🎬 Operating Systems - Introductory Script

[Slide/Scene 1: Title Slide] Title: Introduction to Operating Systems
Subtitle: The Unsung Hero of Every Computer

[Scene 2: Analogy - Restaurant Manager 🍽️] 🎙️ “Think of your computer as a kitchen in a restaurant. You have to cook a lot of different dishes (i.e. run a lot of different programs) with a single set of shared resources. If you just had a wild pack of chefs without any management, it would be chaos. They'd be taking one another's ingredients and using one another's knives and pans. The food is coming out late (if at all), and when it does come out, it's all wrong. You order spaghetti and meatballs and get half a breakfast burrito and a bowl full of chicken bones. Meanwhile, the pantry is COMPLETELY out of onions and NO ONE has done the dishes.

Think of the OS as an executive chef who whips the staff into shape. He oversees what's going on, makes sure everything is in its right place, and tells the staff what to do. Now the kitchen assistants are chopping the right amount of onions, the line chefs aren't stealing one another's ingredients, and food comes out as intended and in a timely fashion. There's no shortage of ingredients because they're being ordered as needed, and the dishes are being cleaned quickly enough to keep up with the food that's going out.

In other words, you need an OS to manage how your programs work. Without direction, you have a horde of programs running amok across the shared hardware of your computer.

There’s a **manager** who takes your order, coordinates with the kitchen (the hardware), and ensures your food is served properly.

That’s exactly what an OS does — it’s the restaurant manager of your computer.”

[Scene 3: What is an OS?] 🎙️ “Let’s start with a simple question — *What is an Operating System, or OS?*

The OS is a **software that acts as an intermediary** between the **hardware** of a computer and the **software or user**.
In simpler words, it’s the **resource manager** of your computer.
It decides who gets what, when, and how."

[Scene 4: Functions of an OS] 🎙️ "Now let's understand some key functions of an OS:

1️⃣ It decides **which CPU** will execute **which app's instructions**.
2️⃣ It manages memory — deciding **what stays in RAM**, the faster memory.
3️⃣ And what will be stored on the **Hard Drive** — the long-term storage.

In short, it's responsible for **resource allocation**, **memory management**, and **process control**."

[Scene 5: 5 Quick MCQs 🧠] 🎙️ "Let's pause here for a quick check!
Try answering these 5 MCQs based on what we've covered so far."

1. What is the main purpose of an Operating System?
A. To compile user programs
B. To translate high-level code into machine code
C. To manage hardware and software resources ✅
D. To connect the computer to the internet

---

2. In the restaurant manager analogy, what role does the Operating System play?
A. The customer placing the order
B. The kitchen where food is prepared
C. The waiter delivering the food
D. The manager coordinating everything between customers and kitchen ✅

---

3. Which of the following is a key function of the Operating System?
A. Writing application software
B. Choosing which app gets CPU time ✅
C. Designing user interfaces
D. Installing antivirus software

