CS5250: Advanced Operating Systems

AY2020-2021 Semester 2

Weng-Fai Wong
COM2-03-56
wongwf@nus.edu.sg

Objectives of CS5250

- Understand the user needs and environmental constraints that drive the design of operating systems
- Understand the overall structure and design of any modern operating system especially in how their interact
- Understand the tradeoffs in the core algorithms and extend or modify them according to user needs
- Be equipped to take on more complex systems by being exposed to some of workings of actual systems

Syllabus

- Operating system design strategies including microkernels, mobile, embedded and real-time operating systems and the component’s interfaces
- Priority and resource allocation strategies
- Scheduling algorithms
- Naming, protection and security
- User interface, windowing systems
- Distributed and shared objects
- File system implementations including network and distributed file systems
- Failure and recovery
- Virtualization and the Internet-ready OS

But how exactly will we do it?

- The challenge: we can’t write a complex OS from scratch
  - Implementing the algorithms we will discuss often requires a lot of infrastructure support
- Deep dive into x86-Linux kernel source code
- Use hands-on assignment to advance our knowledge

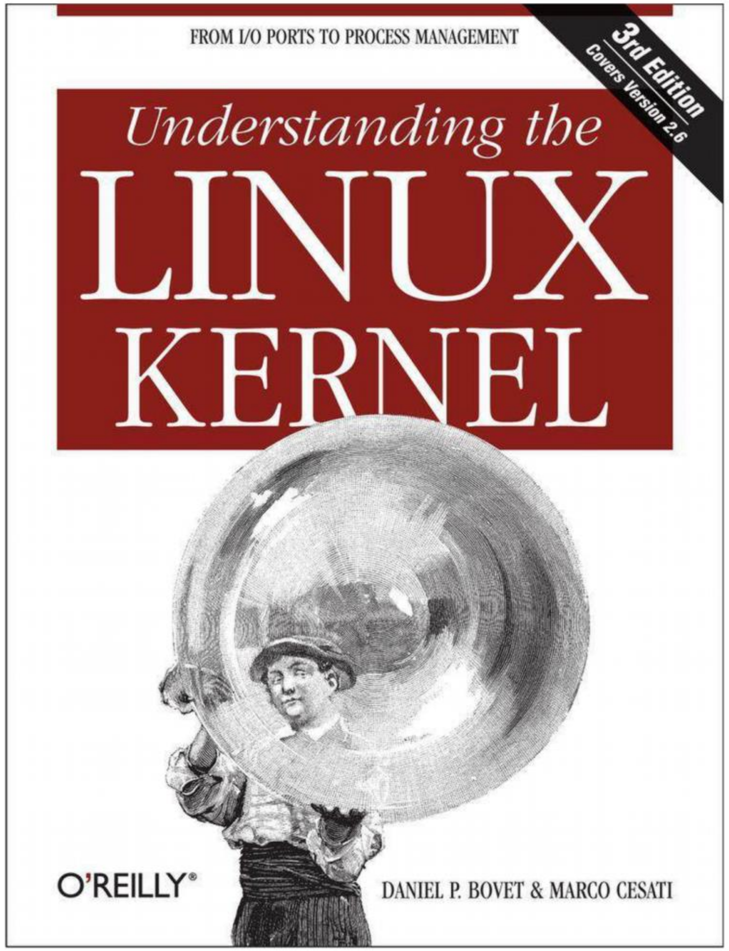
Prerequisites

- C/C++ programming
- X86 assembly programming (if possible)
- A prior undergraduate course in operating systems

Assessments

- 60% continual assessment
  - No tutorial
- 40% final assessment
  - Final assessment in the afternoon of Friday, 7 May 2021
- 4 take-home individual assignments

Reference Text


FROM I/O PORTS TO PROCESS MANAGEMENT
3rd Edition
Covers Version 2.6
Understanding the
LINUX
KERNEL
O'REILLY®
DANIEL P. BOVET & MARCO CESATI

Introduction

What is an Operating Systems?

- Low level programs that sits between the hardware and user applications

Functions of an operating systems

- Manage a computer system's resources
- Provide user interface
- Provide services for applications

Key system resources

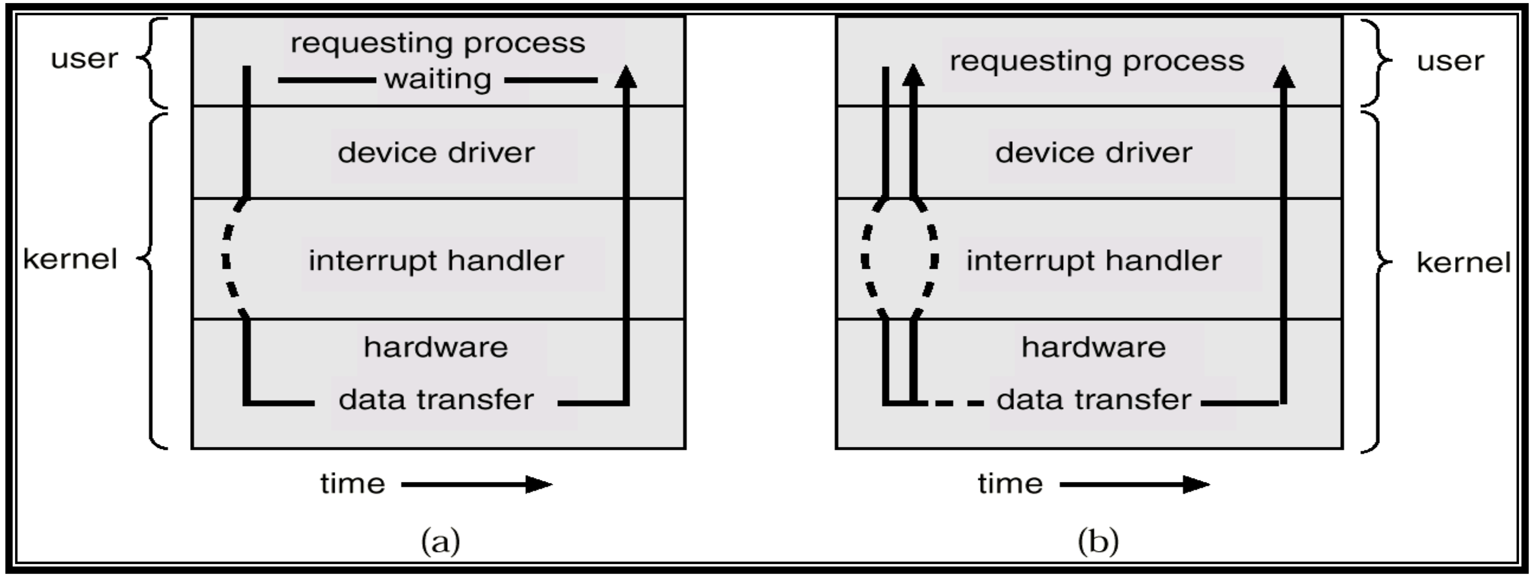
- Processors (cores)
- Memory
- Input/Output devices
  - Keyboard, mouse, monitor, and other human interface devices
  - Graphics card
  - Disk storage
  - Network
  - Printers, USB devices etc.
- Data and storage

Other key services

- Hardware abstraction
- Protection
- Sharing and data exchange
- Caching
- Interrupt handling

Hardware abstraction

- Provide regulated access to hardware
- Manufacturers provide “drivers” – software interfaces
- Provide book-keeping services



