Week 4.a
CS6640
01/27 2026
https://naizhengtan.github.io/26spring/

□ 0. recap: gdb w/ an example
□ 1. OS organization
□ 2. egos desgin
□ 3. egos-2k+ booting process

----

* gdb basics

Scenario 1: "what's wrong?"
- run to failure
- use gdb to see the final status

Scenario 2: "I suspect this is wrong"
- set a breakpoint
- continue the egos
- run until the breakpoint
- single step running & monitoring

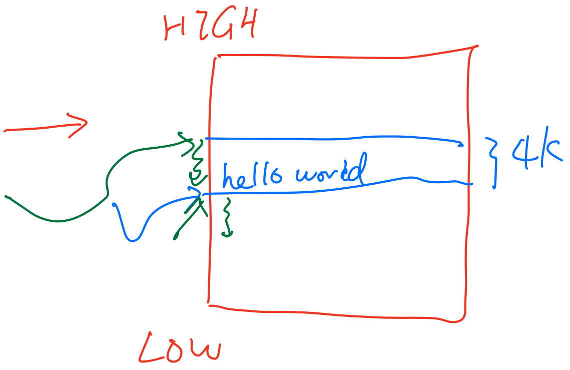
s	si
n	ni

a) Question: At what memory address does the CPU execute the first kernel instruction?

b) Question: How does printf() work?

c) Question: How does context switching work in Lab 2?

