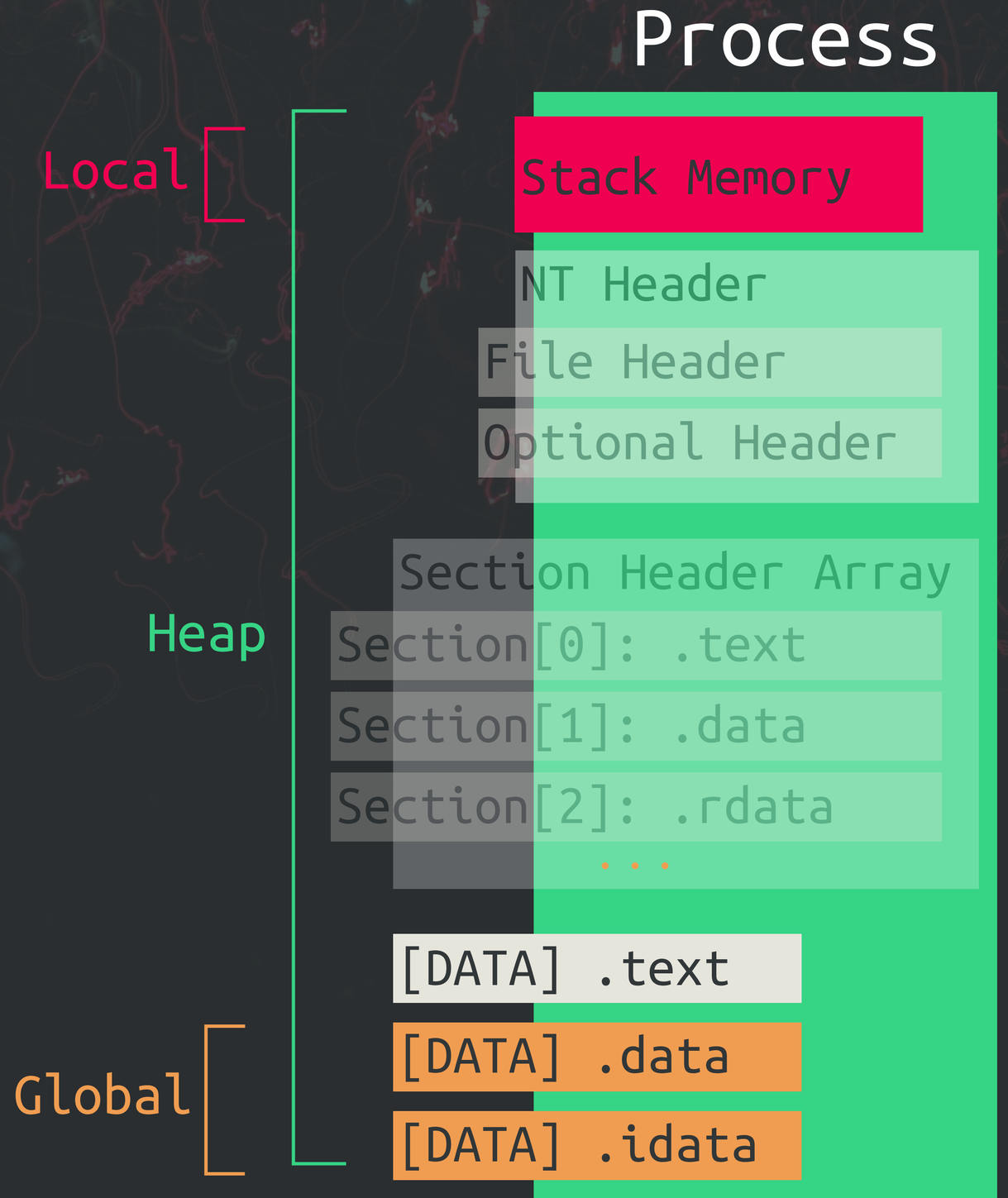
```cpp
#include <cstdio>

int globalNum = 123;
char strHell[] = "Hell";

int strToInt(char* strNum) {
  int v = 0;
  while (*strNum) v = ( 10*v + *strNum++-'0' );
  return v;
}

int main(void) {
  char strLocalNum[] = "456";
  int localNum = strToInt(strLocalNum);

  printf("Hola, %s World %i%i.\n",
    strHell,
    globalNum,
    localNum);
  return 0;
}
```



lifecycle

```
int v = 0;
```

```
char strLocalNum[] = "456";
int localNum = strToInt(strLocalNum);
```

Local

```
3  int globalNum = 123;
4  char strHell[] = "Hell";
5
6  int strToInt(char* strNum) {
7    int v = 0;
8    while (*strNum) v = ( 10*v + *strNum++-'0' );
9    return v;
10 }
11
12 int main(void) {
13   char strLocalNum[] = "456";
14   int localNum = strToInt(strLocalNum);
15
16   printf("Hola, %s World %i%i.\n",
17     strHell,
18     globalNum,
19     localNum);
20   return 0;
21 }
```

Heap

```
int main(void)
int strToInt(char* strNum)
```

```
"Hola, %s World %i%i.\n"
```

```
int globalNum = 123;
char strHell[] = "Hell";
```

