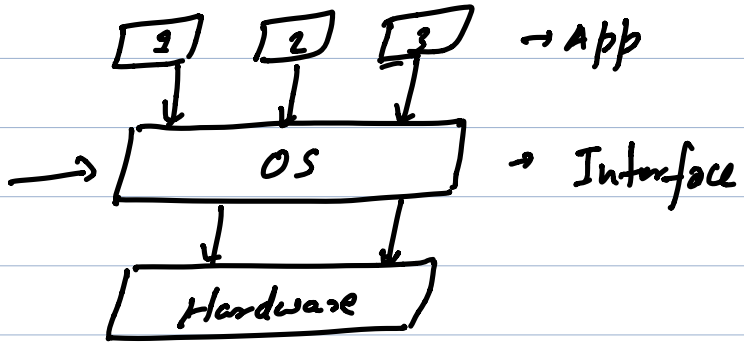
System Calls

1. What is a System Call ? ✓
2. Why system calls are needed ? ✓
3. How System Call works ? ✓
4. Examples of system Calls ✓

System Call, :



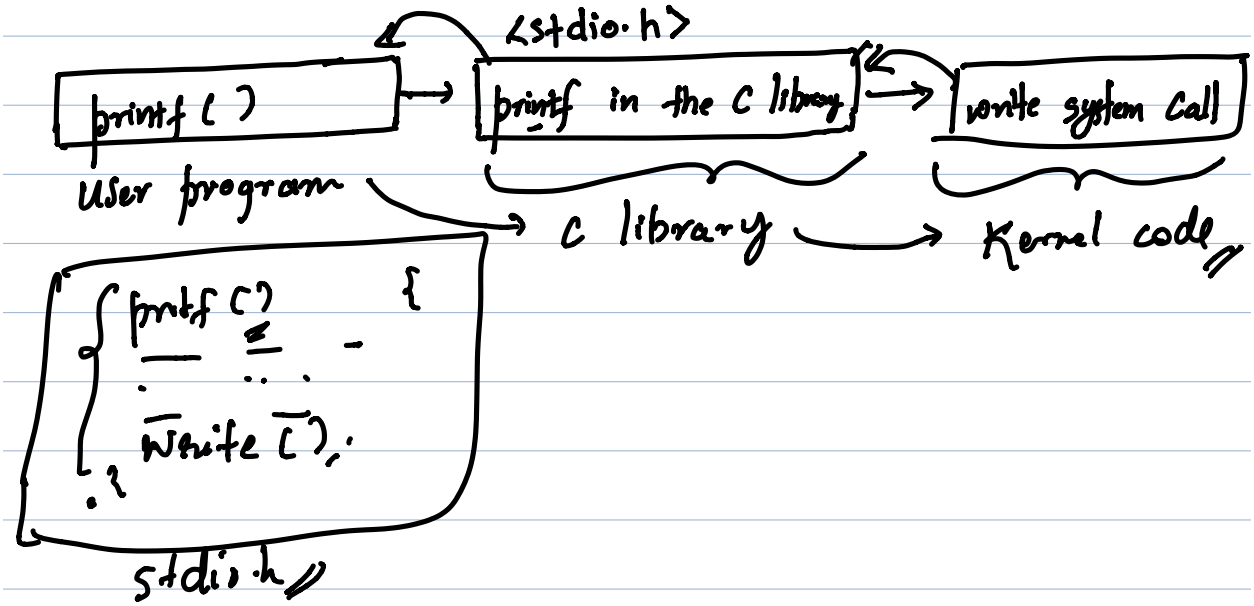
How does the app talks to the OS ?

System Calls: These provide a layer between the user program and the hardware

test.c

```
#include<stdio.h>
printf("hello");
```

→ Screen



Application programing Interface (API):

Programing interface used by user program:

C library printf() →

Syst calls or Com Syst Calls.

→ printf()

→ Sort()

→ Syst calls → write() ←:

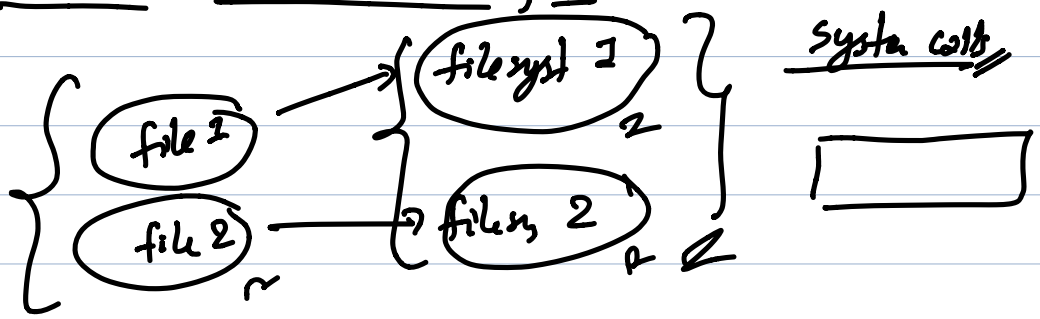
→ printf() → windu → w2

→ driver → write()

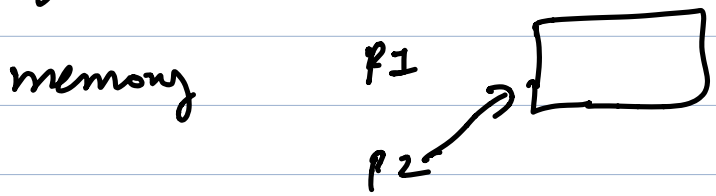
→ write("hello")

Why System Calls ?

i) abstract hardware interface



ii) Security and stability :



Working of system Calls :-

* User program cannot execute Kernel code directly.
* User program must signal to the Kernel that they want to execute a system call.
* The execution should then switch to the Kernel mode from user mode.