Protection

- Protect users (especially in a multiuser environment) against other users who may be malicious or plain ignorant

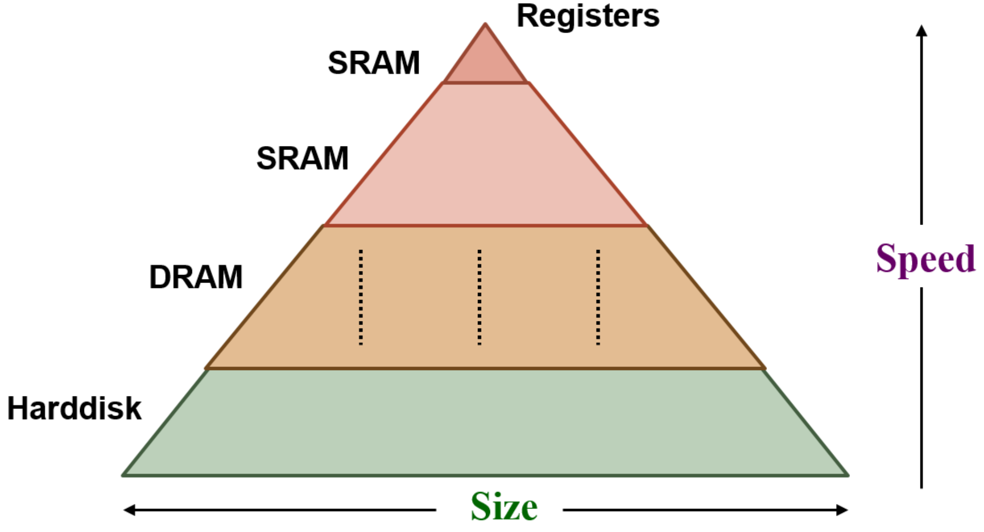
- At least two modes of protection enabled with hardware support:
  - User mode
  - Supervisor mode (root access)

Sharing

- Isolation vs sharing
- Need to establish that the sharing is legitimate
- Inter-process communication
- Via networking

Caching

- Caching is ubiquitious
  - To bridge differences in speeds and capacities
- Hardware caching
  - TLB, processor cache, disk caches
- Software caching
  - Page cache



Interrupt handling

- Interrupts – asynchronous to program execution
  - caused by external events
  - may be handled between instructions, so can let the instructions currently active in the pipeline *complete* before passing control to the OS interrupt handler
  - simply suspend and resume user program

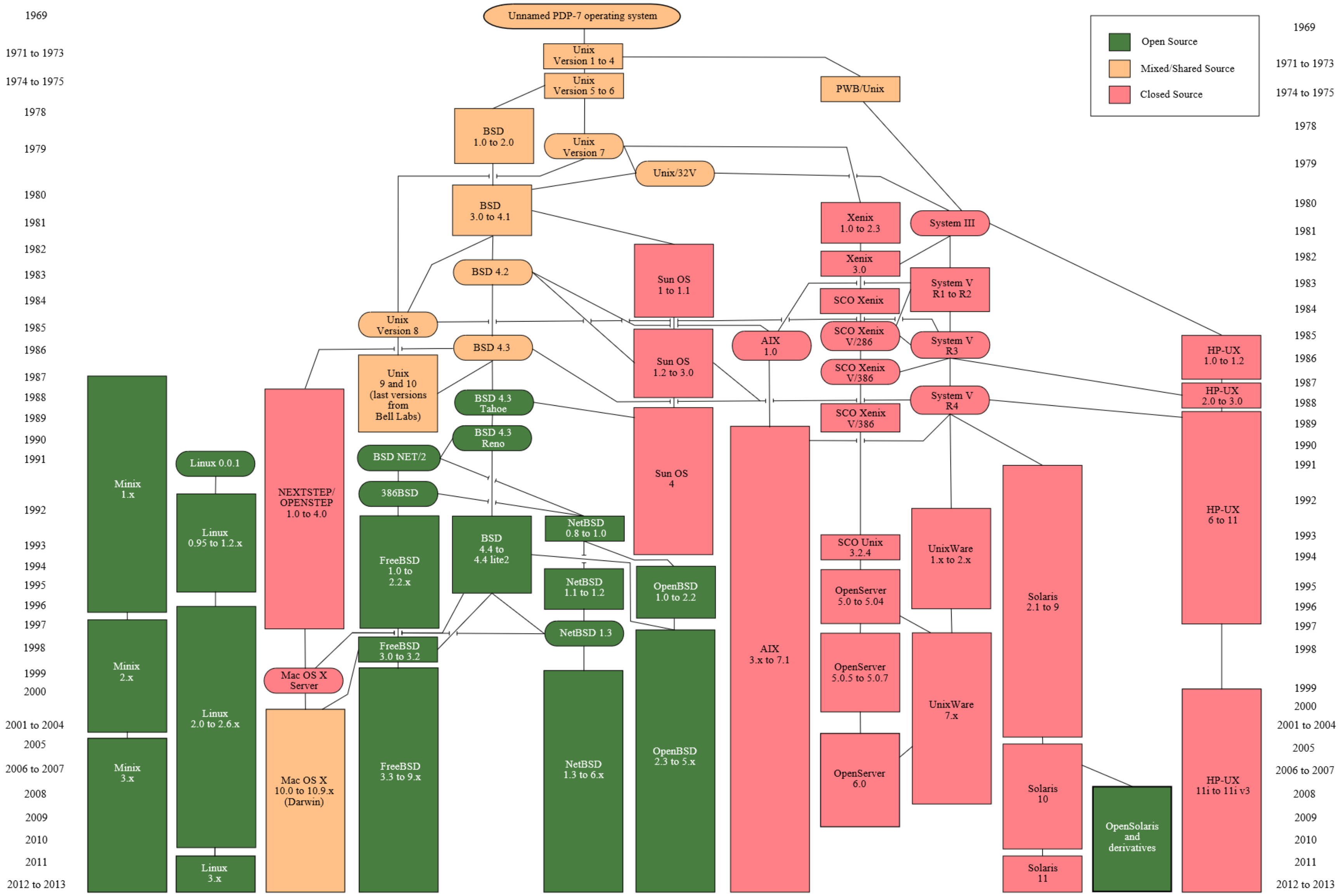
- Traps (Exception) – synchronous to program execution
  - caused by internal events
  - condition must be remedied by the trap handler for that instruction, so much stop the offending instruction *midstream* in the pipeline and pass control to the OS trap handler
  - the offending instruction may be retried (or simulated by the OS) and the program may continue or it may be aborted