Global

Process

Stack Memory

NT Header

File Header

Optional Header

Section Header Array

Section[0]: .text

Section[1]: .data

Section[2]: .rdata

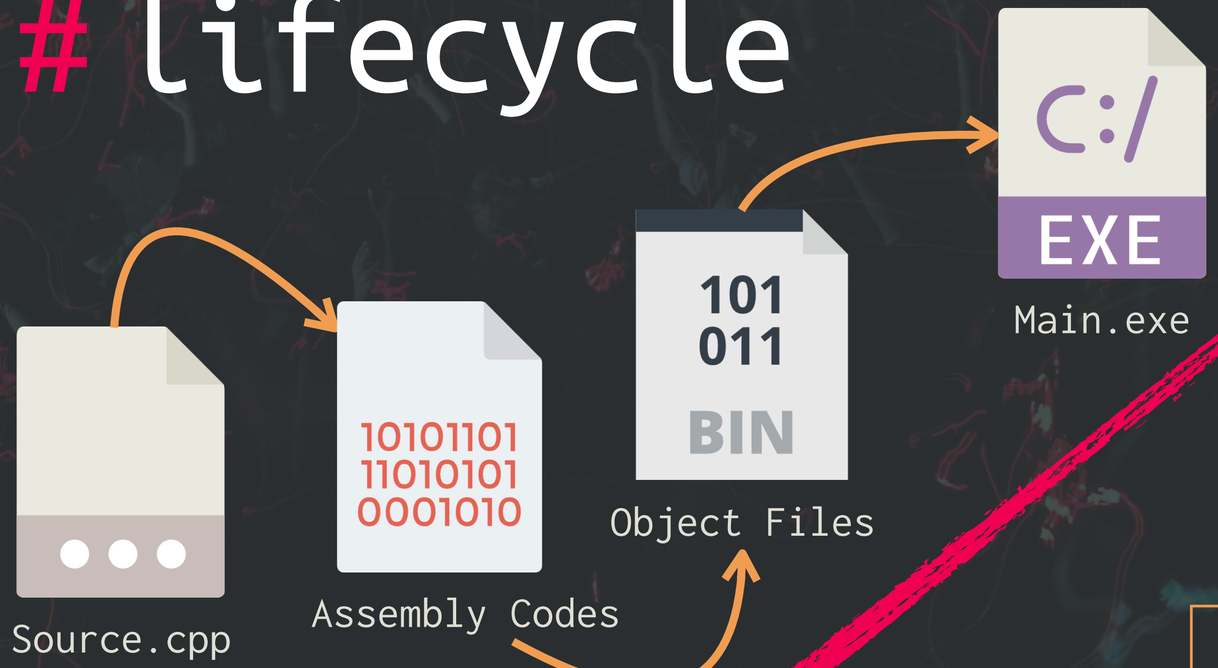
...