---

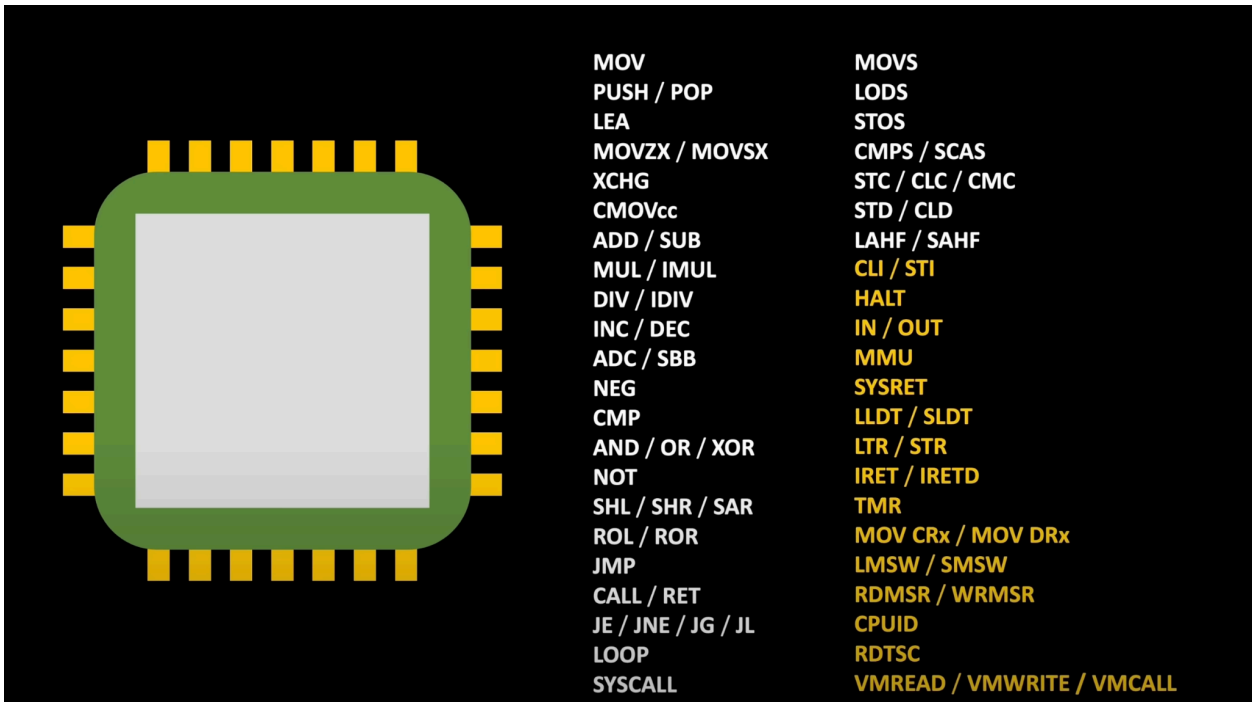
4. What does the Operating System use RAM for?
A. Permanent storage of files
B. Storing unused programs
C. Temporarily holding data and instructions currently in use ✅
D. Backing up user files