sp = malloc(4096);
+4096



- OS architecture

Modern Operating Systems: Principles and Implementation

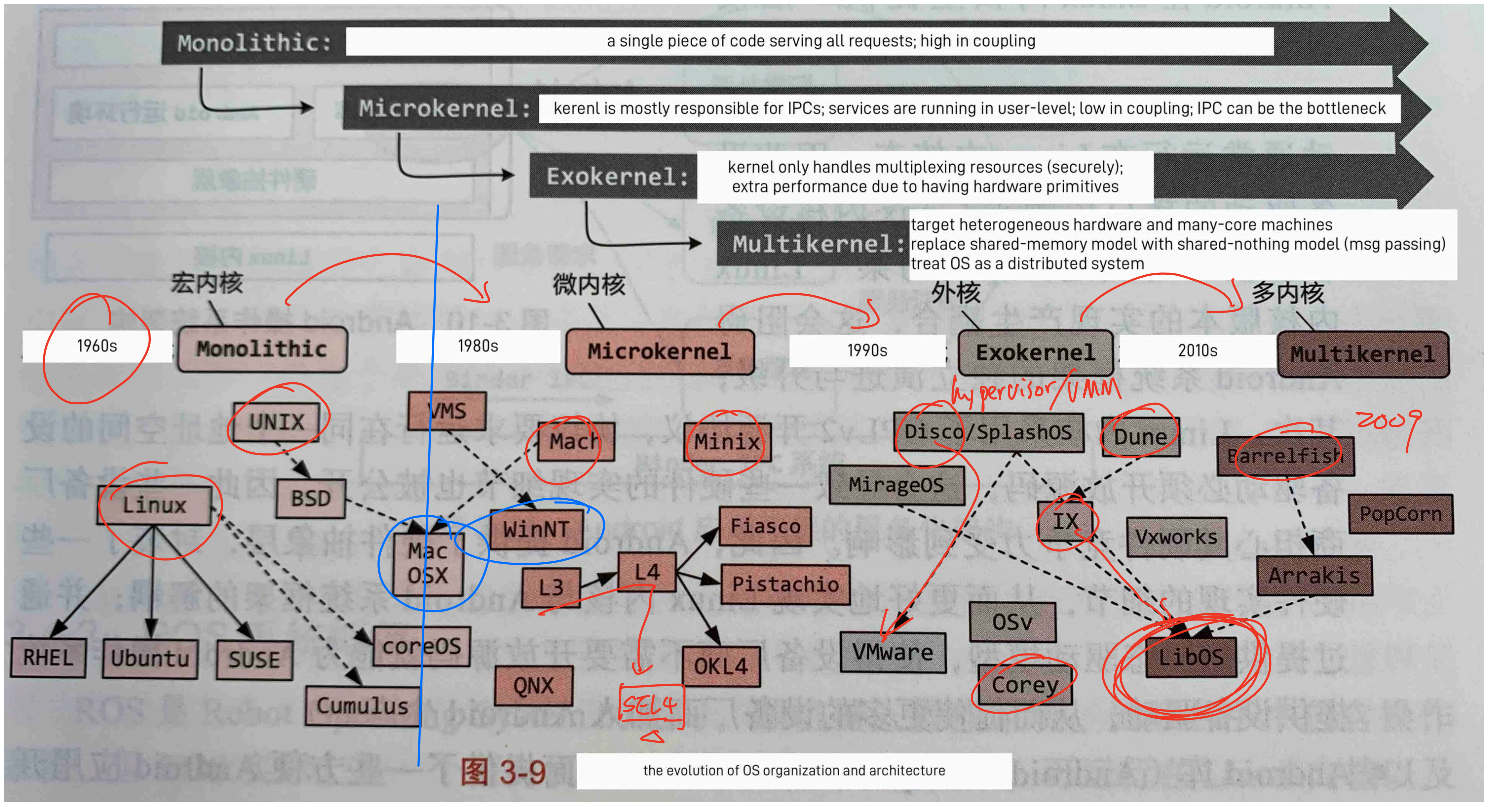
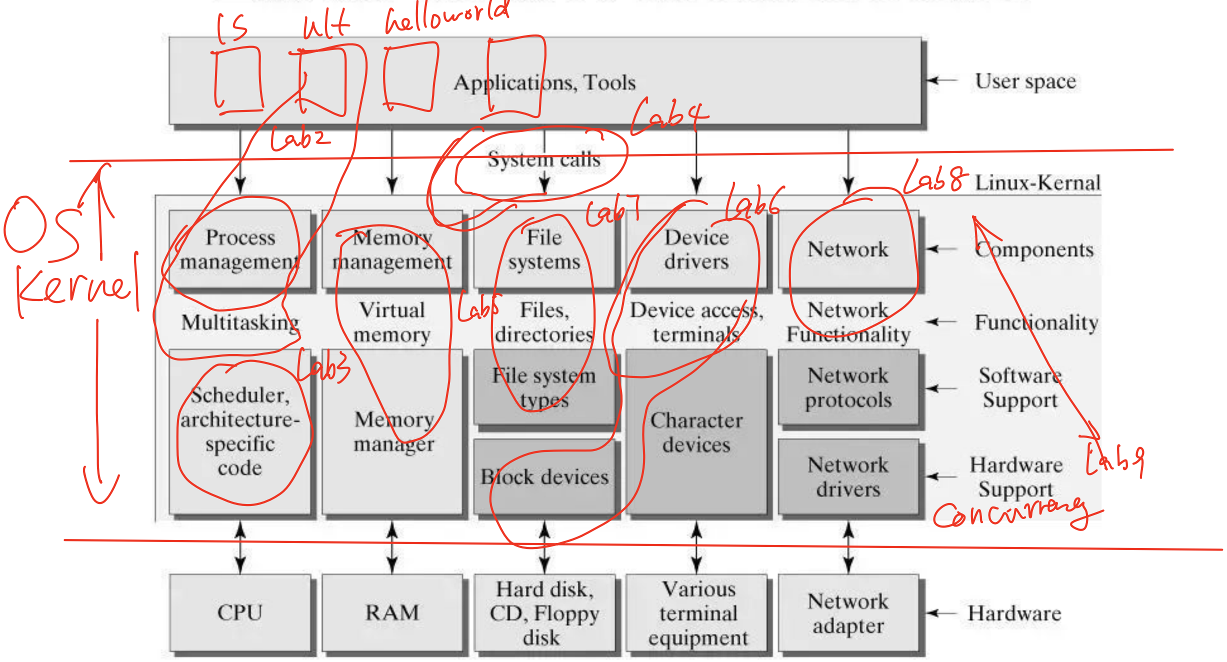
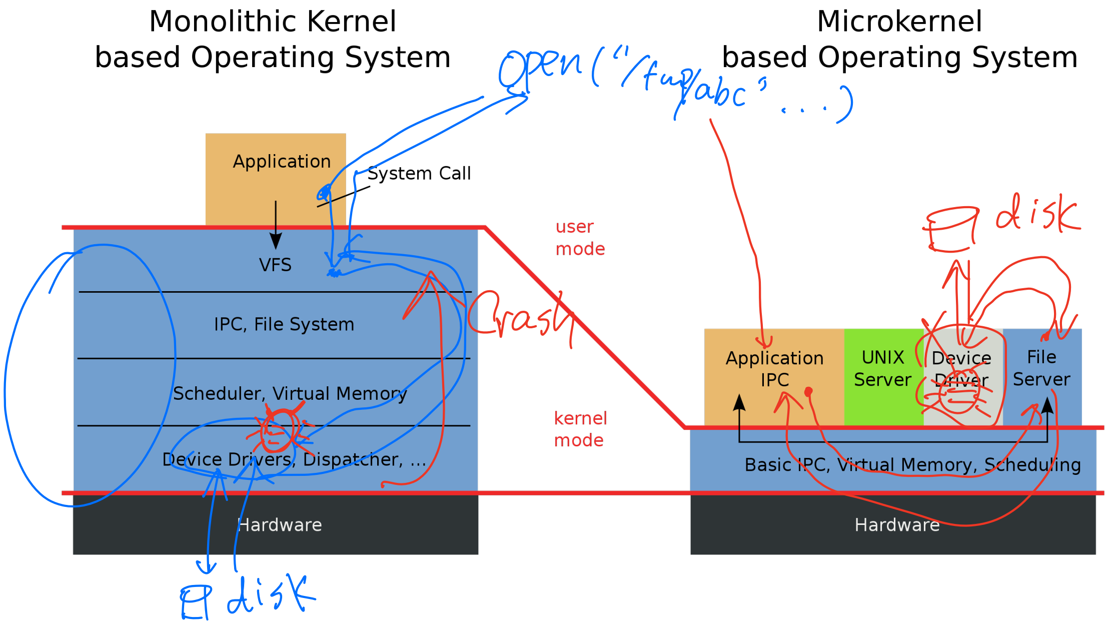
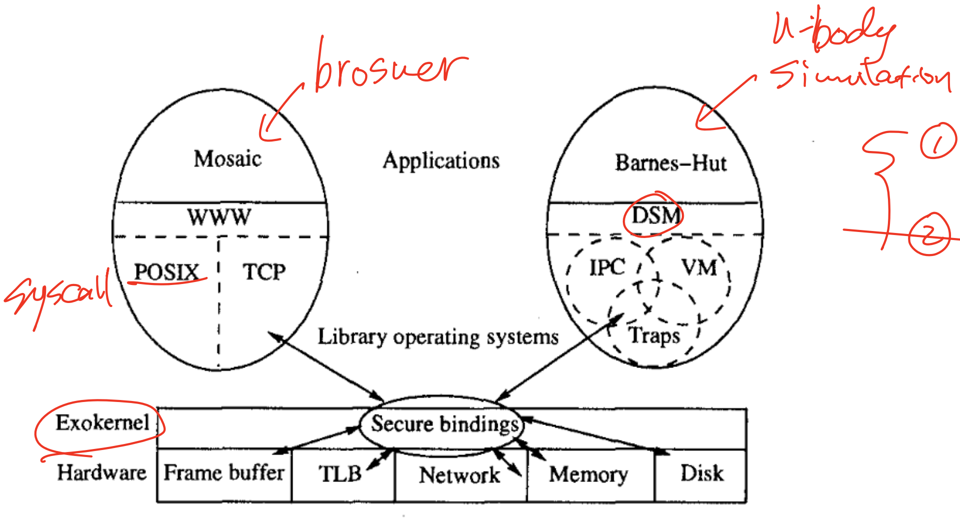

