Encryption Wrapper on OSX

Overview

- OSX Kernel
- What is an Encryption Wrapper?
- Project Implementation

OSX's Origins and Design

NeXTSTEP Legacy

NextStep v1

NextStep v3.3

Mac OS X

Jobs creates NeXT

Apple acquisition

PPC Rhapsody

Mac OS X i386

1985 1989 1995 2001 2005

Workspace
Info
File
Edit
Disk
View
Tools
Windows
Services
Hide h
Log Out q

Disk
Eject e
Initialize...
Check for Disks

File Viewer
paul
tmp
LocalApps
ibmsaa.art
68.6MB available on remote disk
home
paul
Adv.NetInfo.rtf
david
hard_disk.hdf
liz
mailtest
news-archive
orders
paul
Adv.NetInfo.rtf
AppMan.diagram2
Apps
A.SoftPC.local....
Barclays
Bargains.Propos...
calendar
CISMail
CISnews
cmos.ram

Delete
4435 messages (60.2MB) — 31 d
4399 Jan 18 To: li
4402 Jan 18 Eliza
4403 Jan 18 Eliza
4404 Jan 18 Eliza
19 To: A
19 To: A

Attributes Inspector
Attributes
Adv.NetInfo.rtf
Path: /home/paul
Link To:
Size: 30.2KB
Owner: paul
Group: wheel
Compute Size
Permissions
Read ✓ ✓ ✓
Write ✓ ✗ ✗
Owner Group Other
Changed
6:30 PM
SAT 22 APR 1995
Revert
OK

roughly:
s ago, Stev
Now he's ba
software th
bought NeX
much as Je
for releasi
pland project, codenamed "Rhapsody". Rhapsody will be based initially on
the operating system acquired with NeXT, and with a development team
ainly of engineers from NeXT, headed by Avadis Tevanian, formerly head of
t at NeXT.
ll run on all current Power PC Macintoshes. The initial release for developers
penStep API (Application Programming Interface), which is part of the
perating system, and the user interface also from NeXTSTEP, with minor
aken from the Macintosh System 7 interface. The full release of Rhapsody
her adaptations of the user interface, will include additional toolkits from

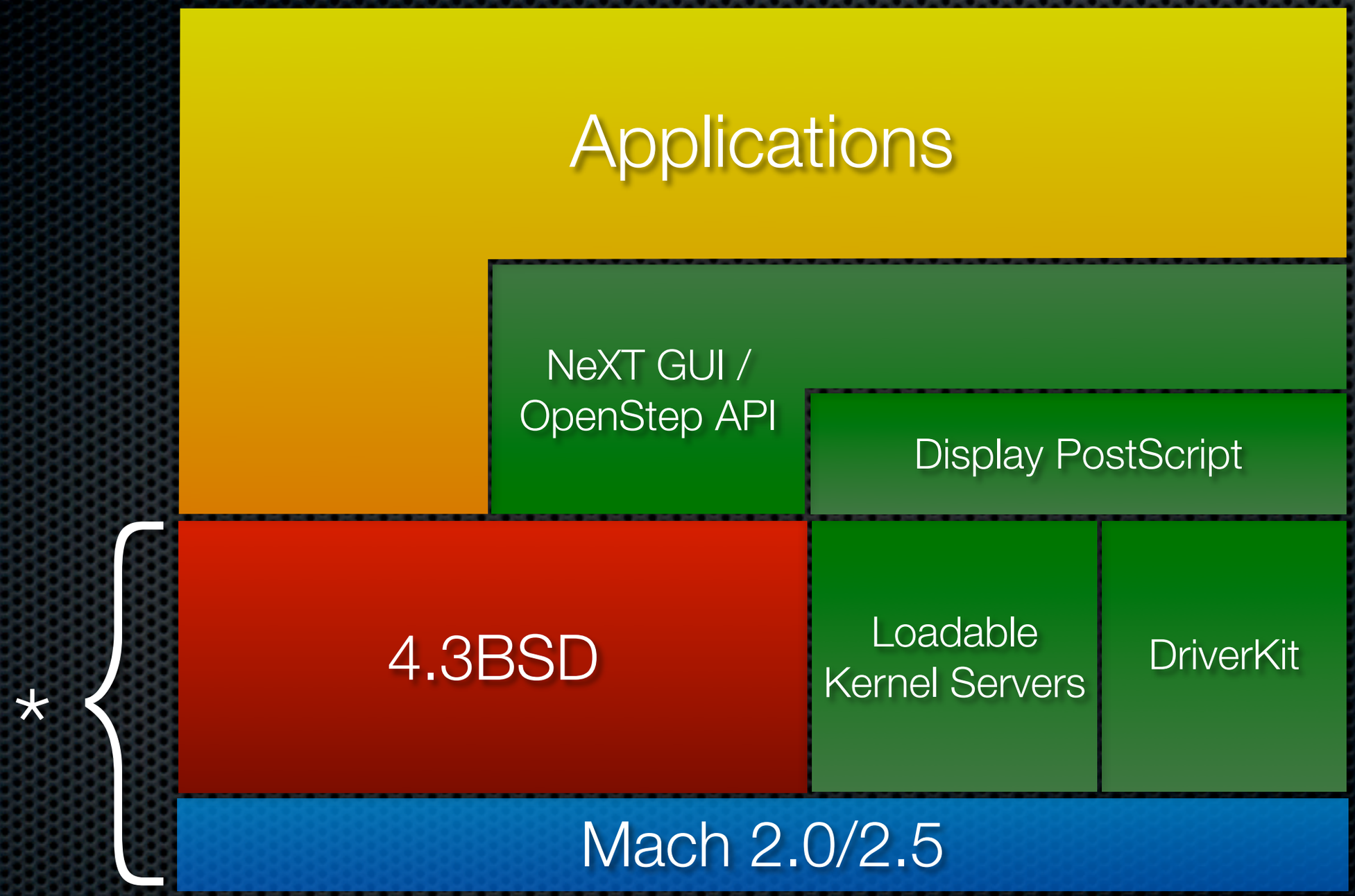
Mailboxes
bbs_post
Create
Dataphile
eof
MagicCap
MagicDev
netinfo-talk
nextdoom
next-admin 5
next-classroom
next-icon
next-jobs
next-managers
NeXT-PPP
next-prog
nsfip-homebrew
nugi
nuukvic
OmniWeb
pdo
ph7
procmail
PRsite
quickbase
Radium
r-thompson 432
SafetyNet
SarNet
uk-next-announce
uk-next-users 3
uk-riders
uk-sun
WebObjects
Name: OmniWeb.mbox
Delete
New
Open
Transfer

8:16 PM
SUN 19 JAN
6

NeXTSTEP Legacy

- Operating System Design
- Object Oriented (APIs, KPIs)
- OpenStep API (now cocoa)
- Objective-C
- Finder :-(