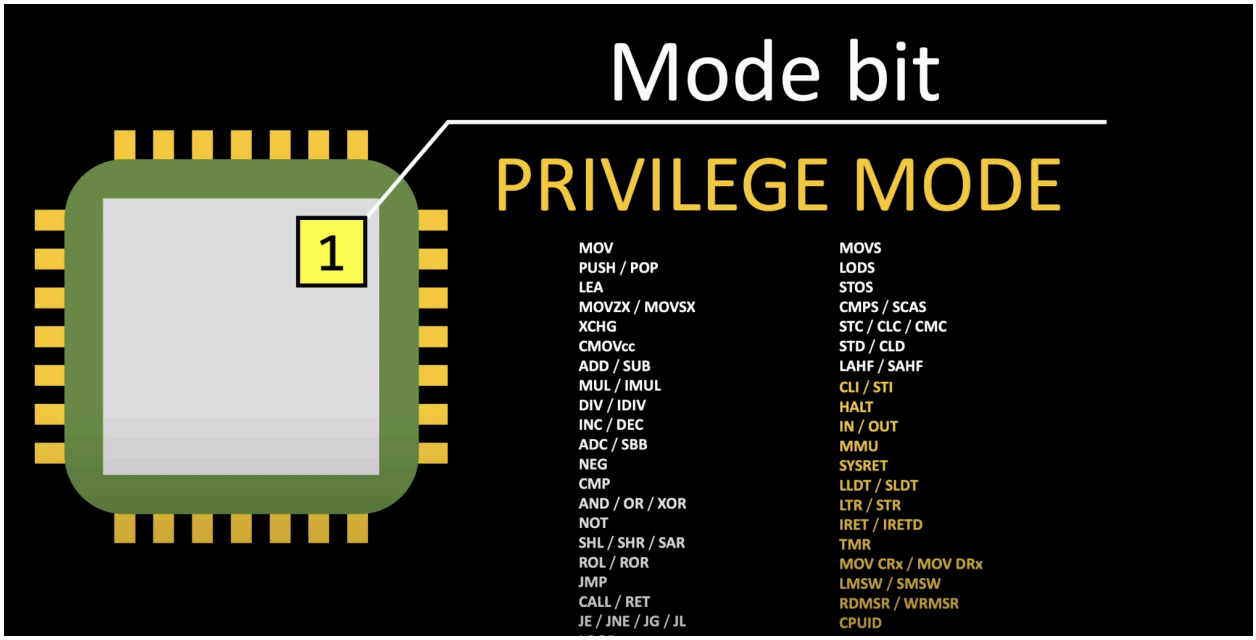
---

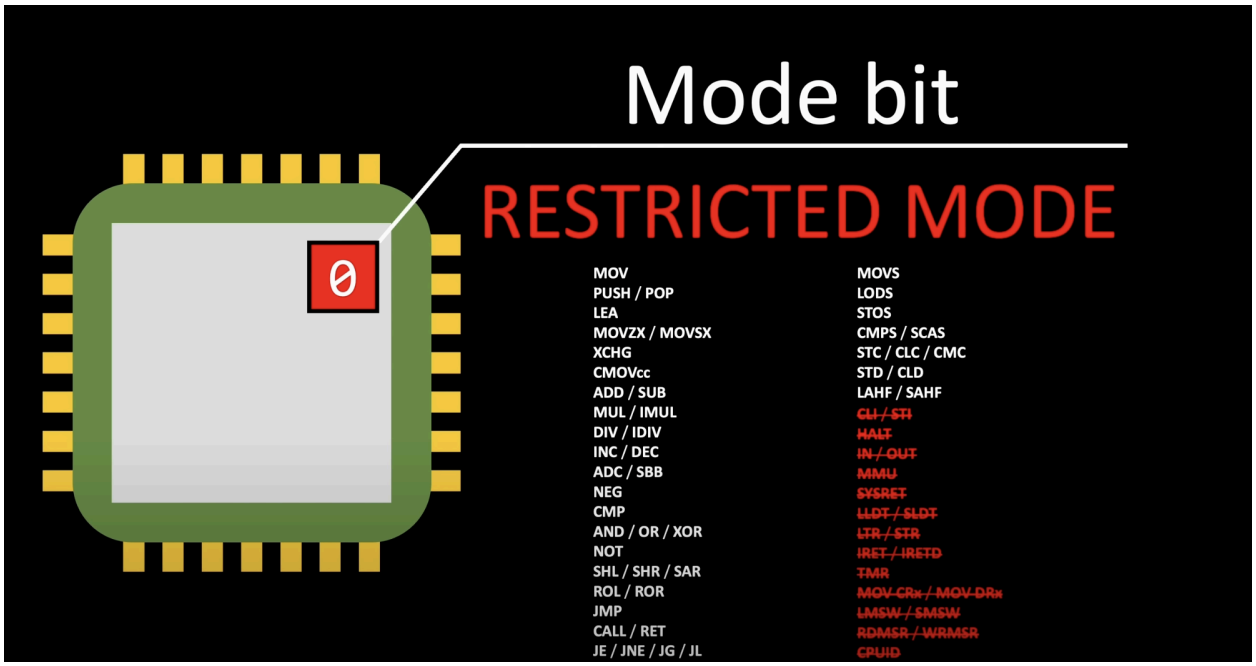
5. Which of these best defines the OS's role in resource management?

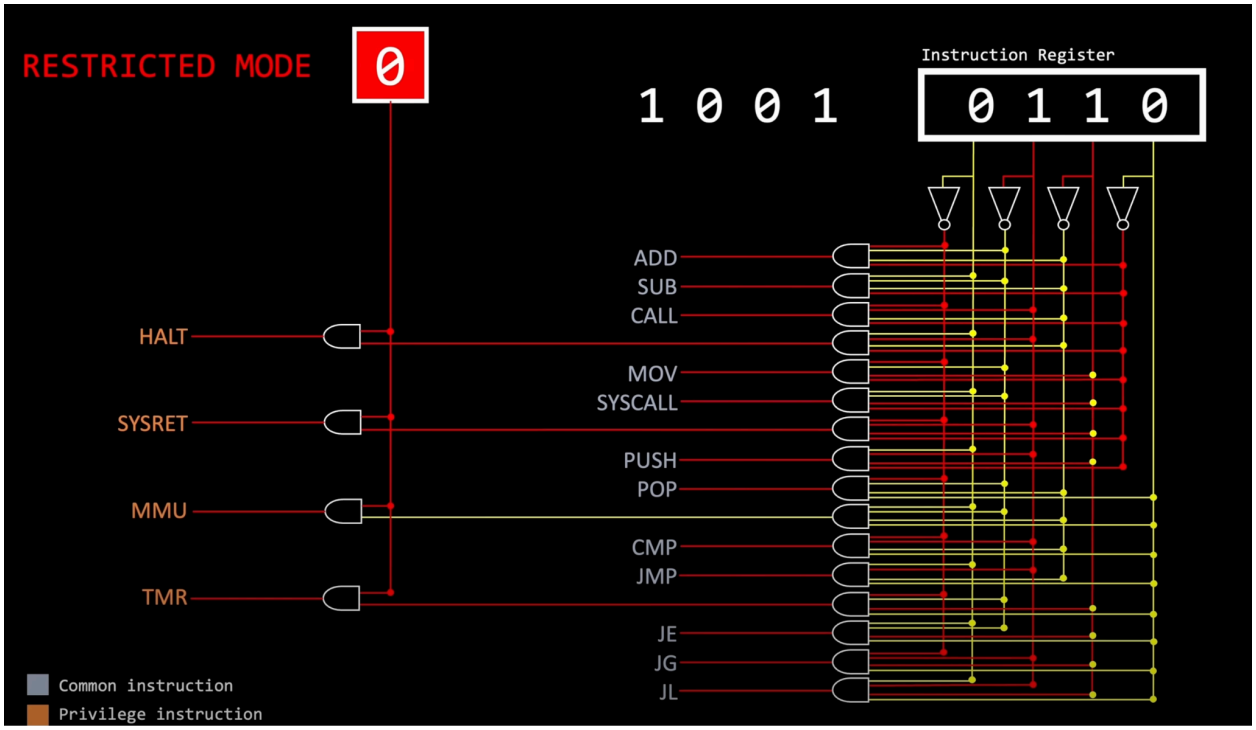
A. It manages the size of the computer screen
B. It decides how resources like CPU and memory are allocated to programs ✅
C. It upgrades system drivers automatically
D. It creates graphical animations for apps

