System Calls

Zilogic Systems

1. Kernel and Syscalls

- With bare hardware, we can work with instructions, memory locations, hard disk sectors.
- But with the OS kernel running on the hardware, we are provided with higher level abstractions like files, processes, sockets, etc.
- The kernel provides functions to create and manipulate these resources. These functions are called system calls.

2. Process Memory Map

- Every process is associated with instructions to execute (code) and global data, stack and heap (data).
- The kernel also has associated code and data.
- The process has to execute functions inside the kernel to manipulate the abstractions provided by the kernel - files, processes, sockets and other resources.
- One way to organize the memory map of the process is to map the user code, user data, kernel code and kernel data into the process virtual address space.
- The top 3GB of virtual address space is used for the mapping the user code and data and the bottom 1GB of virtual address space is used for mapping the kernel code and data.
- When the scheduler switches between various processes the mapping for top 3GB of virtual address will keep switching to the corresponding process code and data, but the bottom 1GB will be always mapped to the same kernel code and data.
- When the process wants to execute the system call, it can directly branch to the kernel function.

3. Protecting the Kernel

- But with this setup, the process can now modify the kernel code and data at will. This can result in jeopardizing the stability of the system.
- The hardware provides the following features to overcome this.
  1. Protection Rings.
  2. Ring 0 pages.
  3. Mechanism to switch to Ring 0.

3.1. Protection Rings

- The processor can operate in two protection rings - Ring 0 and Ring 1.
- In ring 0, the processor can execute any instruction, and can access any memory location.
- In ring 1, the processor cannot execute certain instructions, and cannot access certain memory locations. For example the processor cannot execute instructions that modify the page tables.
- The kernel code executes in ring 0, and user code executes in Ring 1.

3.2. Ring 0 Pages

- The page tables has a flag that indicate whether a page is a Ring 0 page.

- Ring 0 pages can be accessed only when the processor is in Ring 0.
- Ring 1 pages can be accessed when the processor is in either Ring 0 or Ring 1.
- The pages corresponding to user code and data are indicated as Ring 1 pages.
- The pages corresponding to kernel code and data are indicated as Ring 0 pages.

3.3. Entering Ring 0

- The processor switches to ring 0 under only one circumstance - when an interrupt occurs.
- When an interrupt occurs the processor switches to ring 0 and transfers control to the preset address.
- The preset address generally contains kernel code that handles the interrupts.
- Interrupts can occurs due to hardware devices asserting the interrupt line - these are called hardware interrupts.
- Interrupts can also be triggered using an instruction, these are called software interrupts or traps.
- When a trap is executed, the processor switches to ring 0 and transfers control to another preset address.

4. Implementing System Calls

- Each system call is given a unique no.
- When the user process wants to execute a system call, it stores the system call no. in a specific CPU register and triggers a software interrupt.
- The processor switches to ring 0, and starts executing the trap handler. The trap handler is part of the kernel code.
- The trap handler uses the system call no. to index into system call table, and then branches to the specified system call.
- Thus we are able achieve controlled access to the system calls, through the protection rings and the trap mechanism.