NeXTSTEP Design

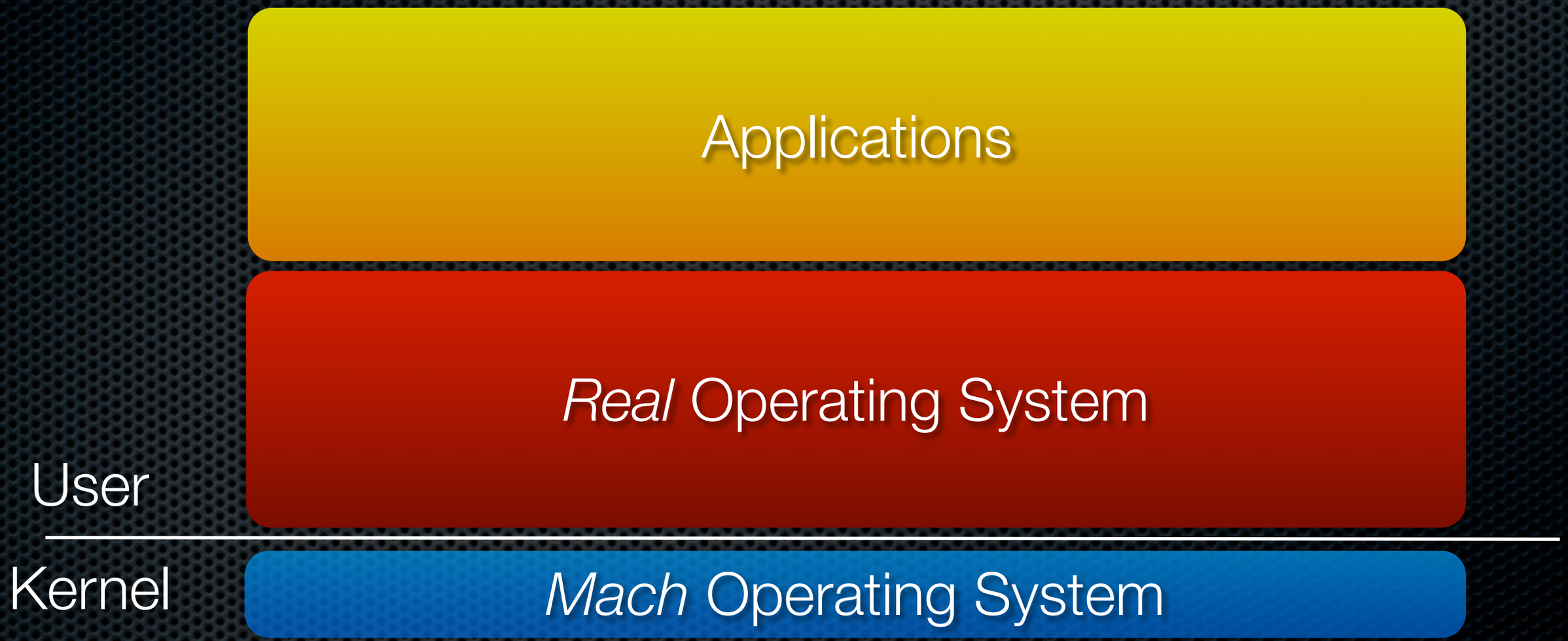


* Berkeley - Carnegie Mellon Universities

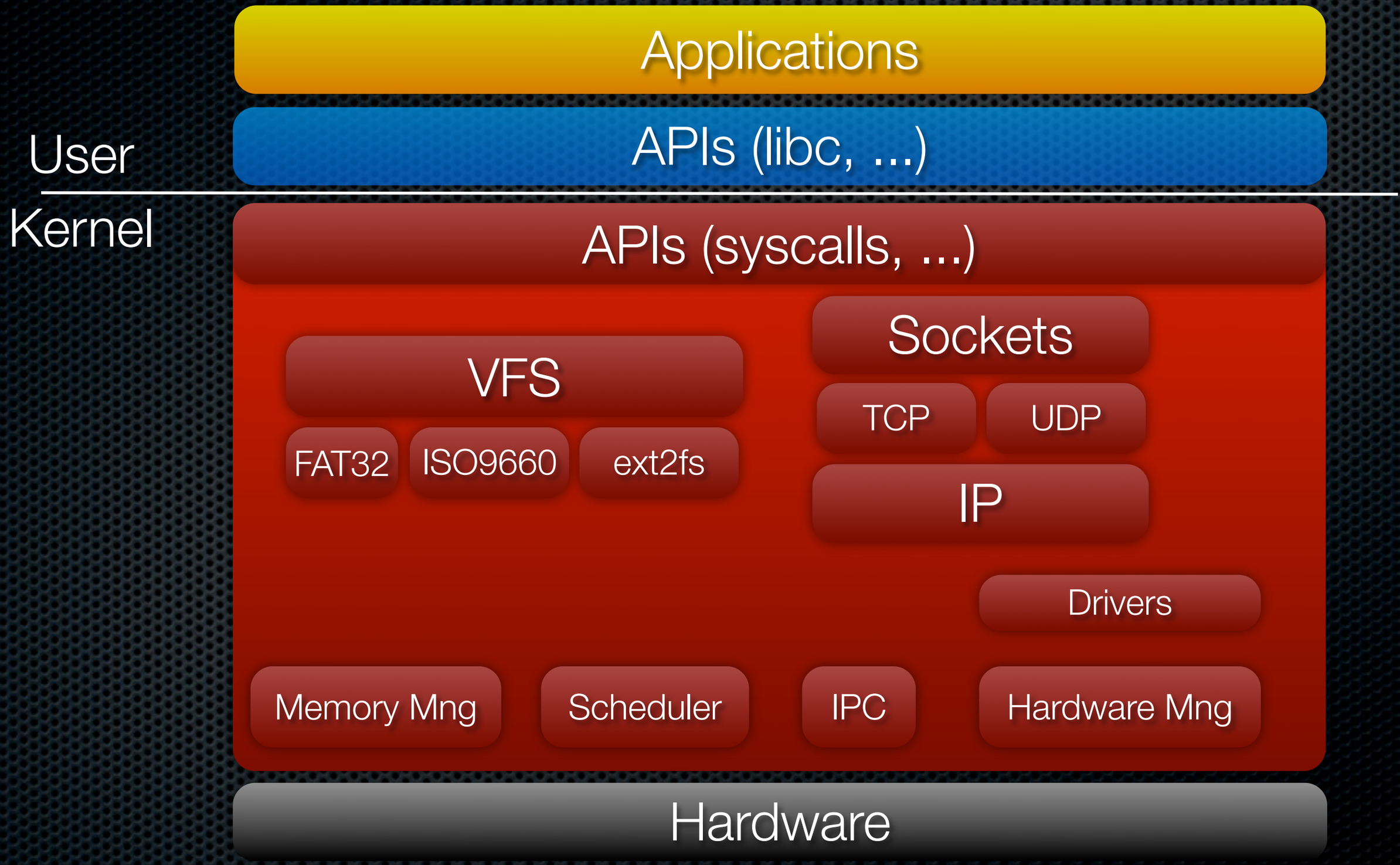
CMU's Mach Original Goals

- *"An operating system to create operating systems"*
- Designed to replace the huge complex buggy Unix kernel
- Promoted portability, stability and simplicity