Monolithic:
a single piece of code serving all requests; high in coupling
Microkernel:
kerenl is mostly responsible for IPCs; services are running in user-level; low in coupling; IPC can be the bottleneck
Exokernel:
kernel only handles multiplexing resources (securely);
extra performance due to having hardware primitives
Multikernel:
target heterogeneous hardware and many-core machines
replace shared-memory model with shared-nothing model (msg passing)
treat OS as a distributed system
宏内核
微内核
外核
多内核
1960s
Monolithic
1980s
Microkernel
1990s
Exokernel
2010s
Multikernel
hypervisor/VMM
2009
UNIX
VMS
Mach
Minix
Disco/SplashOS
Dune
Barrelfish
BSD
MirageOS
PopCorn
Linux
WinNT
Mac OSX
Fiasco
IX
Vxworks
Arrakis
L3
L4
Pistachio
OSv
RHEL
Ubuntu
SUSE
coreOS
VMware
LibOS
Cumulus
QNX
OKL4
Corey
SEL4

图 3-9 the evolution of OS organization and architecture

Linux kernel architecture







HW
① managing Res

② ~~abstraction~~

Figure 1: An example exokernel-based system consisting of a thin exokernel veneer that exports resources to library operating systems through secure bindings. Each library operating system implements its own system objects and policies. Applications link against standard libraries (*e.g.*, WWW, POSIX, and TCP libraries for Web applications) or against specialized libraries (*e.g.*, a distributed shared memory library for parallel applications).