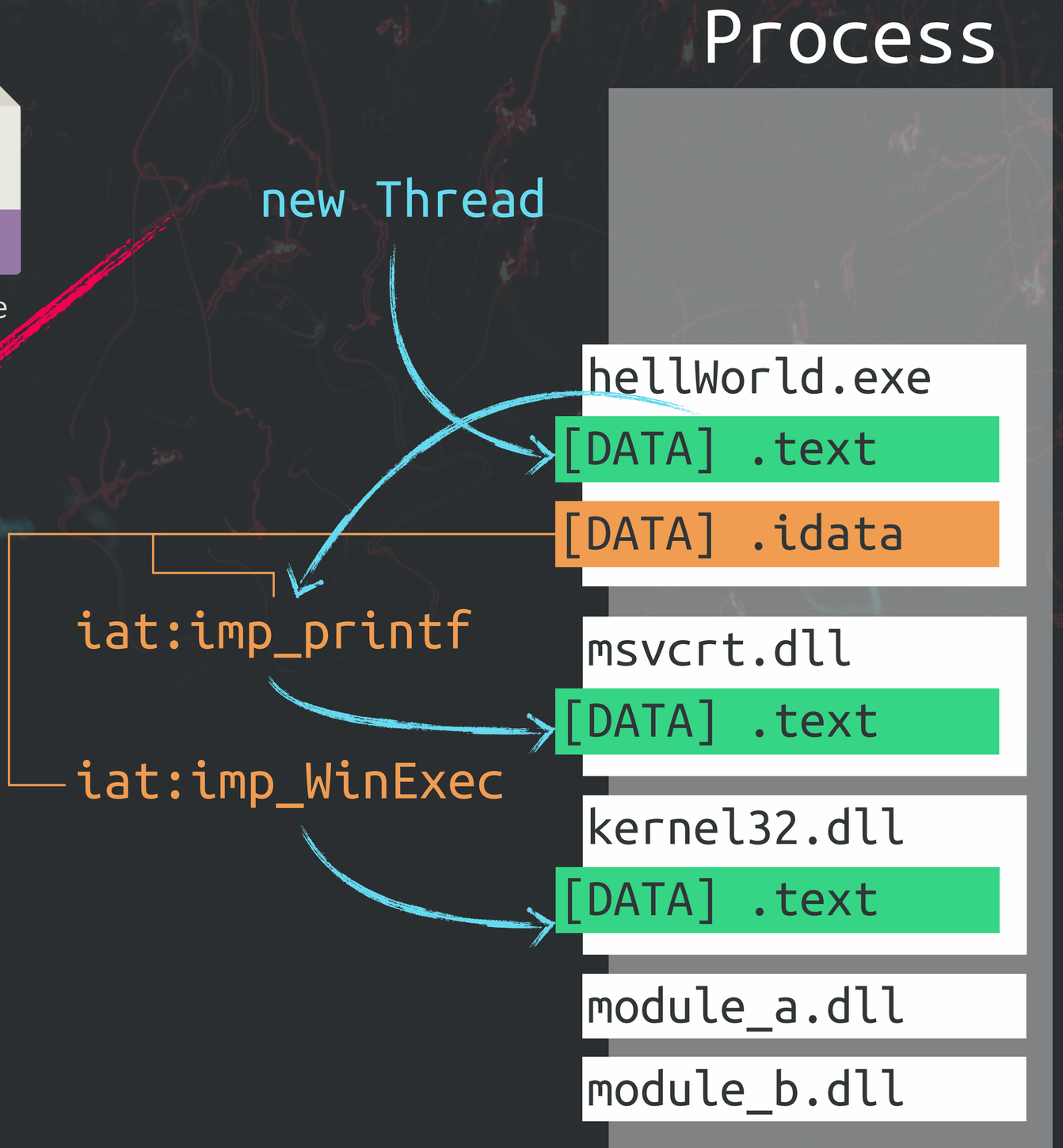
[DATA] .text

[DATA] .data

[DATA] .idata

lifecycle


C:/
EXE
Main.exe
101
011
BIN
Object Files
10101101
11010101
0001010
Assembly Codes
Source.cpp
File Mapping


Process
new Thread
helloWorld.exe
[DATA] .text
[DATA] .idata
iat:imp_printf
iat:imp_WinExec
msvcrt.dll
[DATA] .text
kernel32.dll
[DATA] .text
module_a.dll
module_b.dll

lifecycle

我怎麼知道這一次是哪個模組的執行緒啦，森77。

分時多工

Thread[0]

Thread[1]

Thread[2]

Process

hellWorld.exe

[DATA] .text

[DATA] .idata

msvcrt.dll

[DATA] .text

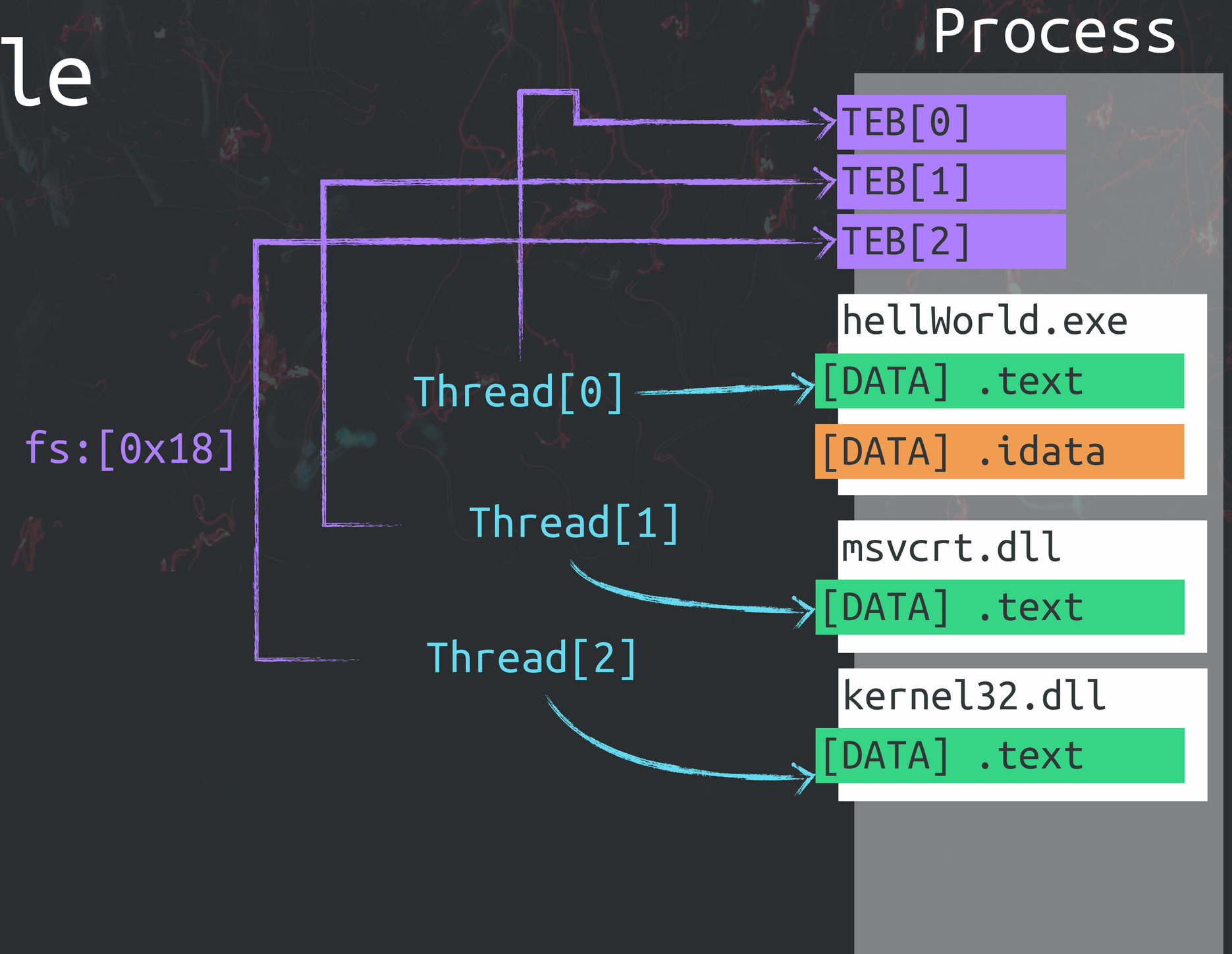
kernel32.dll

[DATA] .text

module_a.dll

module_b.dll

lifecycle


Process
TEB[0]
TEB[1]
TEB[2]
fs:[0x18]
Thread[0]
Thread[1]
Thread[2]
hellWorld.exe
[DATA] .text
[DATA] .idata
msvcrt.dll
[DATA] .text
kernel32.dll
[DATA] .text