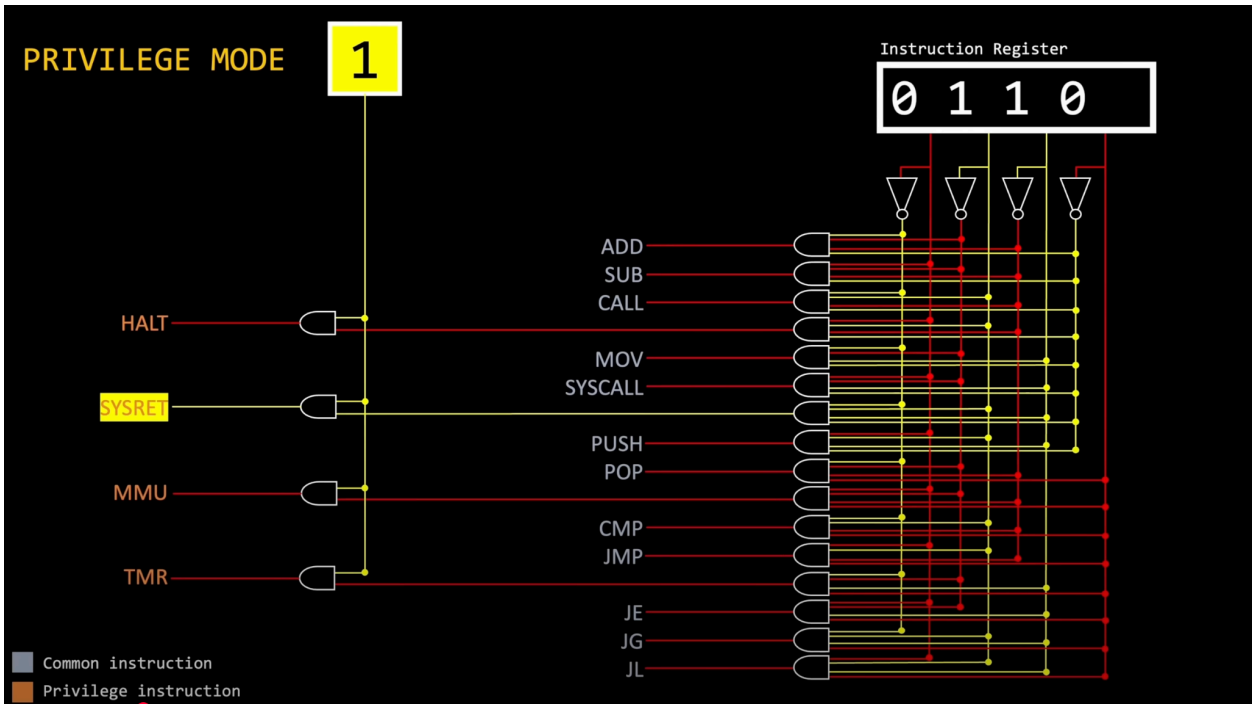
[Scene 6: How does the OS work?] 🎙️ "Now, let's dive deeper — *How does the OS actually work?*