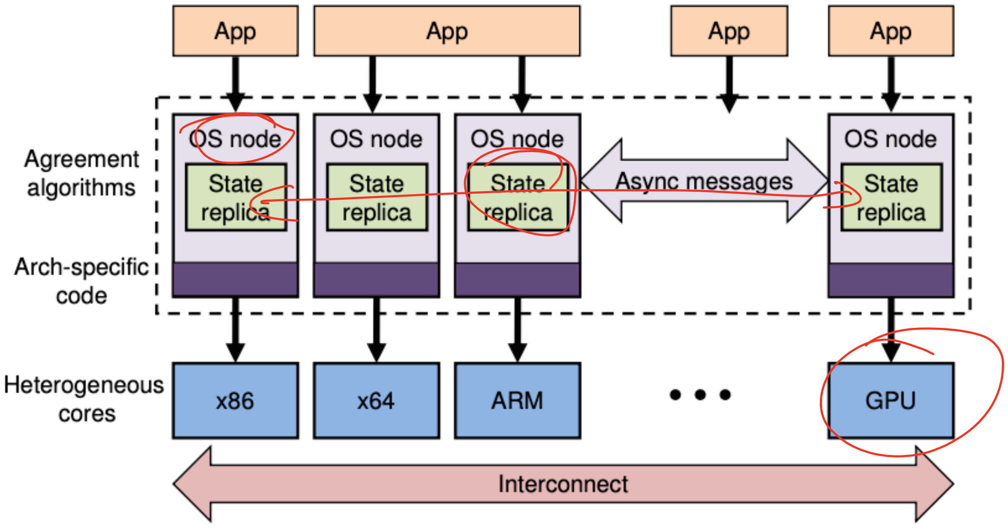
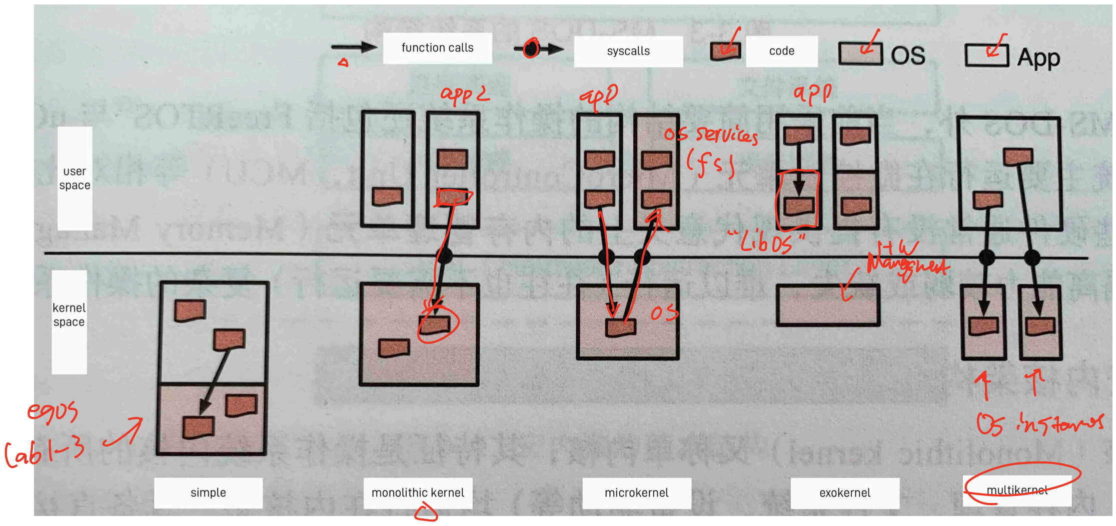


Figure 1: The multikernel model.