Mach Original Concept: a ***MicroKernel***



Monolithic Kernel



Mach, a MicroKernel

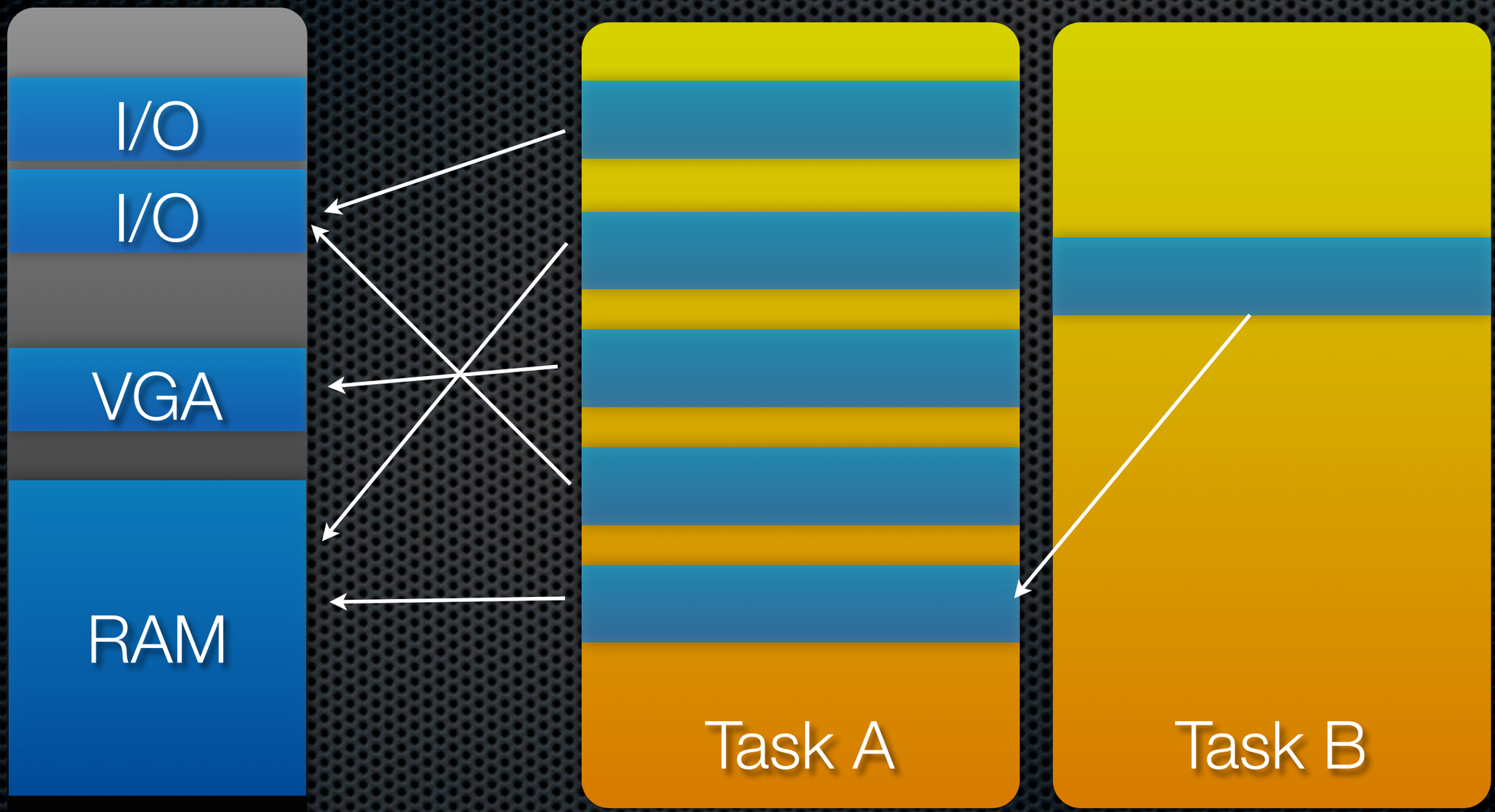
Mach Basics

- Task: simple resources container (VM, ports, threads,...)
- Thread: basic execution unit (shares task's resources with other threads, only owns an execution context)
- Port: message queue protected with privileges
- Message: data that a thread can send or receive
- Memory object: container for data (mapped in the task's virtual address space)

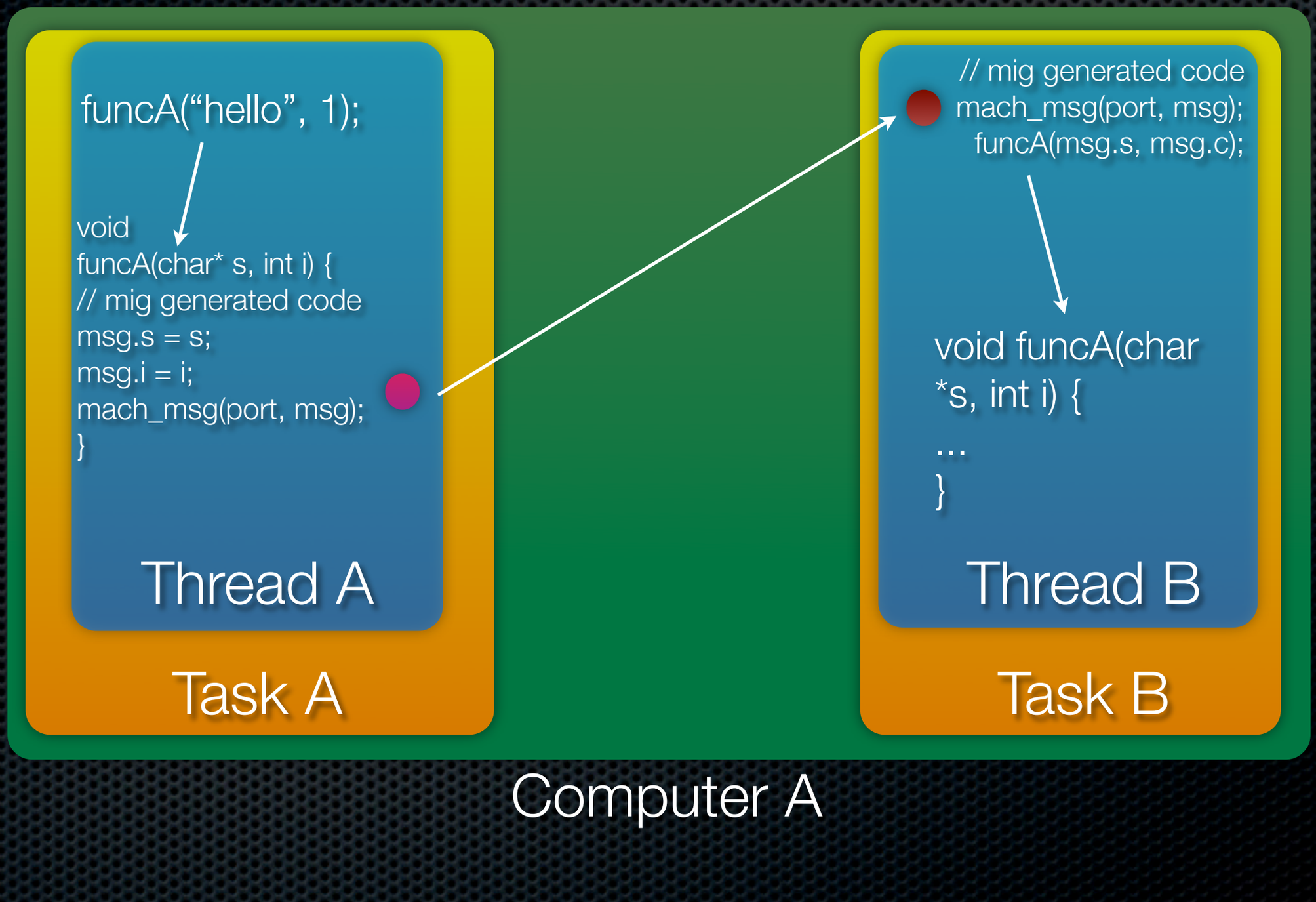
Mach Basics...