Interrupt handling – examples

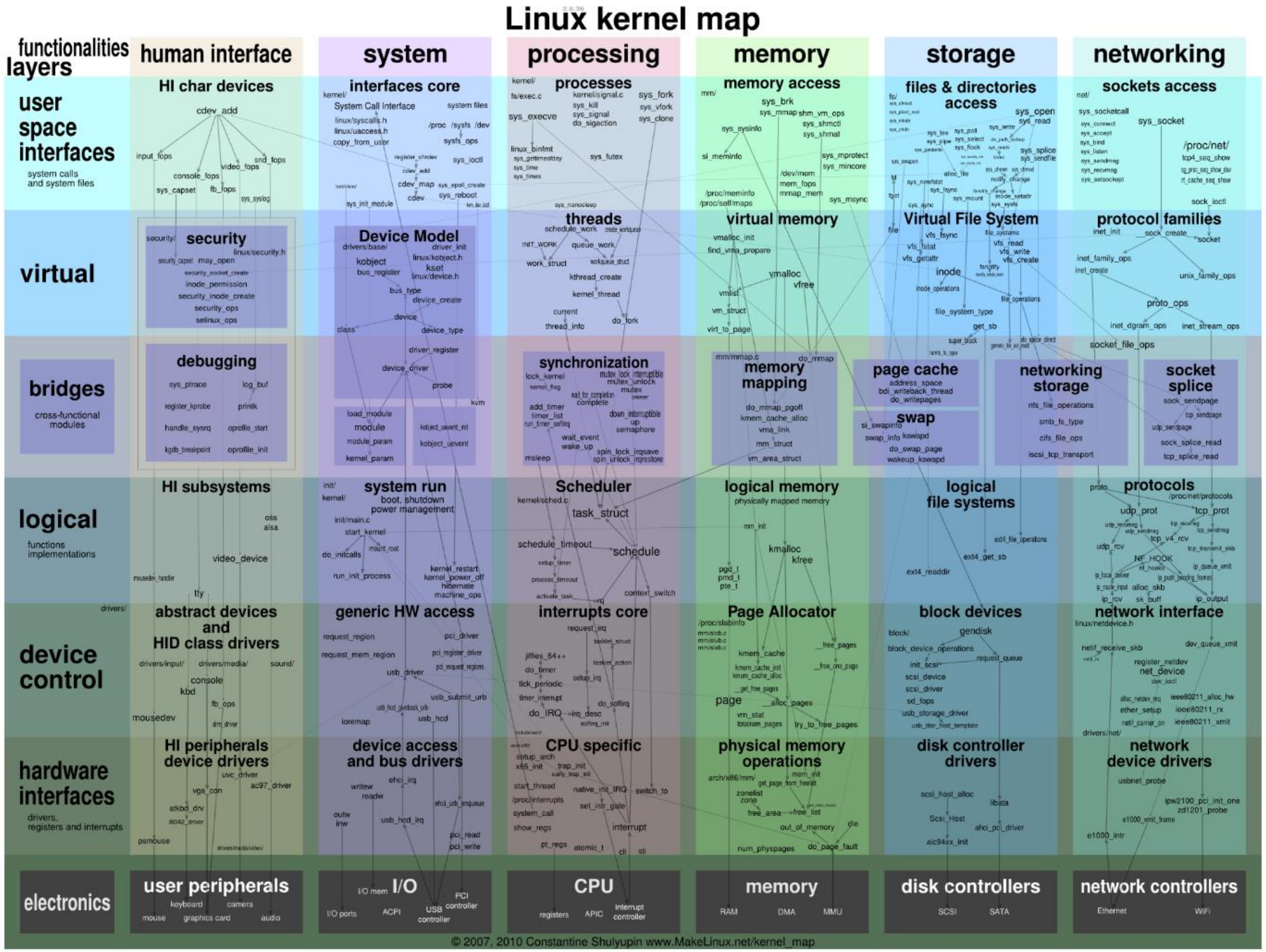
- Arithmetic overflow
- Undefined instruction
- TLB or page fault
- Segmentation fault
- I/O service request
- Hardware malfunction
- Timer

Key Operating Systems

Unix


1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2013
Open Source
Mixed/Shared Source
Closed Source
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
Sun OS 1 to 1.1
SCO Xenix
System V R1 to R2
Unix Version 8
AIX 1.0
SCO Xenix V/286
System V R3
BSD 4.3
Sun OS 1.2 to 3.0
SCO Xenix V/386
HP-UX 1.0 to 1.2
Unix 9 and 10 (last versions from Bell Labs)
BSD 4.3 Tahoe
System V R4
HP-UX 2.0 to 3.0
SCO Xenix V/386
BSD 4.3 Reno
Minix 1.x
Linux 0.0.1
NEXTSTEP/ OPENSTEP 1.0 to 4.0
BSD NET/2
Sun OS 4
386BSD
HP-UX 6 to 11
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4 to 4.4 lite2
NetBSD 0.8 to 1.0
SCO Unix 3.2.4
UnixWare 1.x to 2.x
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Solaris 2.1 to 9
NetBSD 1.3
FreeBSD 3.0 to 3.2
Minix 2.x
Mac OS X Server
AIX 3.x to 7.1
OpenServer 5.0.5 to 5.0.7
Linux 2.0 to 2.6.x
UnixWare 7.x
OpenBSD 2.3 to 5.x
NetBSD 1.3 to 6.x
FreeBSD 3.3 to 9.x
Mac OS X 10.0 to 10.9.x (Darwin)
Minix 3.x
OpenServer 6.0
Solaris 10
HP-UX 11i to 11i v3
OpenSolaris and derivatives
Linux 3.x
Solaris 11
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2013