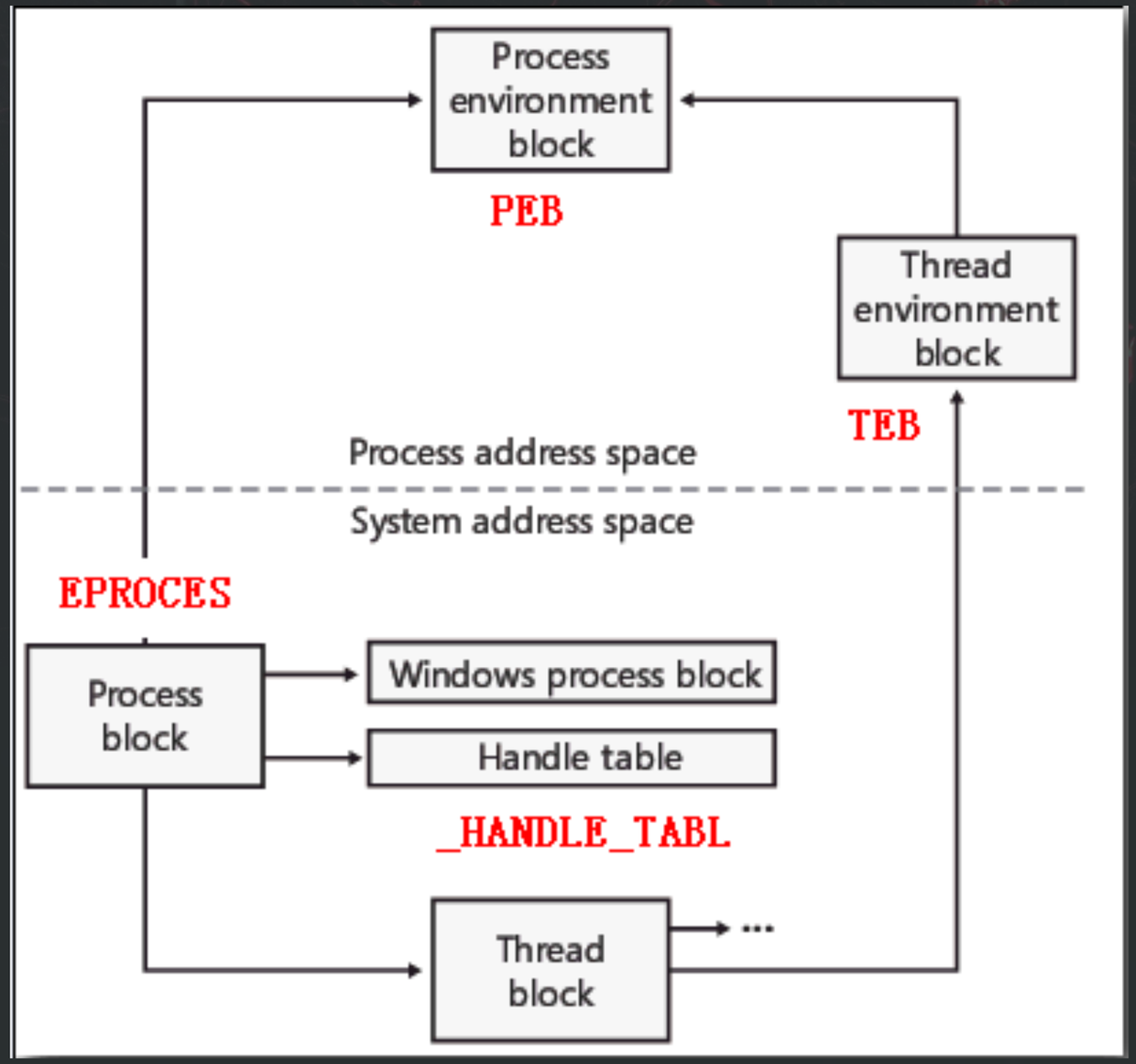
/? TIB

In computing, the Win32 Thread Information Block (TIB) is a data structure in Win32 on x86 that stores information about the currently running thread. This structure is also known as the Thread Environment Block (TEB).

The TIB can be used to get a lot of information on the process without calling Win32 API. Examples include emulating GetLastError(), GetVersion(). Through the pointer to the PEB one can obtain access to the import tables (IAT), process startup arguments, image name, etc. It is accessed from the FS segment register when operating on 32 bits, and from GS in 64 bits.