- **No** user/group/world notions
- **No** FS notions like working directory
- Pure/simple/secure oriented object concept
  - Only one object can own/receive messages on a specific port
  - A thread needs to have the right on a specific port to send a message to it

Mach Memory Management



Mach IPC (RPC w MIG)

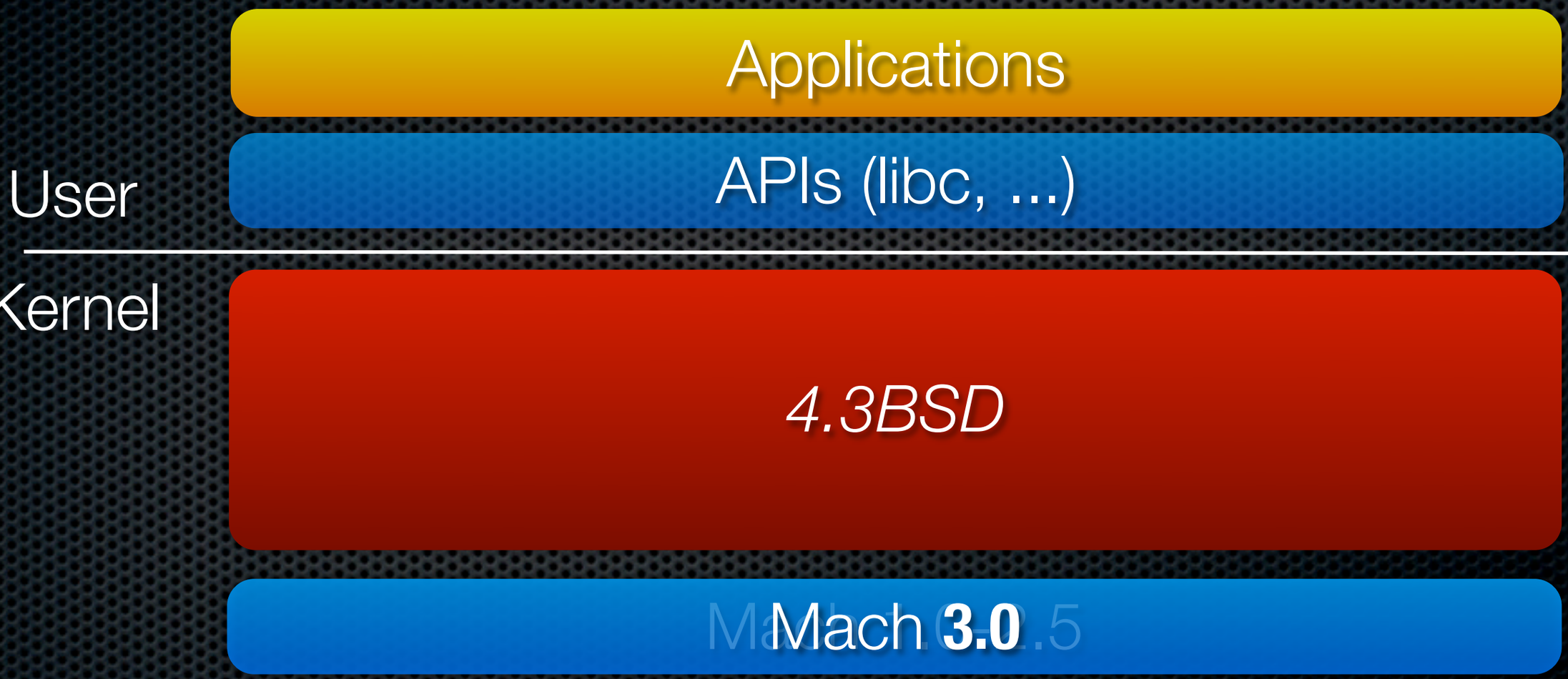


Mach is responsible for...

- Preemptive multitasking (schedule user/kernel threads)
- Interrupt Management
- Protected Memory
- Virtual Memory Management
- IPC
- Real-Time Support
- Debugging support
- Console I/O

From CMU to Apple Implementation

CMU's implementations



NeXTSTEP's implementation

User

Applications

APIs (libc, ...)