Linux


Linux kernel map
functionalities
layers
human interface
system
processing
memory
storage
networking
user space interfaces
system calls and system files
HI char devices
interfaces core
processes
memory access
files & directories access
sockets access
virtual
security
Device Model
threads
virtual memory
Virtual File System
protocol families
bridges
cross-functional modules
debugging
synchronization
memory mapping
page cache
swap
networking storage
socket splice
logical
functions implementations
HI subsystems
system run
boot, shutdown
power management
Scheduler
logical memory
logical file systems
protocols
device control
abstract devices and HID class drivers
generic HW access
interrupts core
Page Allocator
block devices
network interface
hardware interfaces
drivers, registers and interrupts
HI peripherals device drivers
device access and bus drivers
CPU specific
physical memory operations
disk controller drivers
network device drivers
electronics
user peripherals
keyboard
camera
mouse
graphics card
audio
I/O
I/O mem
I/O ports
ACPI
USB controller
PCI controller
CPU
registers
APIC
interrupt controller
memory
RAM
DMA
MMU
disk controllers
SCSI
SATA
network controllers
Ethernet
WiFi
cdev_add
input_fops
console_fops
video_fops
snd_fops
fb_fops
sys_capset
sys_syslog
System Call Interface
system files
linux/syscalls.h
linux/uaccess.h
copy_from_user
/proc /sysfs /dev
sysfs_ops
sys_ioctl
cdev_map
cdev
sys_reboot
sys_init_module
sys_execve
sys_fork
sys_vfork
sys_clone
sys_kill
sys_signal
do_sigaction
linux_binfmt
sys_futex
sys_brk
sys_mmap
shm_vm_ops
sys_shmctl
sys_shmat
sys_sysinfo
si_meminfo
/dev/mem
mem_fops
mmap_mem
sys_mprotect
sys_mincore
sys_msync
/proc/meminfo
/proc/self/maps
sys_open
sys_read
sys_splice
sys_sendfile
sys_socket
sys_socketcall
/proc/net/
tcp4_seq_show
sock_ioctl
security_inode_create
security_ops
selinux_ops
inode_permission
kobject
kset
bus_type
device_create
device
device_type
driver_register
device_driver
probe
class
schedule_work
queue_work
work_struct
kthread_create
kernel_thread
current
thread_info
do_fork
vmalloc_init
find_vma_prepare
vmalloc
vfree
vmlist
vm_struct
virt_to_page
vfs_fsync
vfs_lstat
vfs_getattr
vfs_read
vfs_write
vfs_create
inode
file_operations
file_system_type
get_sb
super_block
inet_init
sock_create
socket
inet_family_ops
inet_create
unix_family_ops
proto_ops
inet_dgram_ops
inet_stream_ops
socket_file_ops
sys_ptrace
log_buf
register_kprobe
printk
handle_sysrq
oprofile_start
kgdb_breakpoint
oprofile_init
load_module
module
module_param
kernel_param
kobject_uevent
lock_kernel
mutex
add_timer
timer_list
complete
down_interruptible
up
semaphore
wait_event
wake_up
spin_lock_irqsave
spin_unlock_irqrestore
msleep
do_mmap_pgoff
kmem_cache_alloc
vma_link
mm_struct
vm_area_struct
address_space
bdi_writeback_thread
do_writepages
swap_info
kswapd
do_swap_page
wakeup_kswapd
nfs_file_operations
smb_fs_type
cifs_file_ops
iscsi_tcp_transport
sock_sendpage
udp_sendpage
sock_splice_read
tcp_splice_read
video_device
tty
init/main.c
start_kernel
do_initcalls
run_init_process
kernel_restart
kernel_power_off
hibernate
machine_ops
task_struct
schedule_timeout
schedule
context_switch
mm_init
kmalloc
kfree
pgd_t
pmd_t
pte_t
ext4_readdir
ext4_get_sb
udp_prot
tcp_prot
tcp_v4_rcv
udp_rcv
NF_HOOK
alloc_skb
ip_rcv
sk_buff
ip_output
console
kbd
fb_ops
mousedev
request_region
request_mem_region
pci_driver
usb_driver
usb_submit_urb
usb_hcd
ioremap
request_irq
jiffies_64++
do_timer
tick_periodic
timer_interrupt
do_IRQ
irq_desc
do_softirq
kmem_cache
page
alloc_pages
vm_stat
try_to_free_pages
gendisk
block_device_operations
request_queue
init_scsi
scsi_device
scsi_driver
sd_fops
usb_storage_driver
netif_receive_skb
dev_queue_xmit
register_netdev
net_device
ieee80211_alloc_hw
ether_setup
ieee80211_rx
ieee80211_xmit
vga_con
ac97_driver
atkbd_drv
psmouse
ehci_irq
writew
readw
outw
inw
usb_hcd_irq
pci_read
pci_write
setup_arch
x86_init
trap_init
start_thread
native_init_IRQ
switch_to
/proc/interrupts
set_intr_gate
system_call
show_regs
interrupt
pt_regs
atomic_t
cli
sti
zonelist
zone
free_area
free_list
out_of_memory
die
num_physpages
do_page_fault
scsi_host_alloc
Scsi_Host
libata
ahci_pci_driver
aic94xx_init
usbnet_probe
ipw2100_pci_init_one
zd1201_probe
e1000_xmit_frame
e1000_intr
© 2007, 2010 Constantine Shulyupin www.MakeLinux.net/kernel_map

Microsoft Windows

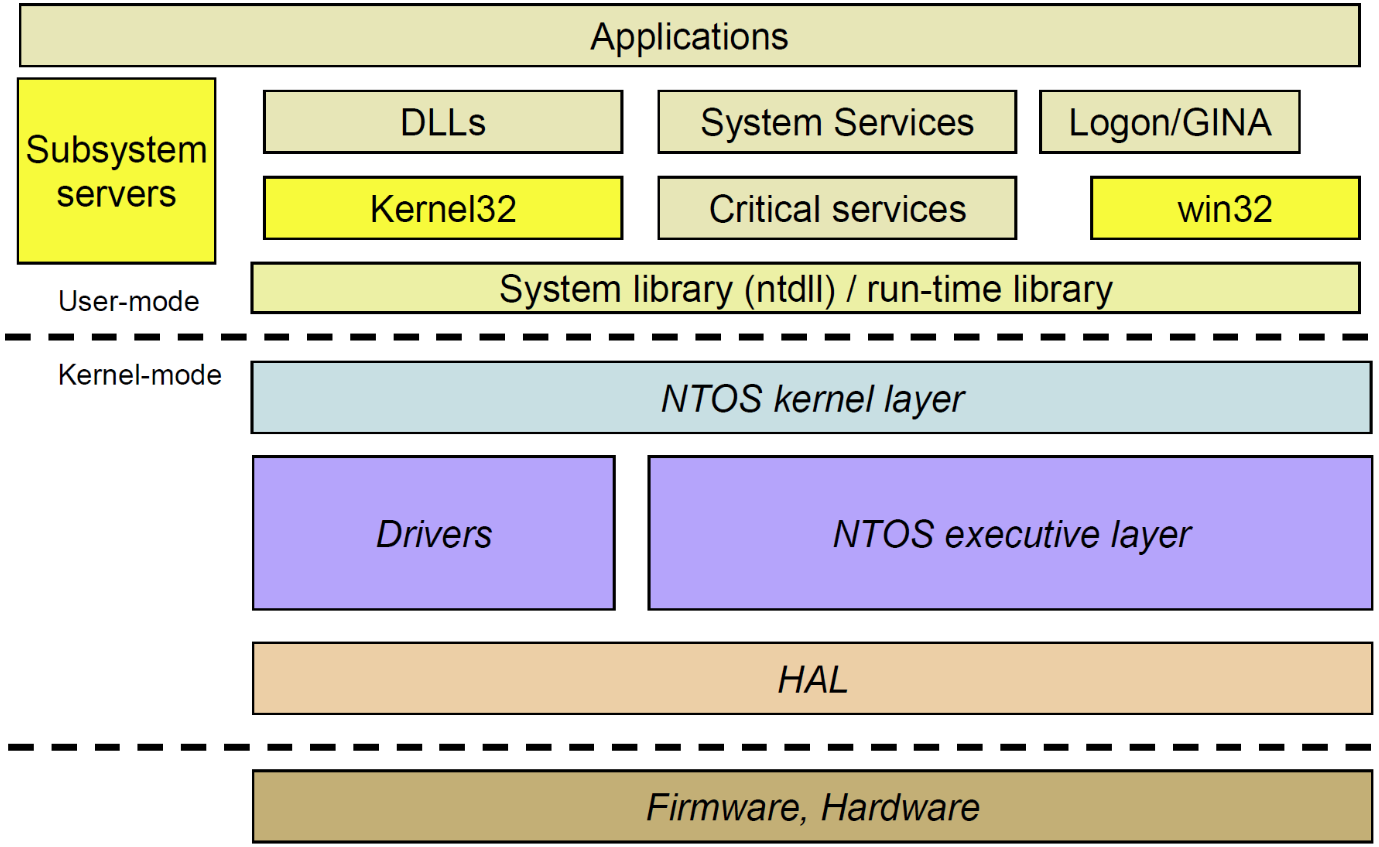
- 16-bit:
  - MS-DOS (various versions, v1.0 in 1985)
  - Windows 1.X – 3.X, Windows 9X, Windows ME (2000)

- 32-bit:
  - Windows NT (32-bit, v3.1 in 1994)
  - Windows 2000, XP, 2003, Vista, 7, 8, 10 (2015)

- 64-bit:
  - Windows XP (2005), Vista, 7, 8, 10 (2015)

- Mostly on PC (Intel Processors) platforms
- Proprietary
  - some sources available under conditions
- Complex architecture, internals info not widely available