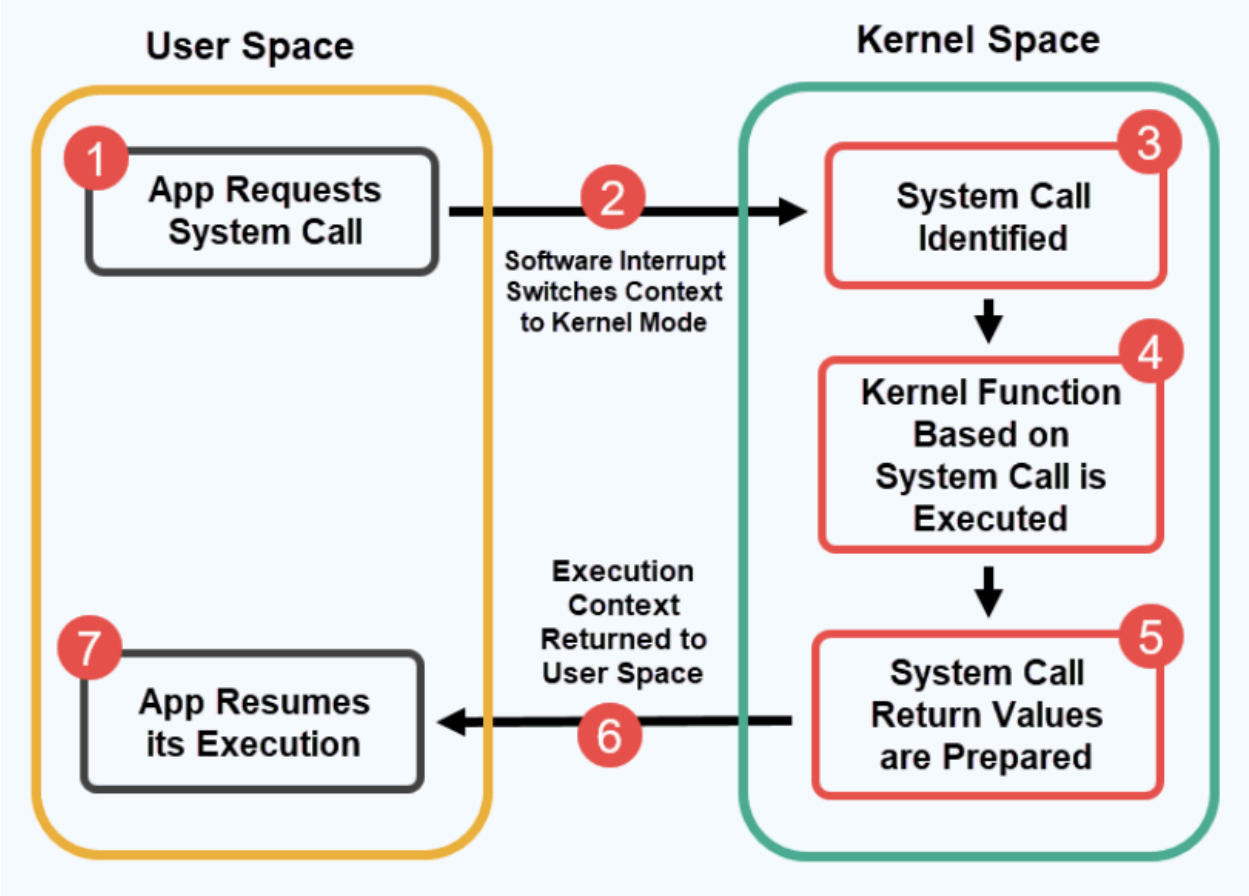



Mode bit
1
PRIVILEGE MODE
MOV
PUSH / POP
LEA
MOVZX / MOVSX
XCHG
CMOVcc
ADD / SUB
MUL / IMUL
DIV / IDIV
INC / DEC
ADC / SBB
NEG
CMP
AND / OR / XOR
NOT
SHL / SHR / SAR
ROL / ROR
JMP
CALL / RET
JE / JNE / JG / JL
MOVS
LODS
STOS
CMPS / SCAS
STC / CLC / CMC
STD / CLD
LAHF / SAHF
CLI / STI
HALT
IN / OUT
MMU
SYSRET
LLDT / SLDT
LTR / STR
IRET / IRETD
TMR
MOV CRx / MOV DRx
LMSW / SMSW
RDMSR / WRMSR
CPUID