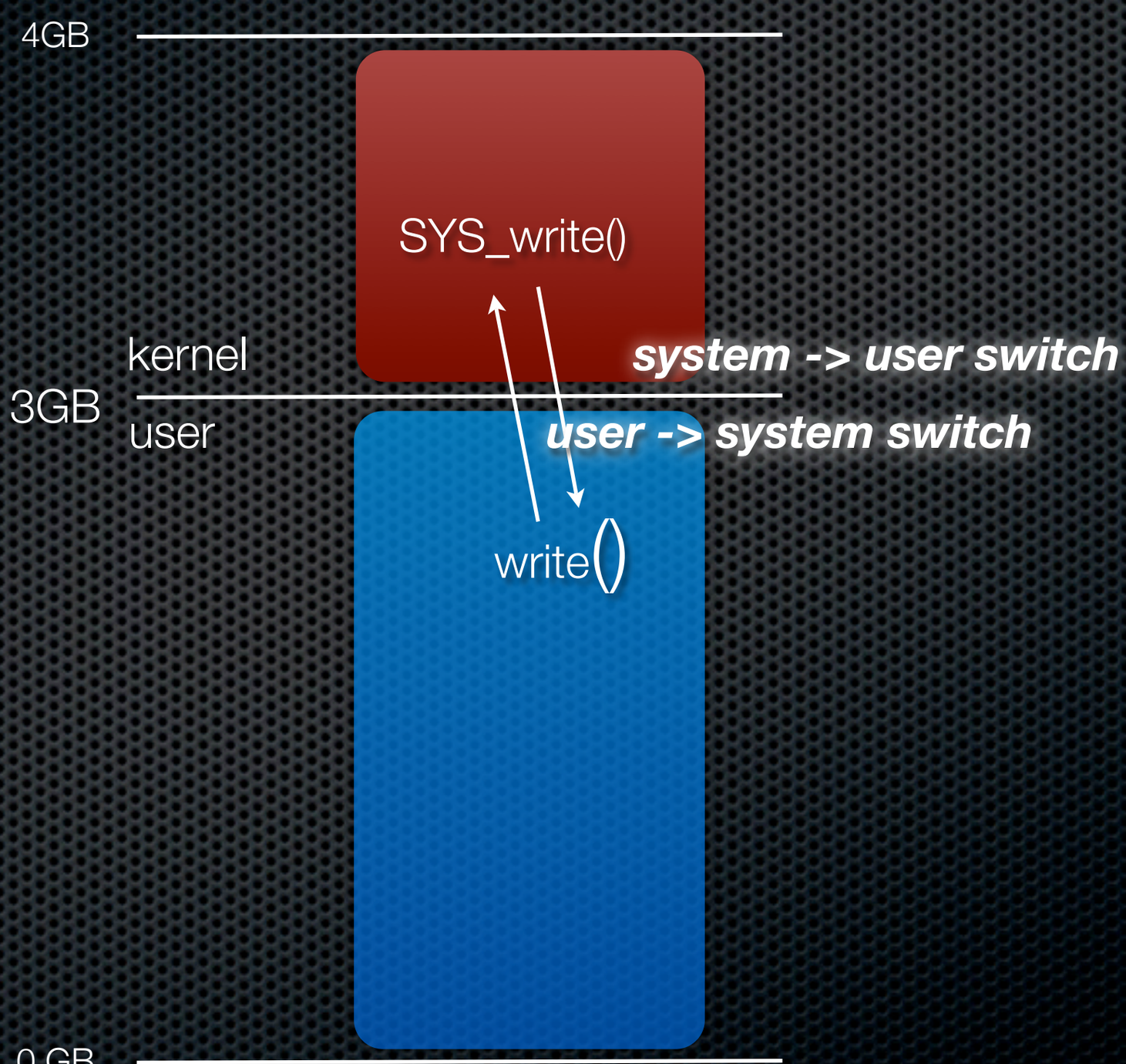
Kernel

FreeBSD

IOKit

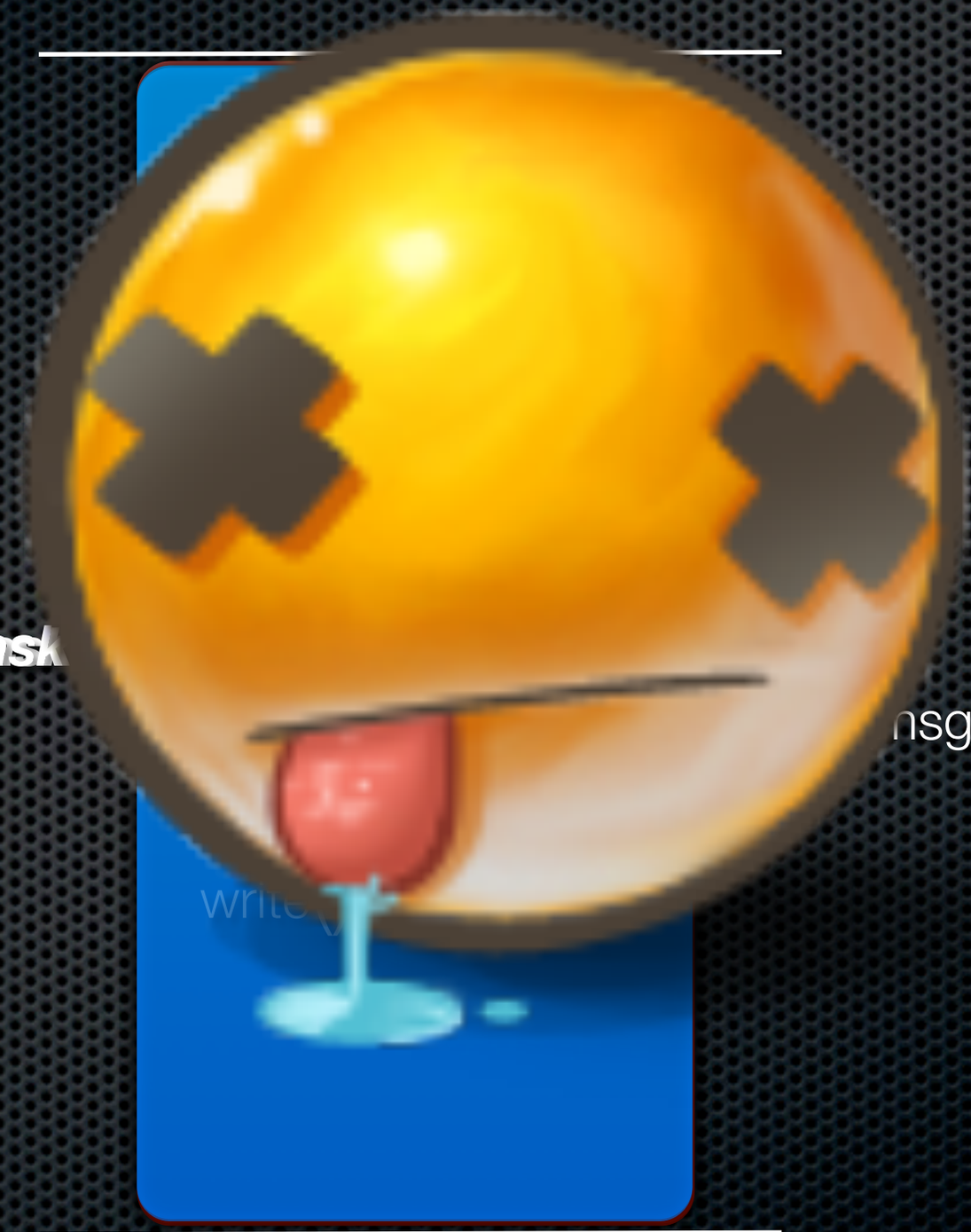
Mach **3.0**

Monolithic system calls (FreeBSD, Linux, ...)



MicroKernel system calls

4GB



Mach kernel task

nsg)

0 GB

task switch

task switch

task switch

task switch

OSX: Colocation

User

Applications

APIs (libc, ...)