Microsoft Windows



Applications

Subsystem servers

DLLs

System Services

Logon/GINA

Kernel32

Critical services

win32

System library (ntdll) / run-time library

User-mode

Kernel-mode

NTOS kernel layer

Drivers

NTOS executive layer

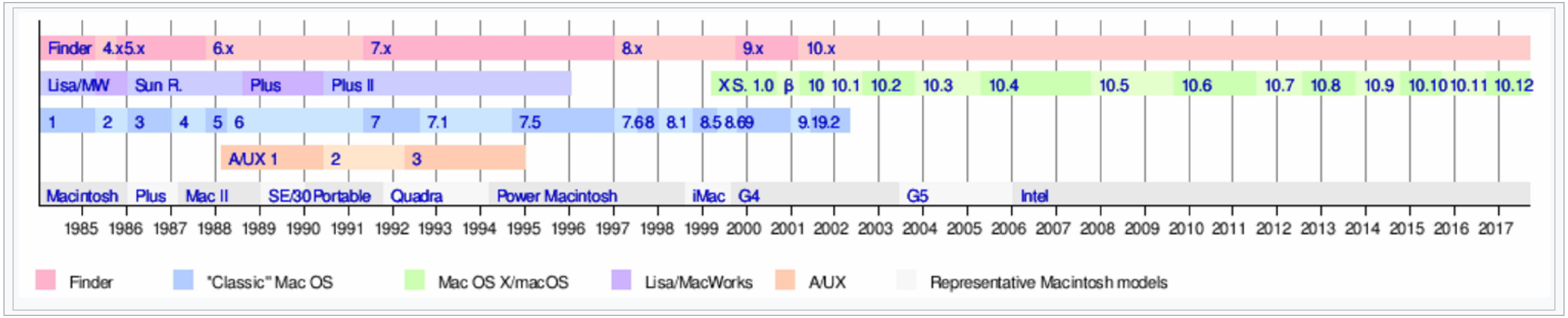
HAL

Firmware, Hardware

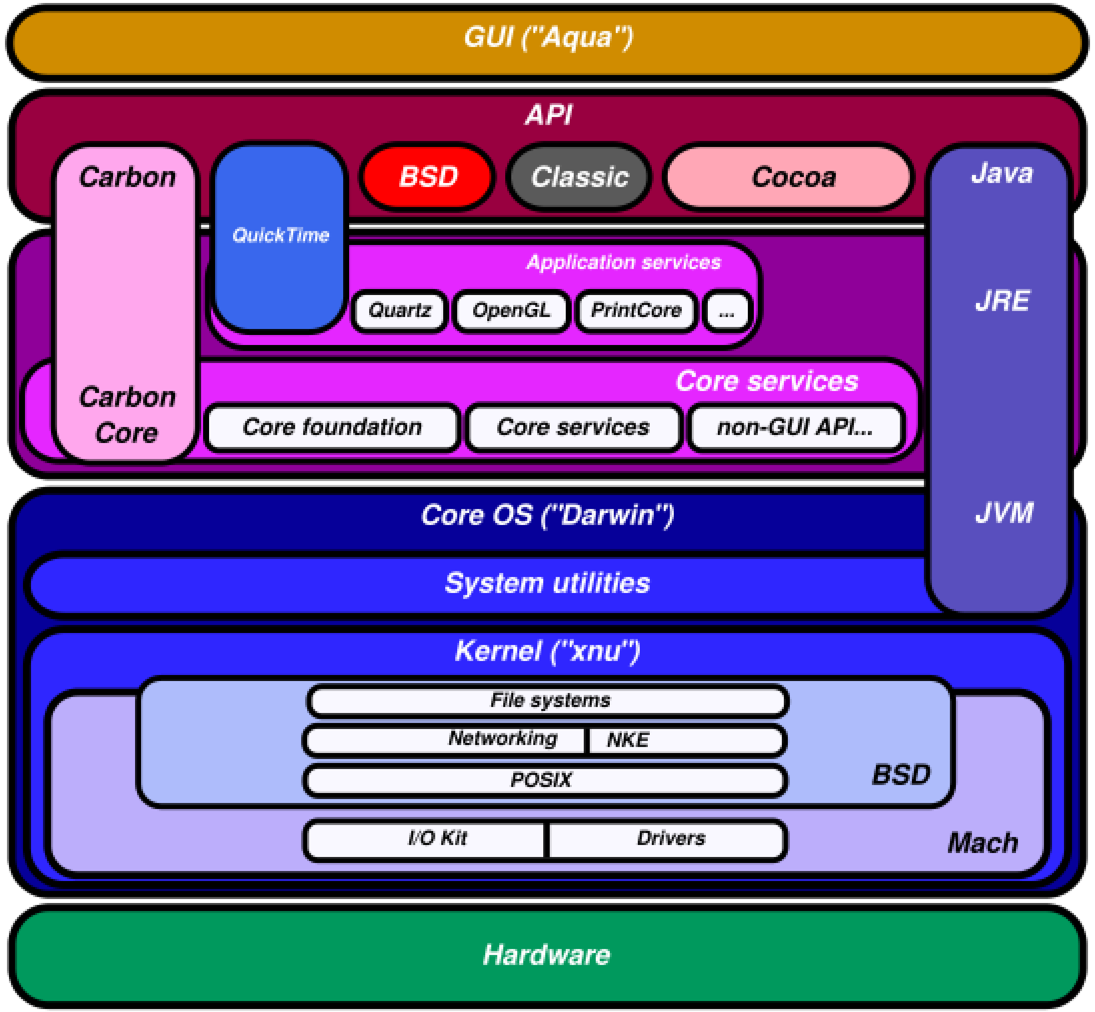
© Microsoft Corporation

MacOS

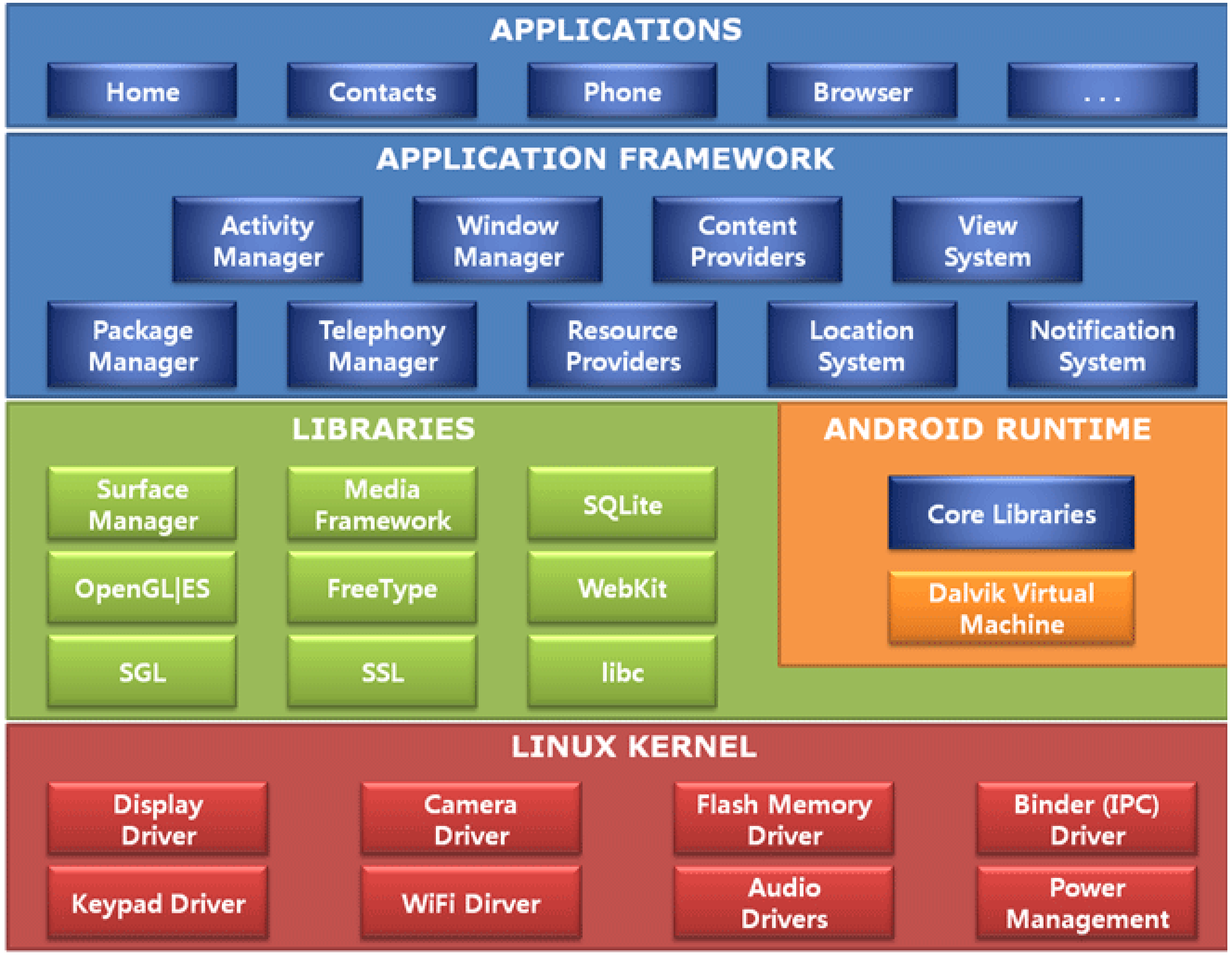
Timeline of Macintosh operating systems



MacOS


GUI ("Aqua")
API
Carbon
QuickTime
BSD
Classic
Cocoa
Java
Application services
Quartz
OpenGL
PrintCore
...
JRE
Carbon Core
Core services
Core foundation
Core services
non-GUI API...
Core OS ("Darwin")
JVM
System utilities
Kernel ("xnu")
File systems
Networking
NKE
POSIX
BSD
I/O Kit
Drivers
Mach
Hardware

Android OS



APPLICATIONS

Home | Contacts | Phone | Browser | . . .

APPLICATION FRAMEWORK

Activity Manager | Window Manager | Content Providers | View System

Package Manager | Telephony Manager | Resource Providers | Location System | Notification System

LIBRARIES

Surface Manager | Media Framework | SQLite

OpenGL|ES | FreeType | WebKit

SGL | SSL | libc

ANDROID RUNTIME

Core Libraries

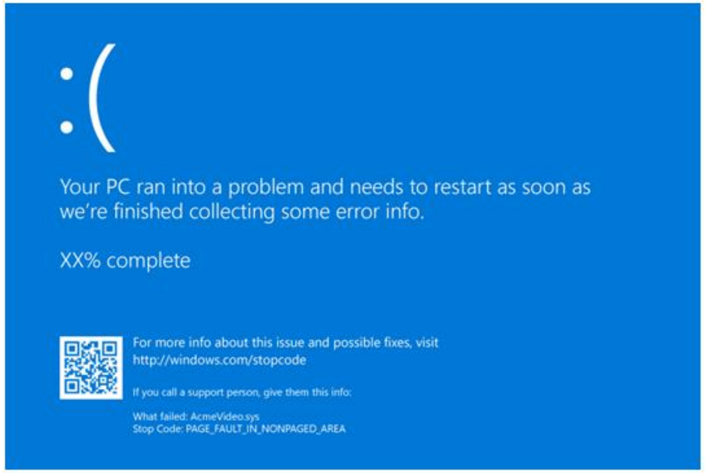
Dalvik Virtual Machine

LINUX KERNEL