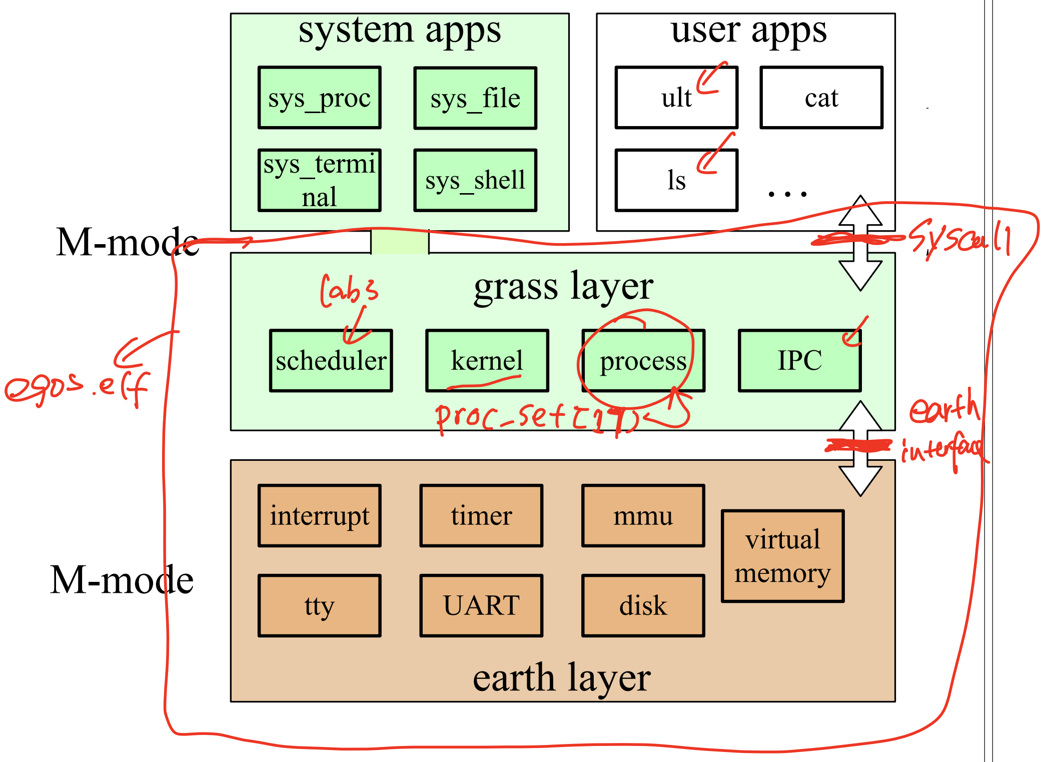
Shared Mem
vs.
Msg passing



* Improving IPC by kernel design, SOSP 1993
* Exokernel: An Operating System Architecture for Application-Level Resource Management, SOSP 1995
* The multikernel: a new OS architecture for scalable multicore systems, SOSP 2009
* Microkernel Goes General: Performance and Compatibility in the HongMeng Production Microkernel, OSDI 2024

CS6640, Week 04a

1. egos architecture



2. egos-2k+ booting process I

```
-------- Simulate on QEMU-RISCV --------
qemu-system-riscv32 -M virt -smp 4 -m 8M -bios tools/egos.bin -nographic
   -drive if=pflash,format=raw,unit=1,file=tools/qemuROM.bin
   -device sdhci-pci,addr=0x1
   -device sd-card,drive=MMC
   -drive if=none,file=tools/disk.img,format=raw,id=MMC
[CRITICAL] --- Booting on QEMU with core #2 ---
[SUCCESS] Finished initializing the tty and disk devices
[INFO] Use direct mode and put the address of the trap_entry into mtvec
[SUCCESS] Finished initializing the MMU, timer and interrupts
[SUCCESS] Enter the grass layer
[INFO] Load kernel process #1: sys_process
[INFO] Load 0x4400 bytes to 0x80200000
[INFO] Load 0x510 bytes to 0x80208000
[SUCCESS] Enter kernel process GPID_PROCESS
[INFO] Load kernel process #2: sys_terminal
[INFO] Load 0x3660 bytes to 0x80200000
[INFO] Load 0x274 bytes to 0x80208000
[SUCCESS] Enter kernel process GPID_TERMINAL
[INFO] sys_process receives: Finish GPID_TERMINAL initialization
[INFO] Load kernel process #3: sys_file
[INFO] Load 0x5040 bytes to 0x80200000
[INFO] Load 0x9c4 bytes to 0x80208000
[SUCCESS] Enter kernel process GPID_FILE
[INFO] sys_process receives: Finish GPID_FILE initialization
[INFO] Load kernel process #4: sys_shell
[INFO] Load 0x4da4 bytes to 0x80200000
[INFO] Load 0x89c bytes to 0x80208000
[CRITICAL] Welcome to the egos-2k+ shell!
➜ /home/cs6640 %
```