Kernel

FreeBSD

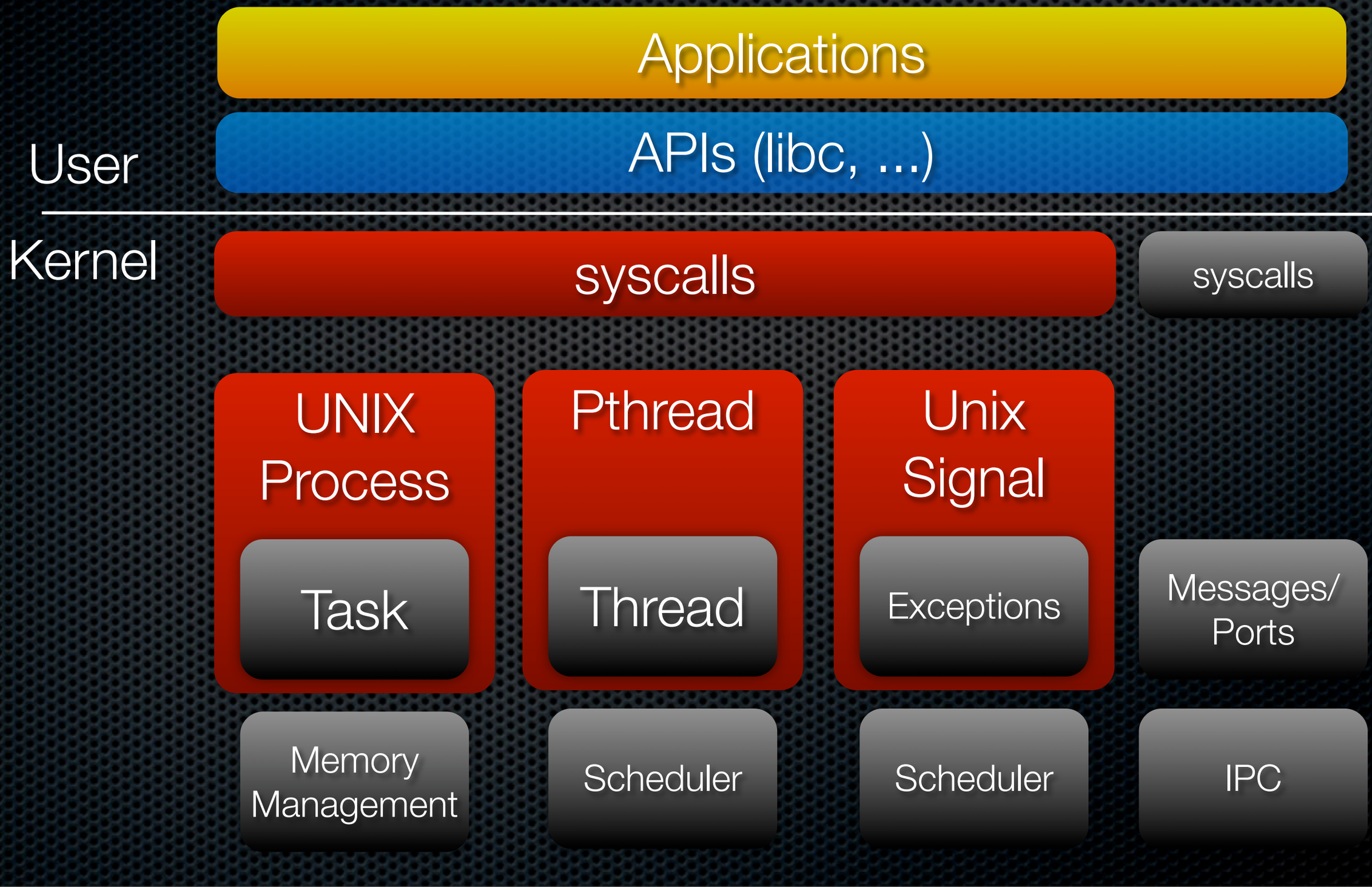
IOKit

Mach **3.0**

BSD on top of Mach

- How to wrap a secure and simple micro-kernel into an insecure and complex monolithic-kernel?

BSD on top of Mach



IOKit

- Driver framework
- Fully object Oriented
- C++ (w/o exceptions, templates and rtti)
- Enable to easily reuse code

IOKit

driver.c

```
static int hello_init(void)
{
  return initlaucher();
}
static void hello_exit(void)
{
}
module_init(hello_init);
module_exit(hello_exit);
```

RocketLauncher
v1.0

driver.c

```
static int hello_init(void)
{
#ifdef v2
  initnewV2stuff();
#endif
  return initlaucher();
}
static void hello_exit(void)
{
}
module_init(hello_init);
module_exit(hello_exit);
```

RocketLauncher
v2

IOKit

driver.cpp

```
bool rocketl::init()
{
   return initlaucher();
}
```

RocketLauncher
v1.0

driver.h

```
class rocketlV2 :
public rocketl
{
};
```

driver.cpp

```
bool rocketlV2::init()
{
  if (initnewV2stuff())
   return rocketl::init();
  return FALSE;
}
```

RocketLauncher
v2

IOKit

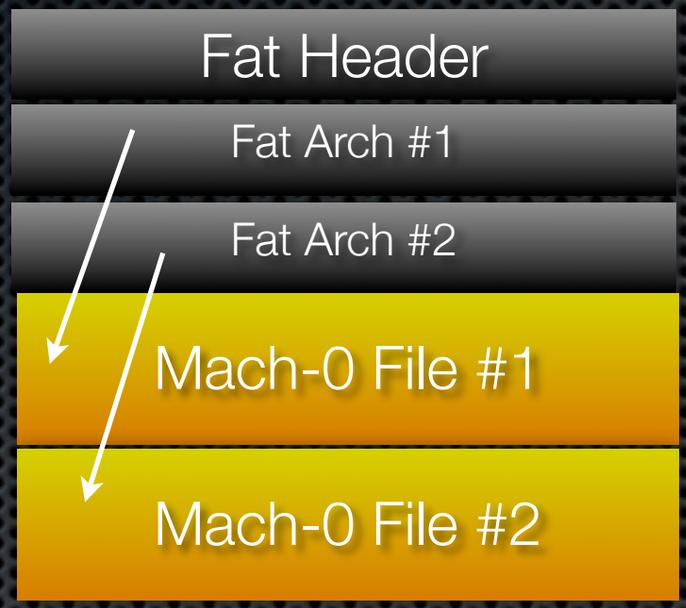
- Meta Class
- Reference Counting
- Inheritance
- WorkLoop (tasklets)