en.wikipedia.org/wiki/Win32_Thread_Information_Block

/? TIB



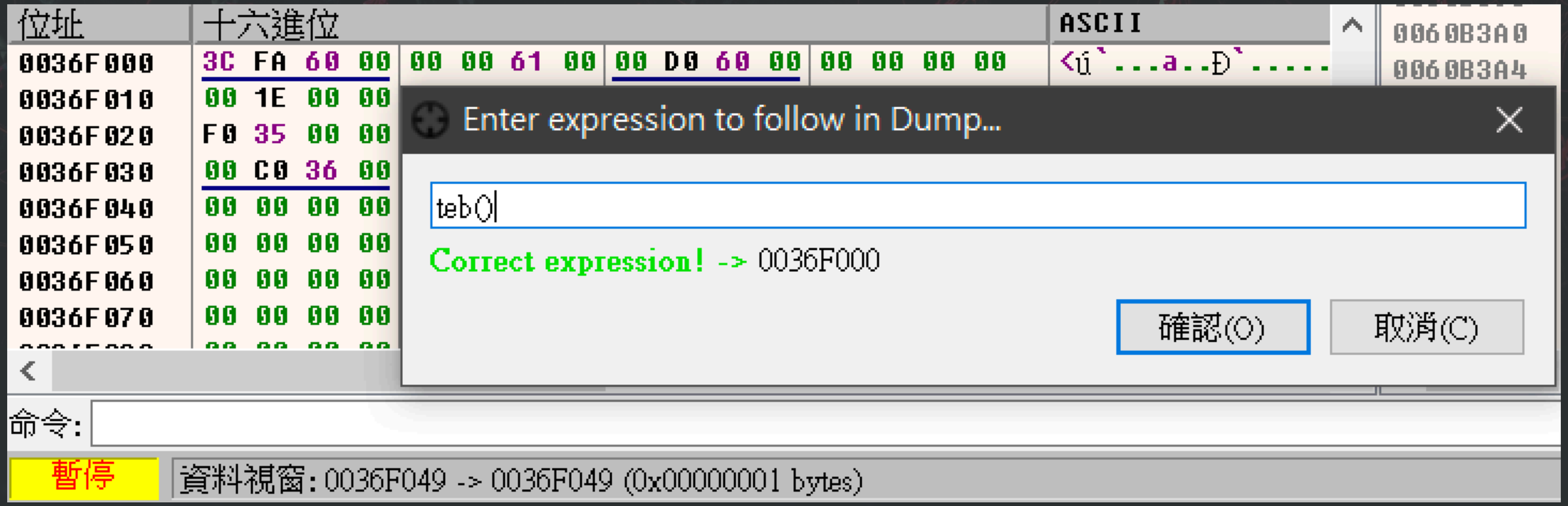
/? TIB # Undocumented

```
struct TEB {
    //NT_TIB structure portion
    EXCEPTION_REGISTRATION*     ExceptionList;  //0x0000 / Current Structured Exception Handling frame
    void*           StackBase;                  //0x0004 / Bottom of stack (high address)
    void*           StackLimit;                 //0x0008 / Ceiling of stack (low address)
    void*           SubSystemTib;               //0x000C
    union {
        void*       FiberData;                  //0x0010
        DWORD       Version;                    //0x0010
    } dword10;
    void*           ArbitraryUserPointer;       //0x0014
    TEB*            Self;                       //0x0018
    //NT_TIB ends (NT subsystem independent part)

    void*           EnvironmentPointer;         //0x001C
    CLIENT_ID       ClientId;                   //0x0020
    //              ClientId.ProcessId          //0x0020 / value retrieved by GetCurrentProcessId()
    //              ClientId.ThreadId           //0x0024 / value retrieved by GetCurrentThreadId()
    void*           ActiveRpcHandle;            //0x0028
    void*           ThreadLocalStoragePointer;  //0x002C
    PEB*            ProcessEnvironmentBlock;    //0x0030
    ...
```