Display Driver | Camera Driver | Flash Memory Driver | Binder (IPC) Driver

Keypad Driver | WiFi Dirver | Audio Drivers | Power Management

The Show Stopper



:(

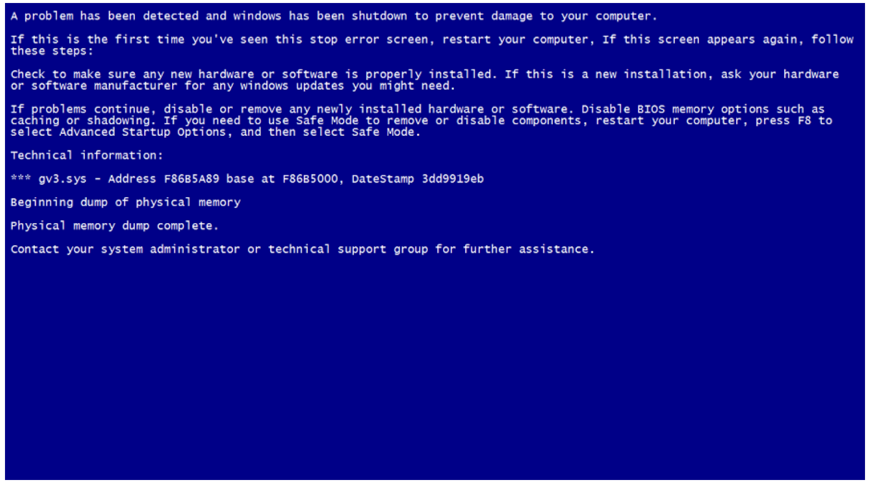
Your PC ran into a problem and needs to restart as soon as we're finished collecting some error info.

XX% complete

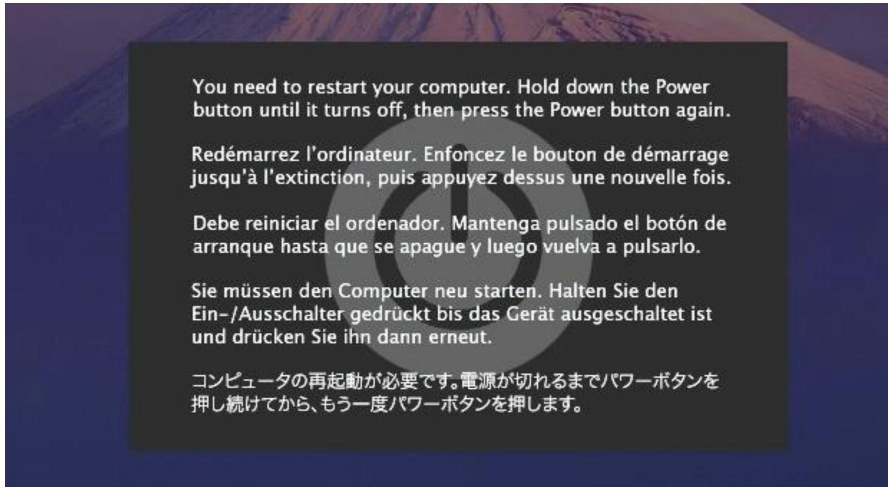
For more info about this issue and possible fixes, visit
http://windows.com/stopcode

If you call a support person, give them this info:

What failed: AcmeVideo.sys
Stop Code: PAGE_FAULT_IN_NONPAGED_AREA



```
A problem has been detected and windows has been shutdown to prevent damage to your computer.

If this is the first time you've seen this stop error screen, restart your computer, If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware
or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as
caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to
select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd9919eb

Beginning dump of physical memory

Physical memory dump complete.

Contact your system administrator or technical support group for further assistance.
```



You need to restart your computer. Hold down the Power button until it turns off, then press the Power button again.