IOKit's author

Binary Formats

- a.out
- coff
- PE
- ELF
- Mach-O

Mach-O

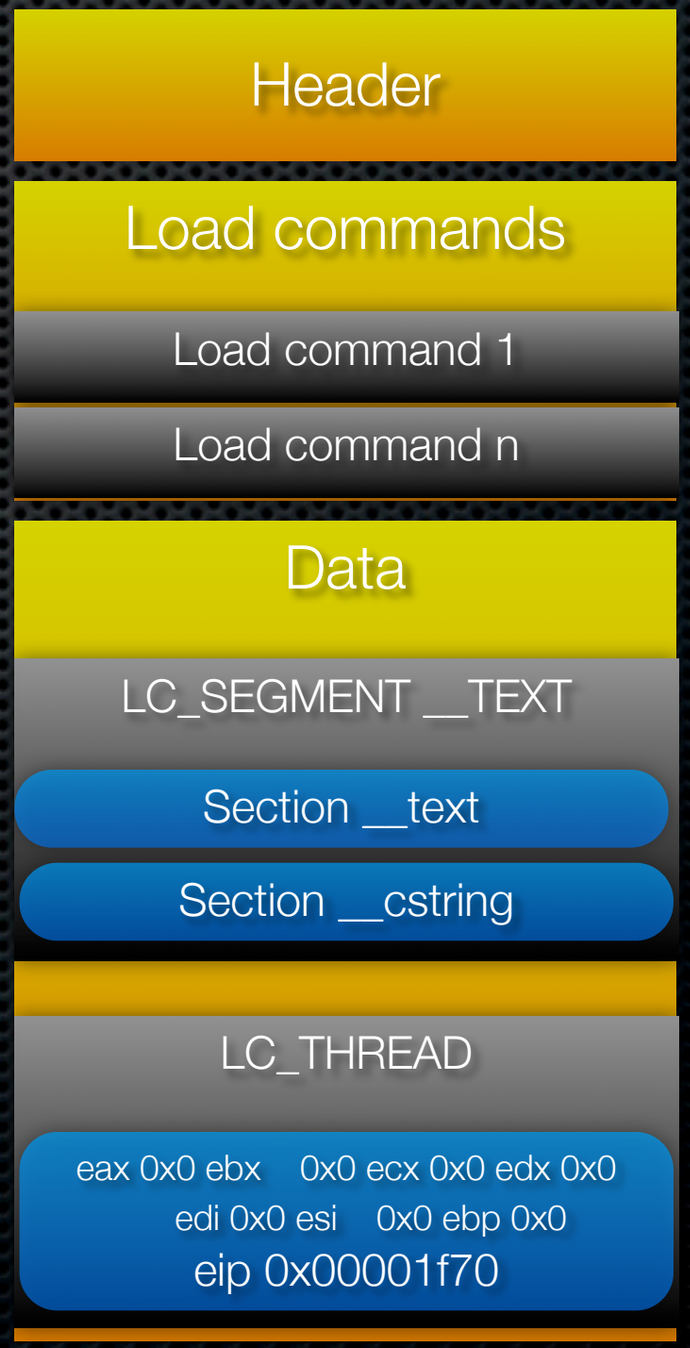


- Multi-Archi
- Kernel is a Mach-O (kernel is always 32bits)

```
$ file /mach_kernel
/mach_kernel: Mach-O universal binary with 2 architectures
/mach_kernel (for architecture i386):	Mach-O executable i386
/mach_kernel (for architecture ppc):	Mach-O executable ppc
```

Mach-O

- LC_UUID
- LC_SEGMENT / LC_SEGMENT_64
- LC_SYMTAB
- LC_DSYMTAB
- LC_THREAD
- ....



Design Considerations

- BSD designed by the academic world for the academic world
- Mach designed by the academic world for BSD
- NeXTSTEP designed for the education world
- OSX designed for???

Encryption Wrapper

What is binary encryption?

- Encrypt the binary image
- Decrypt the code at runtime
- Delay as much as possible the code decryption
- Guarantee that no single snapshot contains the whole unencrypted program

Threats

- Reverse engineers
- Crackers
- Forensic investigators

Why encrypt?

- Adding an anti piracy protection
  - *Prevent unauthorized user from executing it*
- Adding a reverse engineering protection
  - *Prevent static attacks (binary static analysis)*
- Hindering forensic analysis, just in case...

What is binary encryption?

Header

Load commands

Segment command 1

Segment command n

Segment __TEXT

Section 1 data “__text”

(__TEXT,__text) section
start:

```
00001f70        pushl   $0x00
00001f72        movl    %esp,%ebp
00001f74        andl    $0xf0,%esp
00001f77        subl    $0x10,%esp
```

Section 1 data “__cstring”

Header

Load commands

Segment command 1

Segment command n

Segment __TEXT

Section 1 data “__text”

(__TEXT,__text) section

8>^KB^@^@<89>^^P<FF><85>
$<FF><FF><FF><8B><85><F6>^O<85
><91><FC><FF><FF><8B><9D>$
<FF><FF><FF><85><DB>^O<84>{^A^
@^@<8B><85>^@<FF><FF><FF>^O

Section 1 data “__cstring”

Page Fault

Page Fault

```
0x1feb   pushl   %ebp
0x1fec   movl    %esp,%ebp
0x1fee   subl $0x08,%esp
0x1ff1   calll _hello (0x2000)
0x1ff6   leave
0x1ff7   ret
```

```
_hello:
0x2000  pushl    %ebp
....
```

0x1000
0x1feb pushl %ebp
...

0x2000
0x2000 pushl %ebp
...

Task A

Header

Load commands

Segment command 1

Segment command n

Segment __TEXT

Section 1 data "__text"

Section 1 data "__cstring"

hello.exe

Pager.GetPage()
DeviceDriver.Read()
MapPage()

0x1000
0x1feb pushl %ebp
...

0x2000
0x2000 pushl %ebp
...

What didn’t work...

- Patch the Kernel Source
- Add a custom Pager
- Fake Device

Patch the Kernel Source

- Too invasive... No one on OSX installs other kernels than the ones released by Apple
- Really hard to recompile a custom kernel, no support
- Using kernel module is really “mandatory” (cf. Apple)

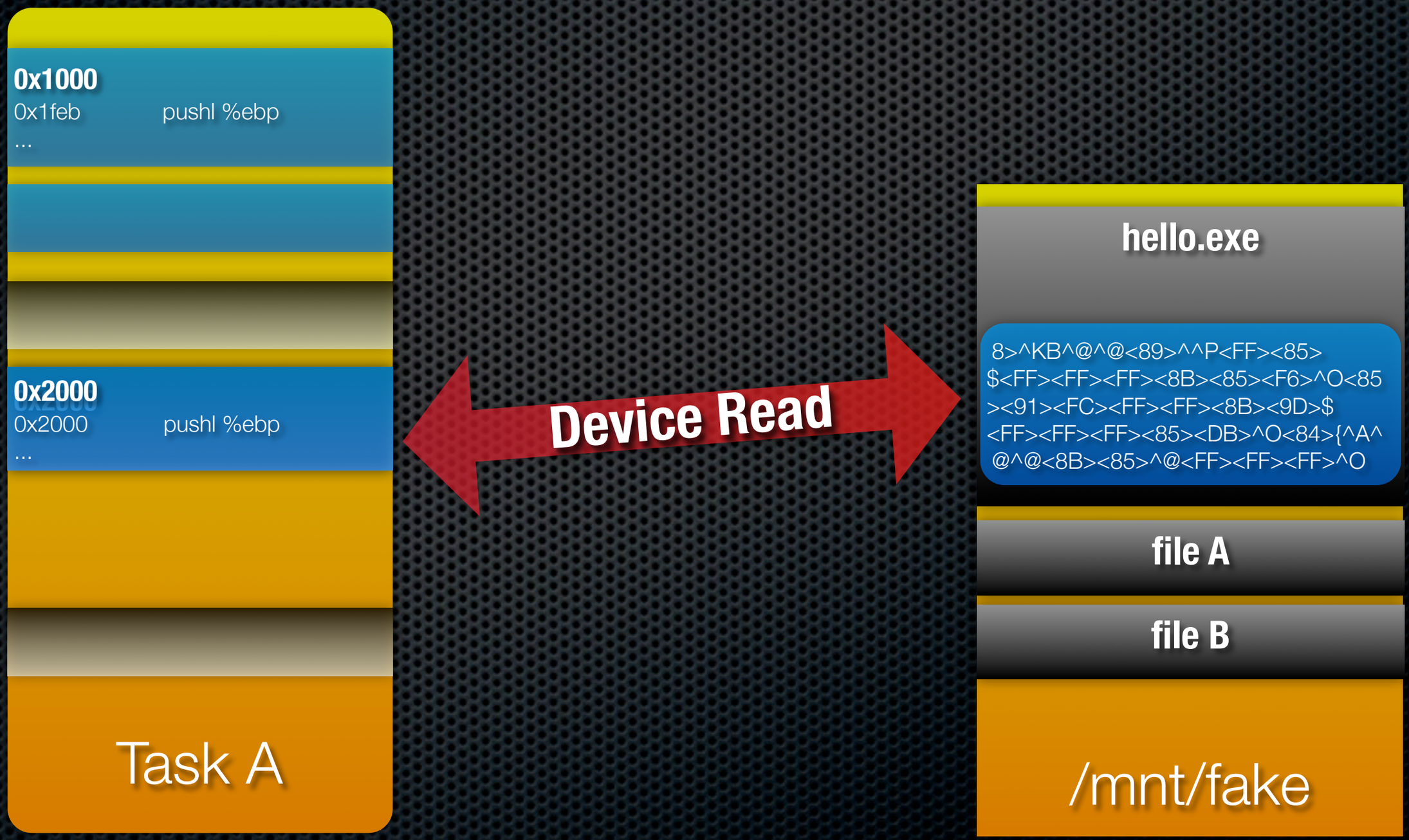
What is a pager?