bytepointer.com/resources/tebpeb32.htm

/? x64dbg



/? C$Windows\Sys32\Kernel32

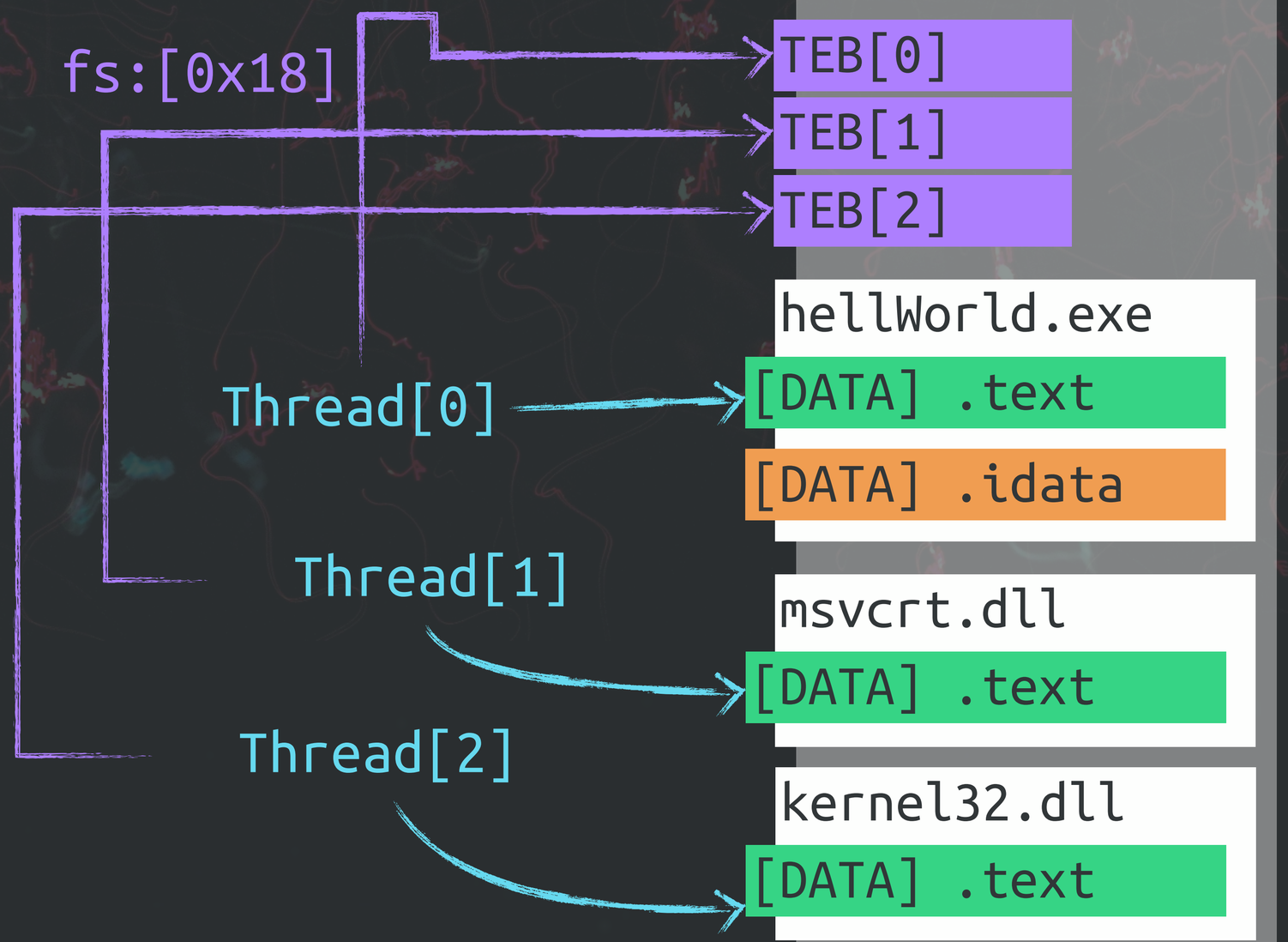
- GetCurrentThread
- GetModuleHandleW
- GetCurrentThreadId
- GetCurrentThread
- IsDebuggerPresent

```
.text:751D8550 ; HANDLE __stdcall GetCurrentThread()
.text:751D8550                 public _GetCurrentThread@0
.text:751D8550 _GetCurrentThread@0 proc near           ; DATA XREF: .rdata
.text:751D8550                 push    0FFFFFFFEh
.text:751D8552                 pop     eax
.text:751D8553                 retn
.text:751D8553 _GetCurrentThread@0 endp
.text:751D8553
.text:751D8553 ; ---------------------------------------------------------
.text:751D8554                 align 10h
.text:751D8560 ; Exported entry 541. GetCurrentThreadId
.text:751D8560
.text:751D8560 ; =============== S U B R O U T I N E =======================
.text:751D8560
.text:751D8560
.text:751D8560 ; DWORD __stdcall GetCurrentThreadId()
.text:751D8560                 public _GetCurrentThreadId@0
.text:751D8560 _GetCurrentThreadId@0 proc near         ; DATA XREF: .rdata
.text:751D8560                 mov     eax, large fs:18h
.text:751D8566                 mov     eax, [eax+24h]
.text:751D8569                 retn
.text:751D8569 _GetCurrentThreadId@0 endp
```

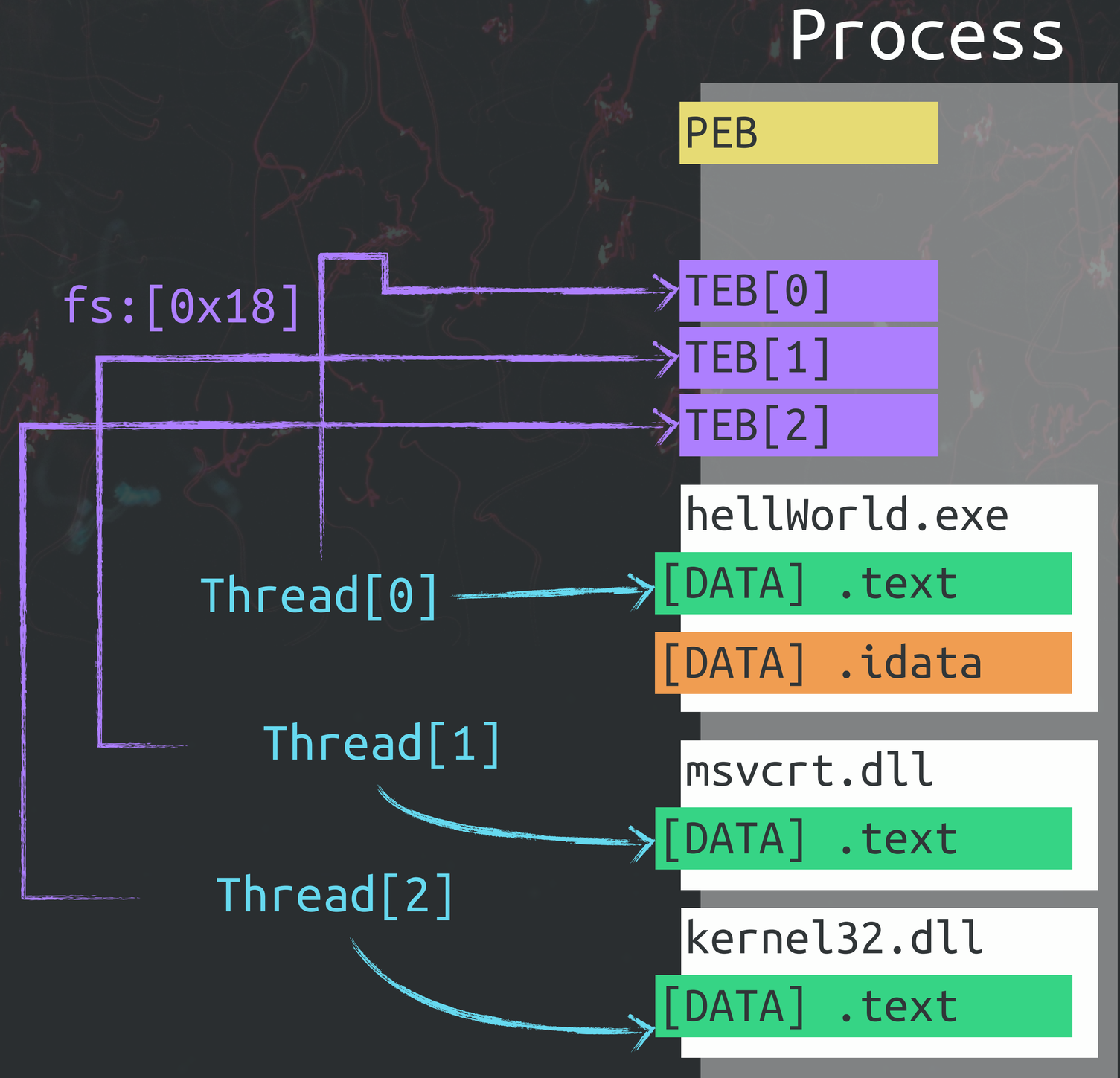
lifecycle

```
struct TEB {
    //NT_TIB structure portion
    EXCEPTION_REGISTRATION*     ExceptionList;
    void*           StackBase;
    void*           StackLimit;
    void*           SubSystemTib;
    union {
        void*       FiberData;
        DWORD       Version;
    } dword10;
    void*           ArbitraryUserPointer;
    TEB*            Self;
    //NT_TIB ends (NT subsystem independent pa

    void*           EnvironmentPointer;         //
    CLIENT_ID       ClientId;                   //
    //              ClientId.ProcessId          //
    //              ClientId.ThreadId           //
    void*           ActiveRpcHandle;            //
    void*           ThreadLocalStoragePointer;  //
    PEB*            ProcessEnvironmentBlock;    //
    ...
```