Mode bit
0
RESTRICTED MODE
MOV
PUSH / POP
LEA
MOVZX / MOVSX
XCHG
CMOVcc
ADD / SUB
MUL / IMUL
DIV / IDIV
INC / DEC
ADC / SBB
NEG
CMP
AND / OR / XOR
NOT
SHL / SHR / SAR
ROL / ROR
JMP
CALL / RET
JE / JNE / JG / JL
MOVS
LODS
STOS
CMPS / SCAS
STC / CLC / CMC
STD / CLD
LAHF / SAHF
CLI / STI
HALT
IN / OUT
MMU
SYSRET
LLDT / SLDT
LTR / STR
IRET / IRETD
TMR
MOV CRx / MOV DRx
LMSW / SMSW
RDMSR / WRMSR
CPUID


RESTRICTED MODE
0
1 0 0 1
Instruction Register
0 1 1 0
ADD
SUB
CALL
MOV
SYSCALL
PUSH
POP
CMP
JMP
JE
JG
JL
HALT
SYSRET
MMU
TMR
Common instruction
Privilege instruction


PRIVILEGE MODE
1
Instruction Register
0 1 1 0
ADD
SUB
CALL
MOV
SYSCALL
PUSH
POP
CMP
JMP
JE
JG
JL
HALT
SYSRET
MMU
TMR
Common instruction
Privilege instruction



The key idea here is **Kernel Mode** vs **User Mode**.

- In **User Mode**, applications run with limited access to system resources.
- In **Kernel Mode**, the OS has complete control over the hardware.

This separation ensures **security and stability**."

📖 "Think of it like this — users can request things, but only the OS, running in kernel mode, has the authority to make it happen."

[Scene 7: Another 5 Quick MCQs 📋] 🎙️ "Alright, another round of 5 MCQs! This time based on OS working and modes."

1. **What is the major difference between Kernel Mode and User Mode?**
   A. Kernel Mode is faster than User Mode
   B. Kernel Mode has full access to hardware, User Mode has limited access ✅
   C. User Mode is used only for mobile devices

D. Kernel Mode is for web browsing

2. **Why is the separation between Kernel Mode and User Mode important?**
   A. It saves electricity
   B. It improves screen resolution
   C. It ensures security and stability ✅
   D. It speeds up Wi-Fi

3. **Which mode is typically used by regular applications like browsers or games?**
   A. Kernel Mode
   B. Hypervisor Mode
   C. Admin Mode
   D. User Mode ✅

4. **What enables an application in User Mode to request services from the OS?**
   A. App Store
   B. Task Manager
   C. System Call ✅
   D. BIOS

5. **If a program tries to access hardware directly without permission, what happens?**
   A. It speeds up
   B. It bypasses the OS
   C. It might crash or be denied access ✅
   D. It runs in safe mode

[Scene 8: System Calls 📞] 🎙️ "Whenever a program needs to do something privileged — like read a file or use the network — it makes a **System Call**.

Think of a system call as a **formal request from the user mode to the kernel mode**."

📊 *[Show Chart of Common System Calls here]*
(e.g., `read()`, `write()`, `fork()`, `exec()`, `open()`, etc.)

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

The following illustrates various equivalent system calls for Windows and UNIX operating systems.

	Windows	Unix

Process control	`CreateProcess()`
	`ExitProcess()`
	`WaitForSingleObject()`
		`fork()`
		`exit()`
		`wait()`

File management	`CreateFile()`
	`ReadFile()`
	`WriteFile()`
	`CloseHandle()`
		`open()`
		`read()`
		`write()`
		`close()`

Device management	`SetConsoleMode()`
	`ReadConsole()`
	`WriteConsole()`
		`ioctl()`
		`read()`
		`write()`

Information maintenance	`GetCurrentProcessID()`
	`SetTimer()`
	`Sleep()`
		`getpid()`
		`alarm()`
		`sleep()`

Communications	`CreatePipe()`
	`CreateFileMapping()`
	`MapViewOfFile()`
		`pipe()`
		`shm_open()`
		`mmap()`

Protection	`SetFileSecurity()`
	`InitlializeSecurityDescriptor()`
	`SetSecurityDescriptorGroup()`
		`chmod()`
		`umask()`
		`chown()`

[Scene 9: Wrap-up] 🎙️ “So far, we've looked at what an OS is, why it's important, how it functions, and how it acts as the brain behind everything running on a computer.

We’ve also seen the importance of system calls and how the OS bridges user requests to hardware execution.”

💡 “Next up: We’ll explore how multitasking and scheduling work!”