- Responsible for reading/writing pages to the backing store
- Abstraction layer between the Kernel and the backing store

OSX's Pager

- Pager selection is hard coded (VNode pager, ...)
- No clean mechanism to add a custom pager
- The only way is by hooking OSX's page fault handler (dirty)

Fake Device

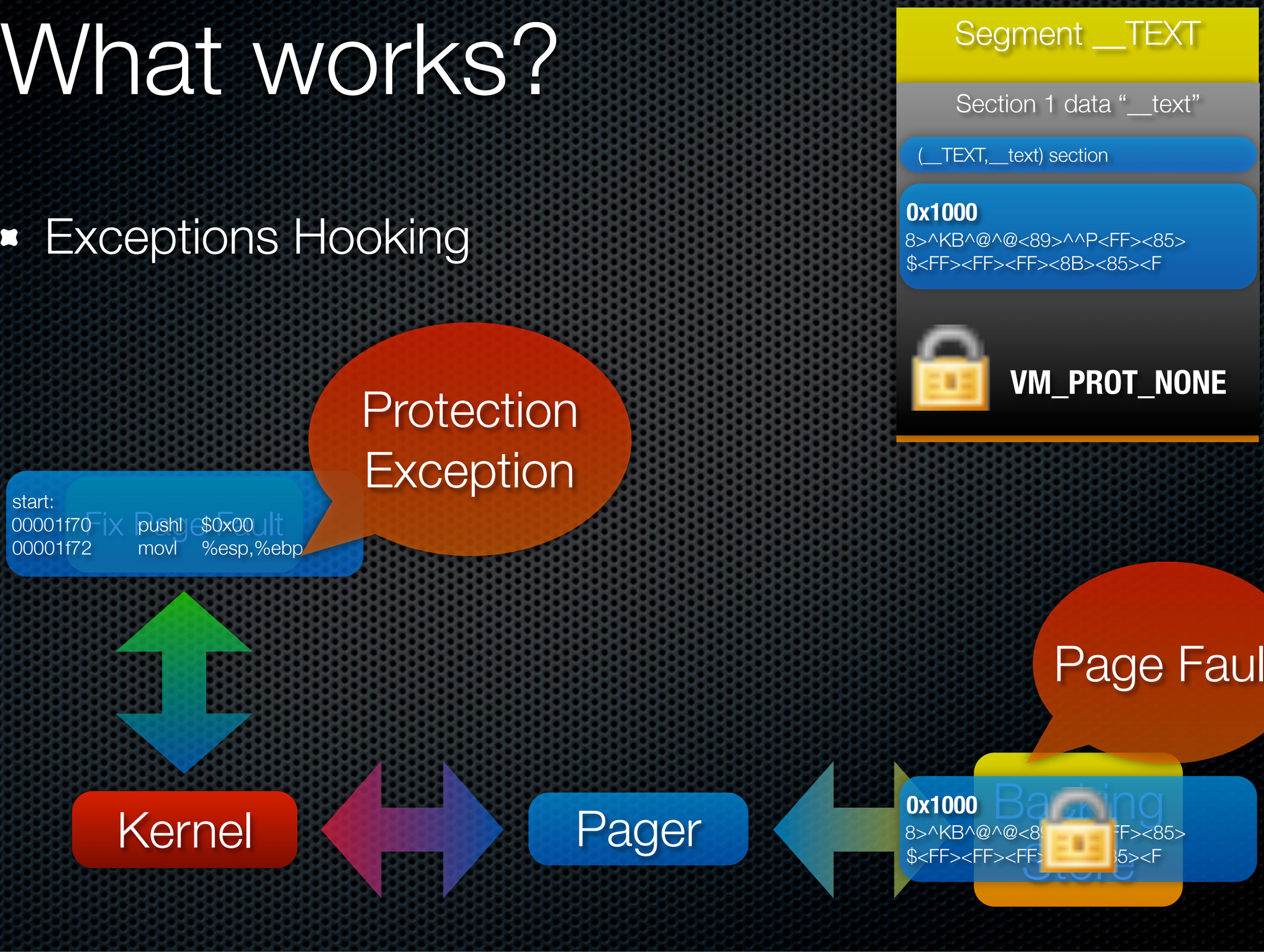

0x1000
0x1feb pushl %ebp
...
0x2000
0x2000 pushl %ebp
...
Task A
Device Read
hello.exe
8>^KB^@^@<89>^^P<FF><85>
$<FF><FF><FF><8B><85><F6>^O<85
><91><FC><FF><FF><8B><9D>$
<FF><FF><FF><85><DB>^O<84>{^A^
@^@<8B><85>^@<FF><FF><FF>^O
file A
file B
/mnt/fake

Fake Device

- Can limit accesses to kernel only
- However, can’t guarantee that the client is really our driver
- Can’t guarantee the amount of unencrypted pages

What works?

- Exceptions Hooking



Exceptions Hooking

- Clear Mach API to override task exception handler
- No overhead (no need to call the original handler)
- Thanks to Mach the User-land and the Kernel-land are both clients of Mach. The same code works in both lands

Demo

Reference

- **Inside the Mac OS X Kernel** - *Lucy <whoislucy(at)gmail.com>*
- **Meet Mach** - *James Scott*
- **IOKit Fundamentals** - *Apple*
- **Phrack p58-10 “Binary Encryption on Unix”** - *grugq <grugq@lokmail.net>*
- **The NeXT Chapter** - *kernelthread.com*
- **Infecting the Mach-o Object Format** - *Neil Archibald*
- **MACH Kernel Interface Manual** - *CMU*