User mode → Kernel mode

How this switch can happen?

printf() ⟶ write()
C library

Interrupts:

It suspends the current proc on a processor and the processor then executes an interrupt handler.

Software: Divide by Zero:
Hardware:

Interrupt handler: Kernel mode

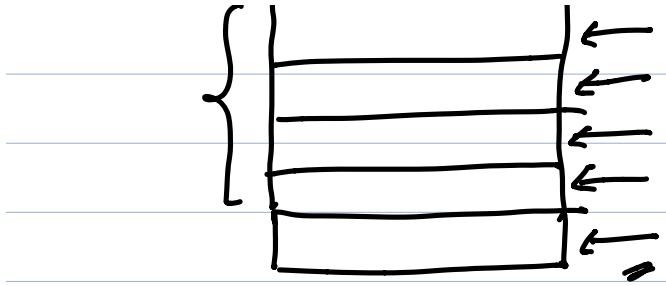
Cause some software interrupt: System call → 128

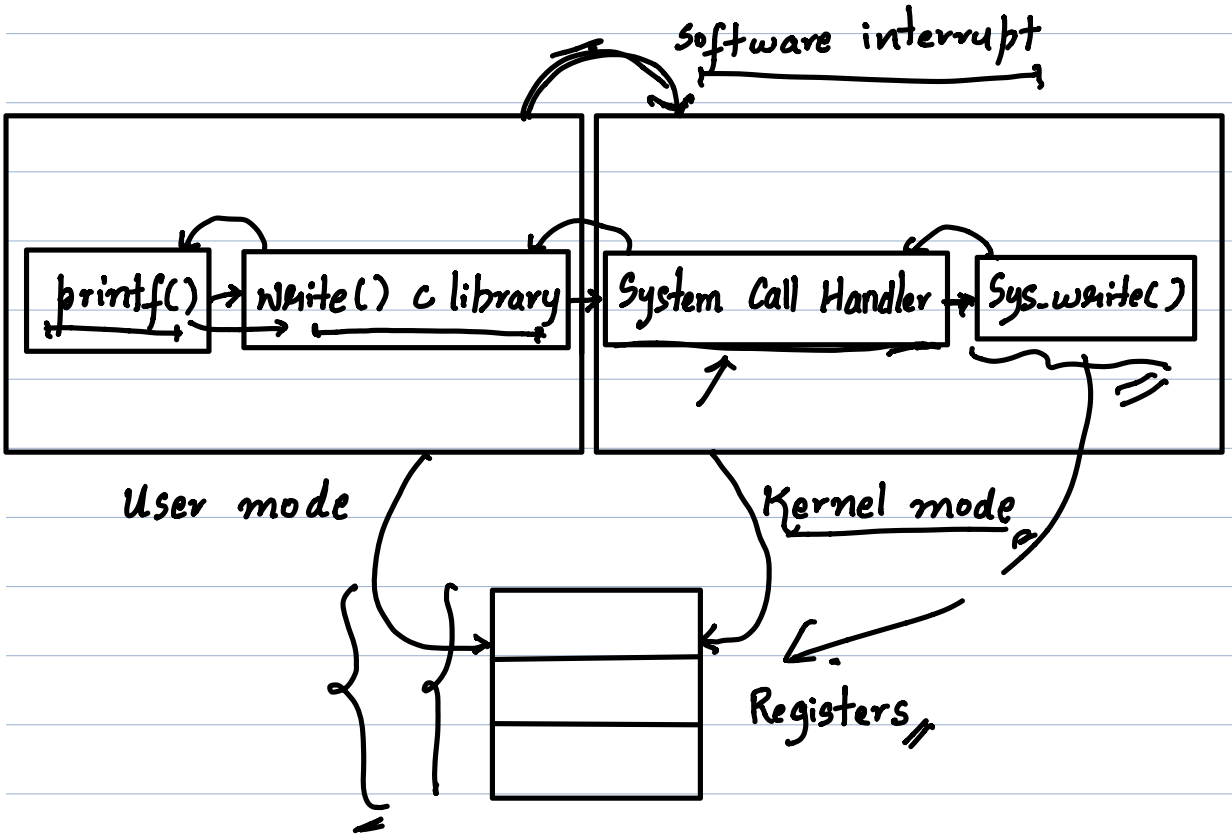
→ System Call Handler

→ Sysytk Call: Unique Number.

write (" ").

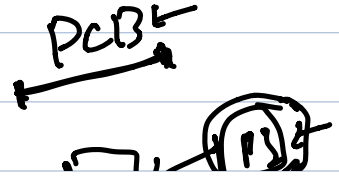
x ← Syscall number

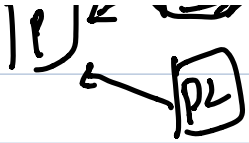




Does Context switch happens during system call ?

* Context switch : Storing the state of a process before Switch happens
  - Register values
  - Program Counter





No Context switch happens

{ Write() → Blocking Call
Read() → }

waiting state

Context swit → Block Syste Calls

Examples :-

	Windows	Linux
Process Control	CreateProcess(),	fork(),
	ExitProcess(),	exit(),
	WaitForSingleObject(),	wait(),
File Manipulation	CreateFile(),	open()
	ReadFile(),	read()
	WriteFile(),	write()