lifecycle


Process
PEB
fs:[0x18]
TEB[0]
TEB[1]
TEB[2]
Thread[0]
Thread[1]
Thread[2]
hellWorld.exe
[DATA] .text
[DATA] .idata
msvcrt.dll
[DATA] .text
kernel32.dll
[DATA] .text

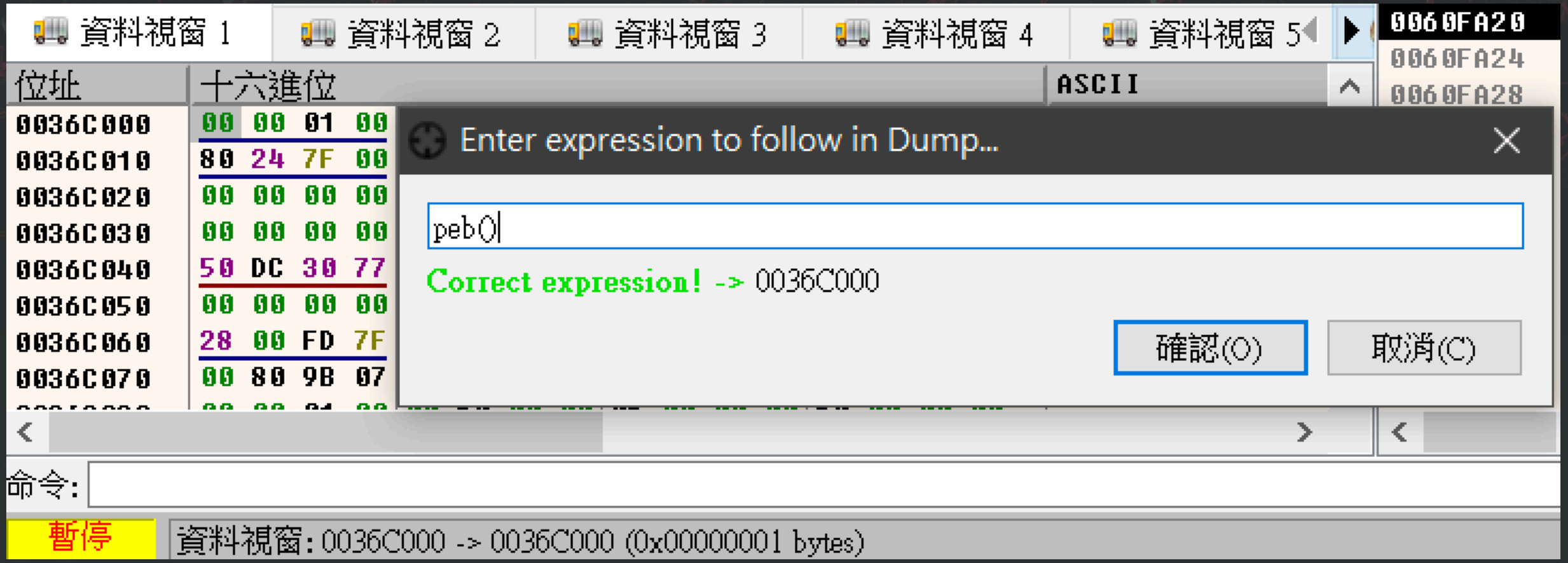
/? PEB

In computing the Process Environment Block (abbreviated PEB) is a data structure in the Windows NT operating system family. It is an opaque data structure that is used by the operating system internally, most of whose fields are not intended for use by anything other than the operating system.

Microsoft notes, in its MSDN Library documentation – which documents only a few of the fields – that the structure "may be altered in future versions of Windows". The PEB contains data structures that apply across a whole process, including global context, startup parameters, data structures for the program image loader, the program image base address, and synchronization objects used to provide mutual exclusion for process-wide data structures.