Redémarrez l'ordinateur. Enfoncez le bouton de démarrage jusqu'à l'extinction, puis appuyez dessus une nouvelle fois.

Debe reiniciar el ordenador. Mantenga pulsado el botón de arranque hasta que se apague y luego vuelva a pulsarlo.

Sie müssen den Computer neu starten. Halten Sie den Ein-/Ausschalter gedrückt bis das Gerät ausgeschaltet ist und drücken Sie ihn dann erneut.

コンピュータの再起動が必要です。電源が切れるまでパワーボタンを押し続けてから、もう一度パワーボタンを押します。

Characteristics of OSes

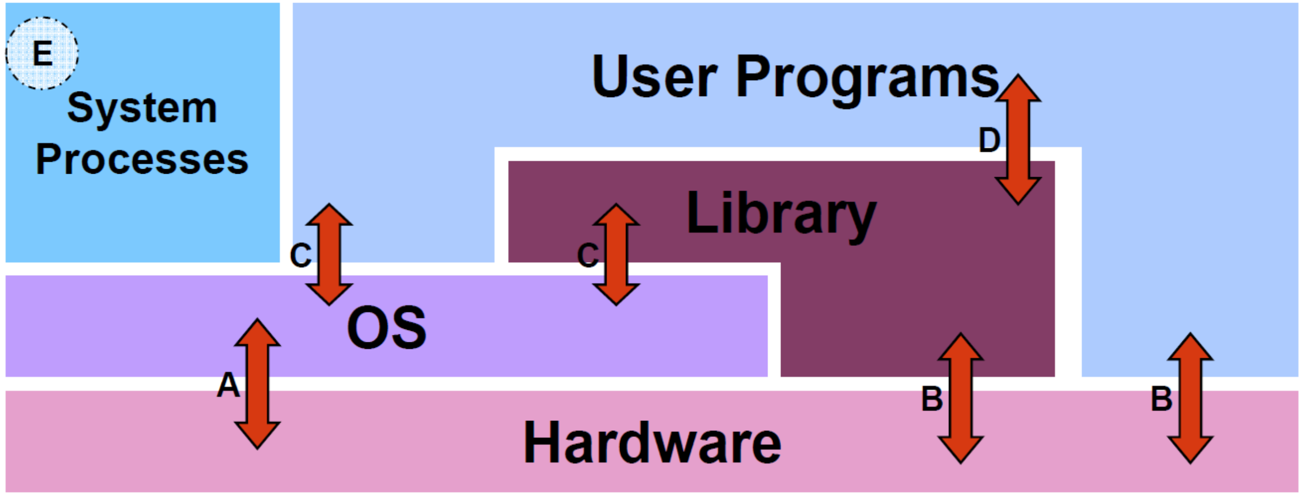
Types of OS

- Batch
  - Data centers
- Interactive
  - PCs
  - Mobile devices
- Real-time
  - Embedded systems
  - Robots
- Hybrid

Structures of OS

- Monolithic
- Microkernel
- Network OS
- Distributed OS
- Exokernels

A Generic OS

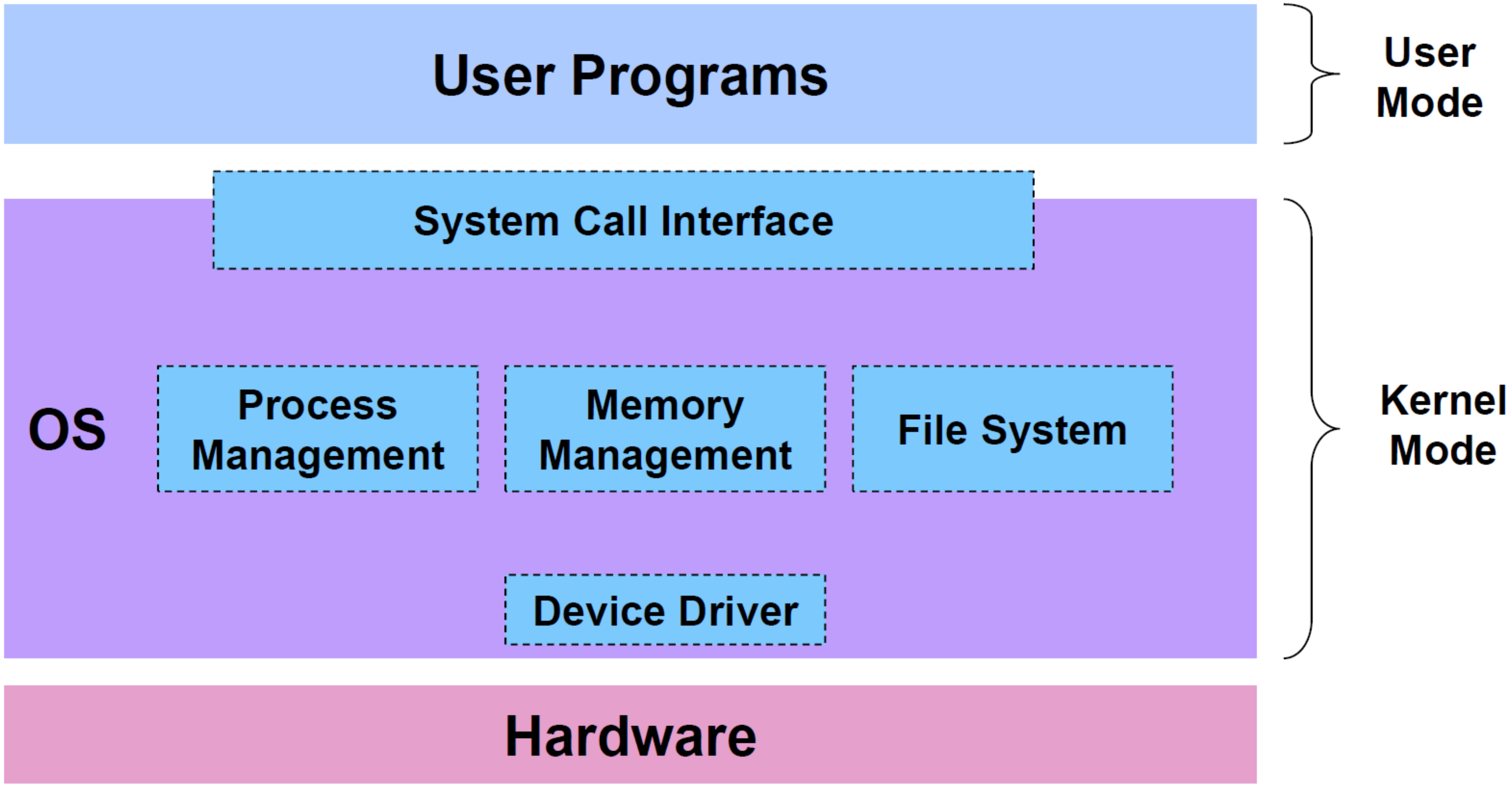


- **A**: OS executing machine instructions
- **B**: normal machine instructions executed (program/library code)
- **C**: calling OS using ***system call interface***
- **D**: user program calls library code
- **E**: system processes
  - Provide high level services, usually part of OS

Monolithic Design

- Usually a single large process
- Runs in a single address space – the kernel space
- Often a single binary file loaded at boot time
    - With other processes and files such as drivers assisting

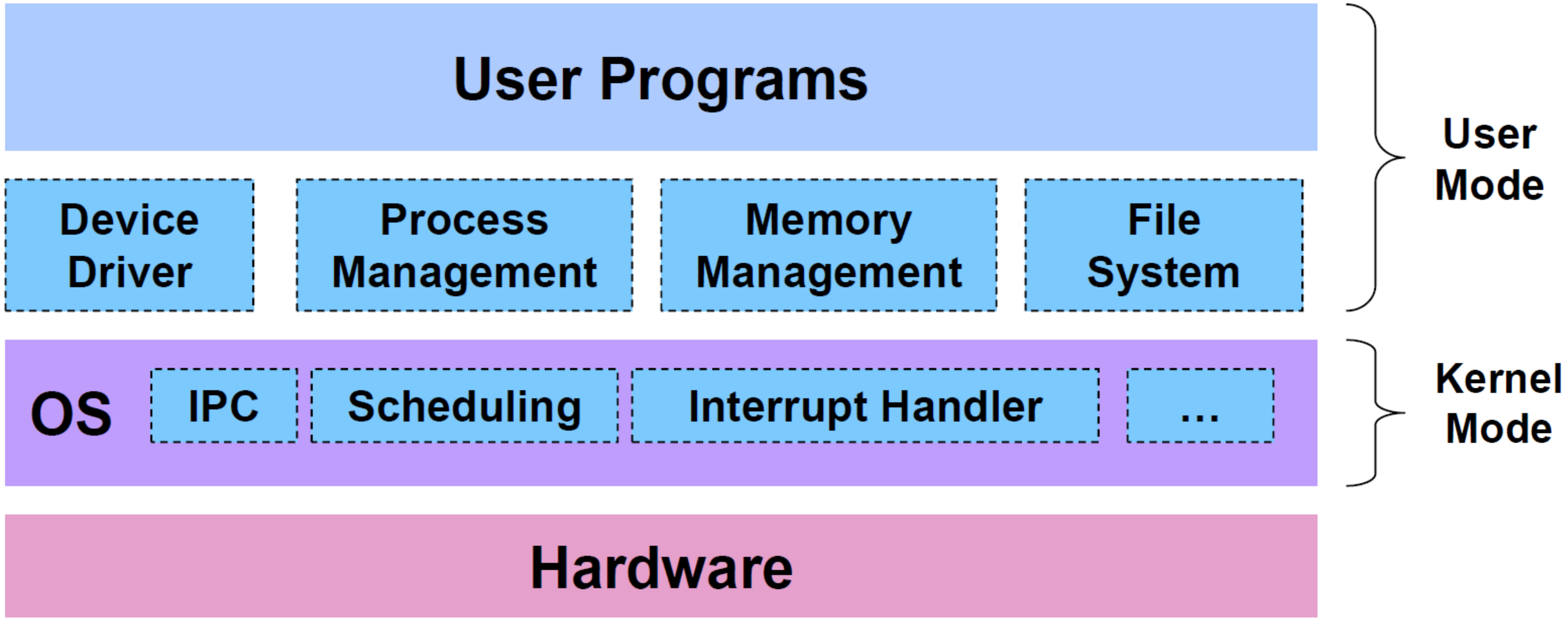
Monolithic Design



Microkernel

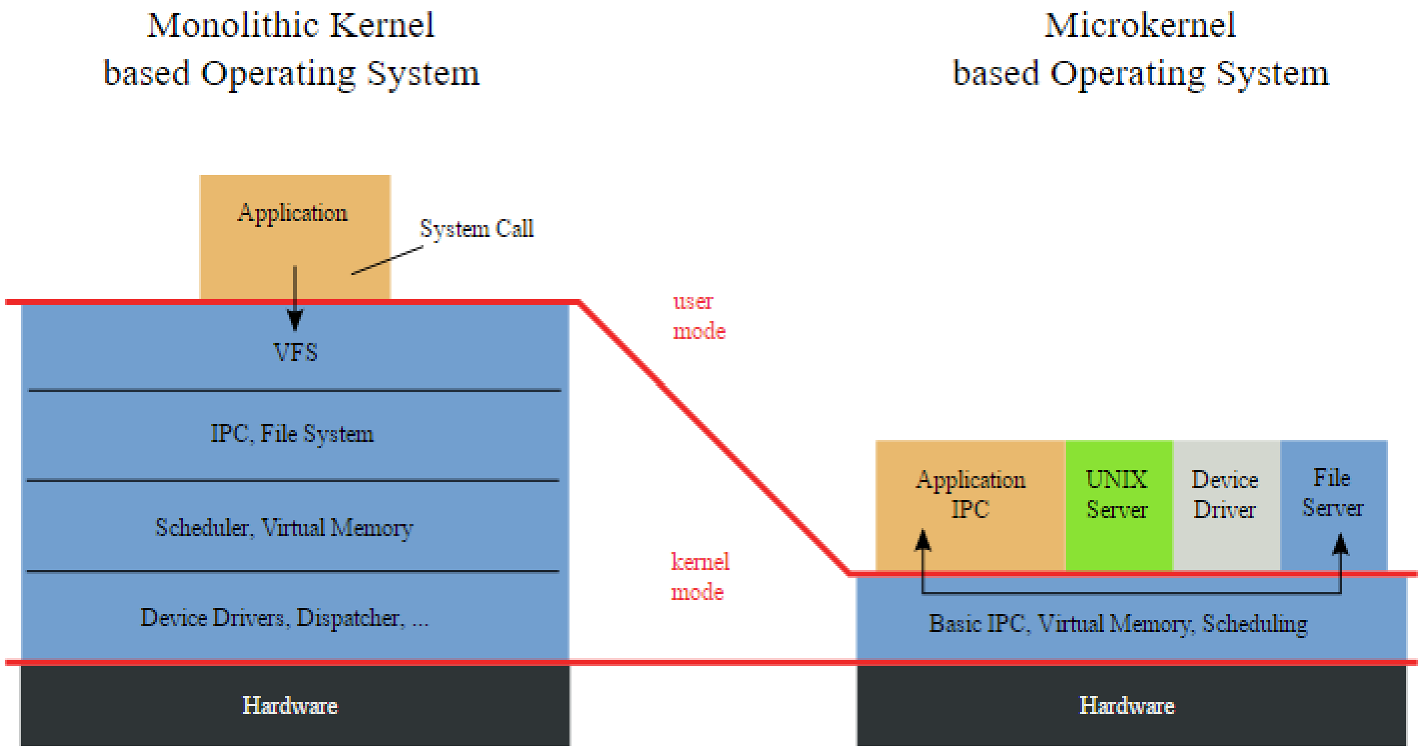
- Small kernel supporting minimal essential services
  - IPC, virtual memory, processing scheduling, interrupt handling, protection
- All other services handled by separate user level server processes
- Applications request for services from the server processes

Microkernel Design



Example: MINIX 3 – kernel only 12,000 lines

Microkernel – using IPC



Microkernel pros and cons

- Security advantage
  - Principle of Least Privilege
    - “only enough privilege to do what is required – and no more”
    - A problem bugging Windows

- Performance disadvantage
  - IPC requires 2 OS kernel crossing and process scheduling

Exokernels

- Squeeze kernel even further by removing more of its management duties
- Give applications direct access to hardware via libraries
- Experimental