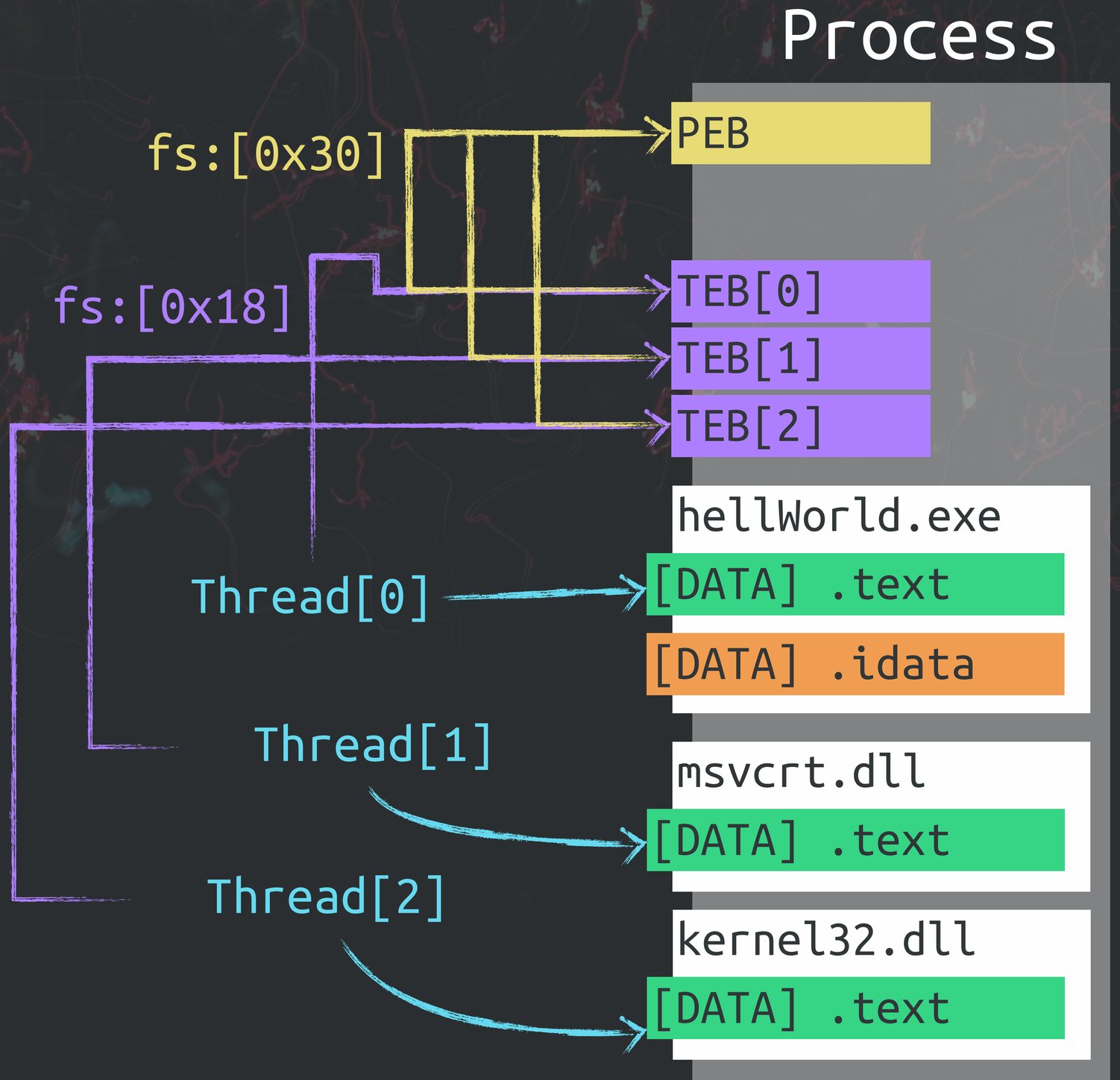
en.wikipedia.org/wiki/Process_Environment_Block

/? x64dbg



lifecycle

```
typedef struct _PEB32 {
  UCHAR InheritedAddressSpace;
  UCHAR ReadImageFileExecOptions;
  UCHAR BeingDebugged;
  UCHAR BitField;
  ULONG Mutant;
  ULONG ImageBaseAddress;
  PPEB_LDR_DATA Ldr;
  ULONG ProcessParameters;
  ULONG SubSystemData;
  ULONG ProcessHeap;
  ULONG FastPebLock;
  ULONG AtlThunkSListPtr;
  ULONG IFEOKey;
  ULONG CrossProcessFlags;
  ULONG UserSharedInfoPtr;
  ULONG SystemReserved;
  ULONG AtlThunkSListPtr32;
  ULONG ApiSetMap;
} PEB32, *PPEB32;
```



/? homework

Back To The Future

```
C:\Users\exploit\Desktop\TwTech_Rev\BackTo1985
λ KeyChecker_patched.exe
 --------------------------
 | B@ck t0 7he Fu7ur3...
 | en.wikipedia.org/wiki/Back_to_the_Future
  --------------------------
[+] It's a time machine built in 1985,
        and you're in 1985 year now.
[!] Time Machine Guarder: [SAFE]
[+] input password to launch time machine: [redacted]
[!] reading ... the.... passw0r..d.....
[+] a flag found by time machine at 1985:
        FLAG{[redacted]

C:\Users\exploit\Desktop\TwTech_Rev\BackTo1985
λ
```

Windows Reversing Basic

aaaddress1@chroot.org

台灣科技大學 資安實務